

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КПІ ім. І. СІКОРСЬКОГО»

Основи програмування

МЕТОДИЧНІ ВКАЗІВКИ
до виконання комп'ютерних практикумів

Київ – 2017

Міністерство освіти і науки України
Національний технічний університет України
«КПІ ім. І. Сікорського»

Основи програмування

Методичні вказівки

до виконання комп'ютерних практикумів
для студентів напряму підготовки 6.050101 – "Комп'ютерні науки"

*Рекомендовано вченою радою
факультету біомедичної інженерії НТУУ «КПІ ім. І. Сікорського»*

Київ
НТУУ «КПІ ім. І. Сікорського»
2017

Основи програмування: методичні вказівки до виконання комп'ютерних практикумів з дисципліни «Основи програмування». Основи програмування мовою Python. / Уклад.: А. В. Яковенко. – К.: НТУУ «КПІ ім. І. Сікорського», 2017. – 87 с.

*Рекомендовано Вченою радою ФБМІ НТУУ «КПІ ім. І. Сікорського»
(Протокол № 11 від 24.05.2017 р.)*

Навчально-методичне видання

Основи програмування

Методичні вказівки

до виконання комп'ютерних практикумів
для студентів напряму підготовки 6.050101 – "Комп'ютерні науки"

Укладачі:	<i>А. В. Яковенко, к.т.н.</i>
Відповідальний редактор:	<i>Носовець Олена Костянтинівна, к.т.н.</i>
Рецензент:	<i>Юрчишин Василь Якович, с.н.с, к.т.н., доцент</i>

ЗМІСТ

Вступ	5
1. Комп'ютерний практикум №1. Внутрішнє машинне представлення цілих та дійсних чисел.....	7
2. Комп'ютерний практикум №2. Розробка блок-схеми алгоритму вирішення задачі.....	11
3. Комп'ютерний практикум №3. Робота в інтегрованому середовищі розробки. Типи даних. Поняття змінної.....	16
4. Комп'ютерний практикум №4. Логічні вирази і логічний тип даних. Умовний оператор. Інструкція if.....	25
5. Комп'ютерний практикум №5. Універсальний організатор циклу – інструкція while.....	32
6. Комп'ютерний практикум №6. Списки. Інструкція for.....	36
7. Комп'ютерний практикум №7. Генерування випадкових чисел. Робота з одновимірними масивами.....	44
8. Комп'ютерний практикум №8. Розробка програм з використанням двовимірних масивів.....	54
9. Комп'ютерний практикум №9. Рядки як послідовності символів. Кортежі. Словники.....	61
10. Комп'ютерний практикум №10. Функції. Модулі.....	72
11. Додаток А. Критерії оформлення звіту за результатами виконання комп'ютерного практикуму	85
Література	87

ВСТУП

Предмет навчальної дисципліни «Основи програмування» – процес навчання і підготовки зі спеціальності 122 «Комп'ютерні науки та інформаційні технології» за спеціалізацією «Інформаційні технології в біології та медицині» першого (бакалаврського) рівня вищої освіти ступеня бакалавра, який дозволить використовувати основні знання основ програмування при розробці інформаційних технологій, що зараз охоплюють майже всі сфери життя і діяльності та відіграють важливу роль в біомедичних застосуваннях.

В структурно-логічній схемі програми підготовки фахівця:

– дисципліну **забезпечують** наступні дисципліни та кредитні модулі: дисципліна – тільки розпочинається в циклі та є основою для подальшого вивчення інших дисциплін;

– дисципліна **забезпечує** наступні навчальні дисципліни та кредитні модулі: «Проектування та аналіз обчислювальних алгоритмів» (5/І), «Алгоритмізація та програмування» (1/ІІ), «Об'єктно-орієнтоване програмування» (2/ІІ).

Метою навчальної дисципліни є формування у студентів **здатностей**:

- Здатність застосовувати професійні знання й уміння на практиці СК-2
- Здатність гнучко адаптуватися до різних професійних ситуацій, проявляти творчий підхід, ініціативу СК-3
- Здатність аналізувати проблеми, ставити постановку цілей і завдань, виконувати вибір способу й методів дослідження, а також оцінку його якості СК-5
- Здатність вирішувати проблеми в професійній діяльності на основі аналізу й синтезу ІК-2
- Здатність застосовувати сучасні парадигми програмування під час програмної реалізації професійних задач ПК-11

Основні завдання навчальної дисципліни.

Згідно з вимогами освітньо-професійної програми студенти після засвоєння навчальної дисципліни мають продемонструвати такі результати навчання:

знання:

- основ наукової та дослідницької діяльності;
- математичних та природничих наук, в тому числі з інтелектуального аналізу даних, технологій видобування інформації;
- інформаційних технологій, мов програмування, інструментарію програміста;

- мов програмування, сучасних теорій організації баз даних та знань, методів і технологій їх розробки;
- парадигм програмування, сучасних мов програмування, основних структур даних і алгоритмів;

вміння:

- використовувати довідкову літературу, технічну документацію;
- застосовувати знання фундаментальних дисциплін для розв'язку професійних задач;
- сприймати, усвідомлювати та передавати інформацію у повному обсязі без спотворення змісту;
- застосовувати мови програмування, мови опису інформаційних технологій, мови специфікацій;
- застосовувати інструментальні засоби при проектуванні та створенні інформаційних систем, продуктів і сервісів інформаційних технологій;
- застосовувати ефективні алгоритми для розв'язання професійних завдань;

досвід в системі типових завдань діяльності:

- визначати структуру даних для забезпечення інформаційних потреб цієї моделі;
- розробляти програмний код комп'ютерної програми реалізації та налагоджувати розроблену програму.

При виконанні комп'ютерних практикумів, студенти мають здобути навички розробки та тестування простих програм із застосуванням мови Python, самостійної розробки та створення програм, які реалізують різні засоби використання типів даних високого рівня та сучасних методів керування програмним процесом. Методичні рекомендації розроблені для проведення занять за допомогою частково-пошукового методу, який сприяє активному пошуку розв'язання поставлених задач та продуктивному аналізу одержаних результатів, для студентів зі спеціальності 122 «Комп'ютерні науки та інформаційні технології» за спеціалізацією «Інформаційні технології в біології та медицині» першого (бакалаврського) рівня вищої освіти ступеня бакалавра факультету біомедичної інженерії.

В методичних вказівках практикум побудований таким чином, що студент повторно ознайомлюється з теоретичними відомостями відповідно до зазначеної теми, може проглянути приклад виконання типових задач для розв'язання поставленої та проаналізувати одержані результати. Для самоконтролю студенти даного курсу дисципліни можуть скористуватись питаннями для самоконтролю, що забезпечить повторення та закріплення пройденого матеріалу.

Комп'ютерний практикум №1. Внутрішнє машинне представлення цілих та дійсних чисел

Мета роботи: ознайомлення зі правилами переведення цілих та дійсних чисел з однієї системи числення в іншу.

Завдання

Проаналізувати варіант завдання. За даними варіантів завдань отримати внутрішнє представлення:

Варіанти завдань:

№	цілого числа без знаку в однобайтовій комірці пам'яті	цілого числа в 2-х байтовій комірці. Записати відповідь у 16-ій формі	Представити число в двійковій системі числення в експоненційному нормалізованому вигляді
1	129	1687 -8542	125,627
2	456	1597 -5687	159,235
3	423	4236 -1667	753,457
4	789	7524 -2546	851,953
5	159	7458 -9874	153,859
6	267	9532 -6523	753,159
7	357	8542 -9832	456,754
8	459	6578 -4596	159,753
9	852	2154 -7425	426,751
10	345	7596 -5623	953,751
11	259	7845 -8742	759,153
12	531	3256 -3657	698,412
13	214	9888 -8563	325,785
14	742	5236 -4253	254,145
15	853	4523 -7425	741,852
16	963	7845 -8632	963,258
17	751	6587 -7823	745,352
18	862	9832 -8523	752,154
19	953	1245 -9632	985,425
20	842	1597 -9874	965,742
21	751	3254 -5236	653,145
22	793	9863 -6497	751,425
23	315	7863 -9713	854,164
24	934	9632 -5612	856,235
25	468	7412 -4555	754,256

Теоретичні відомості

ПРЕДСТАВЛЕННЯ ЦІЛИХ ЧИСЕЛ

Цілі числа в комп'ютері зберігаються в пам'яті у форматі з фіксованою комою або фіксованою точкою. У цьому випадку кожному розряду комірки пам'яті відповідає завжди один і той же розряд числа, а кома знаходиться праворуч після молодшого розряду, тобто поза розрядної сітки.

Множина цілих чисел, які представлені в пам'яті ЕОМ обмежена і залежить від розміру комірок пам'яті (машинного слова), які використовуються для їх зберігання. В k-розрядній комірці може зберігатися 2^k (65536) різних значень цілих чисел.

Взагалі, розряди нумеруються справа наліво, починаючи з 0. Нижче показана нумерація біт в двухбайтовому машинному слові.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

В даній комірці можна зберігати 2^{16} цілих чисел. Діапазон допустимих значень залежить також від того, чи будуть в комірці зберігатися числа зі знаком чи без знаку.

Для запису цілого числа без знаку досить перевести його в двійкову систему числення і при необхідності доповнити результат зліва незначущими нулями.

Приклад 1

Отримати внутрішнє представлення цілого числа 129 без знаку в однобайтовій комірці пам'яті.

Рішення:

1) $129_{10} = 10000001_2$

2) 1000 0001

Приклад 2

Дано внутрішнє машинне представлення числа в однобайтовому машинному слові – 9C. Визначити, що це за число.

Рішення:

1) $9C_{16} = 1001\ 1100_2$

2) $2^7 + 2^4 + 2^3 + 2^2 = 128 + 16 + 8 + 4 = 156_{10}$

Щоб записати внутрішнє машинне подання цілого числа зі знаком, необхідно знайти його доповняльний код.

Подання цілих додатніх чисел

Додатковий код цілого додатнього числа збігається з його прямим кодом. Для його отримання необхідно:

1. Перевести число N в двійкову систему числення.

2. Отриманий результат доповнити зліва незначущими нулями до k розрядів.

3. При необхідності перевести число в стисло шістнадцяткову форму.

Приклад 3

Отримати внутрішнє представлення цілого числа 1607 в 2-х байтовій комірці пам'яті. Записати відповідь в 16-ій формі.

Рішення:

1) $1607_{10} = 11001000111_2$

2) Внутрішнє представлення цього числа: 0000 0110 0100 0111

3) 16-а форма: 0647.

Подання цілих від'ємних чисел

Для представлення від'ємних чисел використовується доповняльний код. Доповняльний код дозволяє замінити арифметичну операцію віднімання операцією додавання, що істотно спрощує роботу процесора і збільшує його швидкодію.

Алгоритм отримання внутрішнього представлення цілого від'ємного числа N , що зберігається в k -розрядному машинному слові:

1. Отримати внутрішнє представлення додатнього числа N .
2. Отримати зворотний код цього числа заміною 0 на 1 і 1 на 0, тобто значення всіх біт інвертувати.
3. До отриманого числа додати 1 (отримати доповняльний код).
4. При необхідності записати стисле внутрішнє машинне подання.

Приклад 4

Отримати внутрішнє представлення цілого числа -1607 в 2-х байтовій комірці пам'яті. Записати відповідь в 16-ій формі.

Рішення:

- 1) Внутрішнє представлення цього додатнього числа: 0000 0110 0100 0111
- 2) Зворотний код – 1111 1001 1011 1000
- 3) доповняльний код – 1111 1001 1011 1001
- 4) Стислий 16-ий код – F9B9

У разі подання величини зі знаком найлівіший (старший) розряд вказує на позитивне число, якщо містить нуль, і на негативне, якщо - одиницю. Це необхідно враховувати при зворотному перекладі.

Приклад 5

Дано стисле 16-е внутрішнє представлення числа – CF18. Визначити, що це за число.

Рішення:

- 1) CF18 = 1100 1111 0001 1000
- 2) Число від'ємне, оскільки старший розряд дорівнює 1, тому отримуємо зворотній код – 1100 1111 0001 0111 (відняти 1)
- 3) Прямий код – 0011 0000 1110 1000
- 4) $30E8_{16} = 12520_{10}$

ПРЕДСТАВЛЕННЯ ДІЙСНИХ ЧИСЕЛ

Для подання дійсних чисел у пам'яті комп'ютера був розроблений стандарт **IEEE 745**. Він використовується багатьма мікропроцесорами і програмними засобами.

Нормалізована і денормалізована форма дійсних чисел

Дійсні числа в пам'яті комп'ютера представлені в форматі з плаваючою десятковою комою (**експоненційній формі**).

Будь-яке число A може бути представлено в експоненційній формі:

$$A = m \cdot q^N,$$

де m - мантиса числа, q - основа системи числення, N - порядок числа.

Наприклад: $555,55 = 0,55555 \cdot 10^3$

Так як варіантів представлення одного і того ж числа в експоненційній формі безліч, то для внутрішнього машинного представлення домовилися представляти числа у **нормалізованій або денормалізованій формі**.

У **денормалізованій** формі мантиса повинна відповідати такій умові: вона повинна бути правильним дробом і мати після коми цифру, відмінну від нуля, тобто $1/n \leq |m| < 1$

Наприклад, 555,55 – звичайна форма, $0,55555 \cdot 10^3$ – денормалізована форма. Це стосується і від'ємних чисел, тому що мантиса в умові взята по модулю.

Приклад 6

Записати числа в денормалізованій формі (в десятичному виді):

а) 217,93410 б) 7532110 в) 10,010110 г) 20045010

Нормалізована мантиса містить свій старший біт зліва від точки. У загальному випадку мантиса повинна задовольняти умові $1 \leq |m| < 10$. Так як числа представлені в двійковому вигляді, то цей біт завжди дорівнює 1. Іншими словами нормалізована мантиса належить інтервалу $1 \leq |m| < 2$. У пам'яті машини цей біт не зберігати, тобто є "прихованим".

Для додатніх і від'ємних чисел мантиса в пам'яті представлена в **прямому коді**.

Приклад 7

Представити число 155,625 у двійковій системі числення в експоненційному нормалізованому вигляді.

$155,625_{10} = 10011011,101_2$ - число у двійковій системі

Наведемо отримане число до нормалізованого виду в десятковій і двійковій системах:

$$1,55625 \cdot 10^2 = 1,0011011101 \cdot 2^{11}$$

В результаті отримали основні складові експоненційного нормалізованого двійкового числа:

Мантису $m = 1,0011011101$

Експоненту $\text{exp}_2 = +111$

Контрольні запитання:

1. Що прийнято називати основою системи числення?
2. Що є доповненням числа 0 у двійковій системі числення?
3. Що є основою двійкової системи числення: а) 10_{10} ; б) 1_2 ; в) 2_{10} ; г) 0_{10} ?
4. Яке з двох чисел більше: 1111_2 чи 11_{10} ? Дайте пояснення.
5. Яку операцію необхідно виконати для перетворення цілої частини числа з однієї системи числення у іншу: а) ділення; б) віднімання; в) множення; г) додавання?
6. Які символи використовують у вісімковій системі числення: а) 0 і 1; б) $1 \div 8$; в) $0 \div 9$; г) $0 \div 7$?

Комп'ютерний практикум №2. Розробка блок-схеми алгоритму вирішення задачі

Мета роботи: ознайомлення зі правилами виконання блок-схем.

Завдання Проаналізувати варіант завдання. Скласти блок-схему алгоритму обчислення значень за даними варіантів завдань. Побудувати блок-схему у середовищі Microsoft Visio.

Варіанти завдань:

№	РОЗРАХУНКОВІ ФОРМУЛИ	Значення вихідних даних
1	$Y = \sqrt{x^3 + ax^2 + bx + c};$	$x=0,35; a=x+0.52b; b=cx^2+1; c=0,8$
2	$Y = 5 \sin^2 \ln(cx^3 + 1) ;$	$x=\cos^2(a+b); a=0.52; b=\sin^2 a$
3	$Y = \ln(e^x + bx^2);$	$x=\cos^2(a+b); a=0.52; b=\sin^2 a$
4	$Z = \frac{\sqrt{x^4 + ax + b}}{\sqrt{x^4 + ax + b}};$	$x=0,75; a=-x^2+\lg 0,08+e^{-x}; b=x+4a$
5	$C = \frac{D + \sqrt{x}}{\ln(\sqrt{R + E})};$	$x=1,08; D=12,5+\ln E; E=0,7; R=8D-x^4+1$
6	$Y = (1+z) \frac{x + \frac{y}{a - \frac{1}{1+x}}}{1};$	$x=0,8 \cdot 10^{-2}; y = e^{\sqrt{x^3+1,75z}}; z=5 \cdot 10^{-1,8}$
7	$Y = \frac{(a+b)^n}{1 + \frac{x}{x^m + b^{m-n}}};$	$x=0,81; a=\lg 0,083; b=e^a; n=e^{a/b}; m=n-1$
8	$A = \frac{\alpha_1 \beta_1 - \alpha_2 \beta_2}{m \alpha_1 - \alpha_2^2};$	$\alpha_1 = \sin 0,18; \beta_1 = e^{\alpha_1}; \alpha_2 = \ln 0,05 + e^{\alpha_1}; \beta_2 = 1,7$
9	$S = K_1 a_1 + K_2 a_2 - \beta;$	$K_1=0,05; K_2=0,03; \alpha_1 = a_0 + \Delta a; \alpha_2 = T \sin 30^\circ + \omega; \omega = e^{K_1+K_2}; \Delta a = 0,1$
10	$Z = (a\sqrt{b} - c\sqrt{d})^2 \frac{5,6}{a+b+c};$	$a = \sqrt{(b-c)^3}; b=a^2-1; c=a^2-b; d=(b^2-a^2)$
11	$Y = \sin \frac{1}{x+0.2} + \lg 0,08e^x;$	$x = -2,5a^2 + b\sqrt{c}; a=0,8c+bx; c=0,5$
12	$Y = \frac{\cos^2 ax + b}{\sin^2 x};$	$x=0,82; a = 0,5c + e^x; b=x^2-ac; c=0,8\ln 0,072$
13	$Y = \frac{x + 3a - K_1 x}{K_2 x + K_3 x};$	$X=5a+K_1 K_2; K_1=0,8; K_2=K_3=0,5; a=0,56$
14	$Y = \frac{x^4 - x^2 - b}{x-b} \frac{x^3 - x - a}{x-a}$	$b=2^x-1; x=\lg 0,005+1$

№	РОЗРАХУНКОВІ ФОРМУЛИ	Значення вихідних даних
15	$Y = 5 \sin^2 \ln(cx^3 + 1) ;$	$x = \cos^2(a+b); a=0.52; b=\sin^2 a$
16	$Z = \frac{\sqrt{x^4 + ax + b}}{\sqrt{x^4 + ax + b}};$	$x=0,75; a=-x^2+\lg 0,08+e^{-x}; b=x+4a$
17	$Y = (1+z) \frac{x + \frac{y}{x}}{a - \frac{1}{1+x}};$	$x=0,8 \cdot 10^{-2}; y = e^{\sqrt{x^3+1,75z}}; z=5 \cdot 10^{-1,8}$
18	$A = \frac{\alpha_1 \beta_1 - \alpha_2 \beta_2}{m \alpha_1 - \alpha_2^2};$	$\alpha_1 = \sin 0,18; \beta_1 = e^{\alpha_1}; \alpha_2 = \ln 0,05 + e^{\alpha_1};$ $\beta_2 = 1,7$
19	$Z = (a\sqrt{b} - c\sqrt{d})^2 \frac{5,6}{a+b+c};$	$a = \sqrt{(b-c)^3}; b=a^2-1; c=a^2-b; d=(b^2-a^2)$
20	$A = \frac{\alpha_1 \beta_1 - \alpha_2 \beta_2}{m \alpha_1 - \alpha_2^2};$	$\alpha_1 = \sin 0,18; \beta_1 = e^{\alpha_1}; \alpha_2 = \ln 0,05 + e^{\alpha_1};$ $\beta_2 = 1,7$
21	$Y = \sqrt{x^3 + ax^2 + bx + c};$	$x=0,35; a=x+0.52b; b=cx^2+1; c=0,8$
22	$Y = \ln(e^x + bx^2);$	$x = \cos^2(a+b); a=0.52; b=\sin^2 a$
23	$C = \frac{D + \sqrt{x}}{\ln(\sqrt{R+E})};$	$x=1,08; D=12,5+\ln E; E=0,7; R=8D-x^4+1$
24	$Y = \frac{(a+b)^n}{1 + \frac{x}{x^m + b^{m-n}}};$	$x=0,81; a=\lg 0,083; b=e^a; n=e^{a/b}; m=n-1$
25	$S = K_1 a_1 + K_2 a_2 - \beta;$	$K_1=0,05; K_2=0,03; \alpha_1 = a_0 + \Delta a;$ $\alpha_2 = T \sin 30^\circ + \omega; \omega = e^{K_1+K_2}; \Delta a = 0,1$

Теоретичні відомості

АЛГОРИТМ. ВЛАСТИВОСТІ АЛГОРИТМУ

Особливе місце при підготовці задачі до рішення на ЕОМ займає розробка чи вибір алгоритму. Поняття алгоритму широко використовується як у математиці, так і в програмуванні.

Алгоритм – сукупність правил, однозначно визначаючих процес перетворення вхідних і проміжних даних у результат рішення задачі. Опис алгоритму являє собою загальну схему рішення задачі.

Для алгоритму характерні наступні *властивості*:

- **дискретність роботи алгоритму** – алгоритм виконується по кроках та при цьому на кожному кроці виконується тільки одна операція.

- **детермінованість**, чи визначеність, тобто однозначність – його розуміння для будь-якого виконавця, що приводить до точного виконання однієї і тієї ж послідовності дій;

- **елементарність кроків алгоритму** – закон отримання наступної системи величин з попередньої повинен бути простим та локальним.

- **виконуваність операцій** – в алгоритмі не має бути не виконуваних операцій.

- **скінченність алгоритму**. Опис алгоритму повинен бути скінченним.

- **результативність**, чи спрямованість, тобто властивість досягнення за кінцеве число досить простих кроків шуканого результату розглянутої задачі. Якщо спосіб отримання наступної величини з деякої заданої величини не дає результату, то має бути вказано, що треба вважати результатом алгоритму;

- **масовість**, тобто придатність для рішення будь-якої задачі з деякого класу задач.

Розрізняють наступні способи представлення алгоритмів: текстуальний, операторний і графічний.

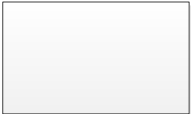
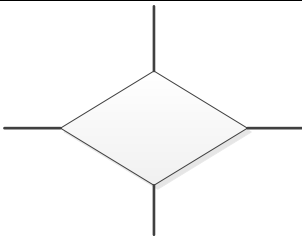
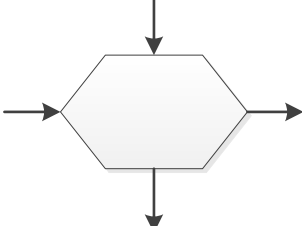
Найбільше поширення в даний час одержав графічний спосіб, при якому обчислювальний процес розчленовується на окремі операції, що відображаються у виді умовних графічних символів (блоків).

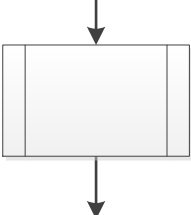
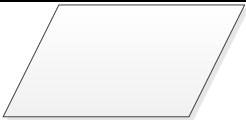
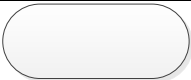
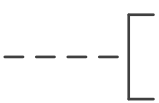
З 01.01.92 введений ГОСТ 19.701-90 (ISO 5807-85), що визначає правила виконання схем і перелік блоків, їхнє найменування, форму і розміри. Блоки з'єднуються між собою у визначеній послідовності стрілками. У середині блоків у вигляді формул чи тексту вказується інформація, що пояснює, характеризує виконувані ними дії.

У табл. 1 приведені деякі найбільш часто використовувані блоки і пояснення до них. Теорія структурного програмування доводить, що алгоритм будь-якої складності можна побудувати за допомогою основного базового набору структур:

- 1) послідовна (лінійна) структура;
- 2) структура, що розгалужується;
- 3) циклічна структура.

Таблиця 1

Блоки	Опис
	Обчислювальна дія чи послідовність обчислювальних дій
	Перевірка умов
	Початок циклу

Блоки	Опис
	Обчислення за підпрограмою чи стандартною підпрограмою
	Ввід чи вивід даних
	Розрив лінії потоку, з'єднувач
	Початок, кінець, зупинка, вхід і вихід у підпрограмах
	Пояснення, зміст підпрограм, формули

Найпростішими для розуміння і використання є лінійні структури (рис. 1).

Лінійним називається алгоритм (фрагмент алгоритму), в якому окремі команди виконуються послідовно одна за одною, незалежно від значень вхідних даних і проміжних результатів.

Приклад алгоритму лінійної структури (рис. 1).

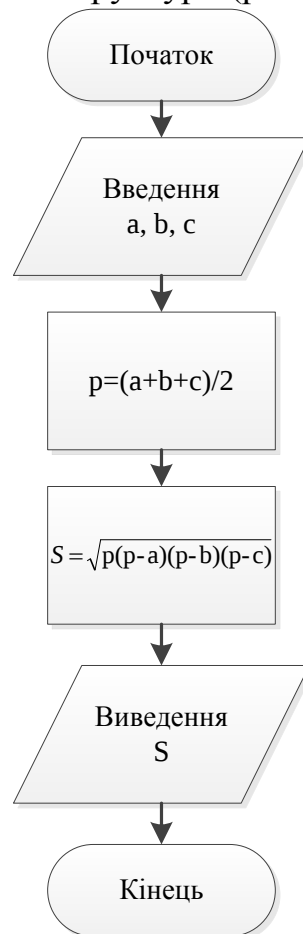


Рис. 1. Блок-схема алгоритму обчислення площі трикутника

СТВОРЕННЯ СХЕМ АЛГОРИТМІВ ЗАСОБАМИ MICROSOFT VISIO

MS Visio – засіб створення різних діаграм, у тому числі інформаційних моделей процесів, блок-схем, структурних схем, графіків робіт, призначений для фахівців; керівників проектів, а також дозволяє описувати (моделювати) різні інформаційні технології і системи в керуванні й проектуванні.

Початок роботи. Створення будь-якої діаграми починається з команди **Файл – створити – Вибір типу діаграми**.

Створення блок схеми починається з того, що на діалоговій панелі обираємо елемент меню – *Блок-схема*, потім – *Проста блок-схема*.

Після вибору типу діаграми відкривається порожнє вікно (*blank drawing*) – Visio готове для створення діаграми.

Ліворуч (на зеленому полі) розташовується перелік символів, специфічний для обраного типу діаграми.

Символи можуть розташовуватися на бланку діаграми простою операцією перетаскування (ліворуч/праворуч). Для кожного символу надається короткий опис, для цього необхідно установити курсор миші на відповідному символі.

Процес побудови діаграм в Visio:

Спочатку на робочий аркуш перетягуються необхідні символи-примітиви (*SmartShapes*), потім вибудовуються зв'язки між ними і розташовуються підписи (*текст*). Технологія (*Smart Connector*) дозволяє переміщувати блоки по робочому аркушу, при цьому зв'язки автоматично перебудовуються, а у випадку перетинання сполучних ліній автоматично створюється "перехрестя" у вигляді арки, що складається з дуги або декількох (від двох до семи) відрізків.

Перш ніж почати створення діаграми, необхідно обрати режим, в якому всі її елементи будуть з'єднуватися автоматично. Для цього використовуємо кнопку – Автосоединение  (вкл/выкл), розташовану на панелі інструментів

Основні елементи МЕНЮ панель Standard:

1. Кнопка – Жирна Стрілка – Показчик [Pointer tool]  – для вибору будь-якого елемента на аркуші. Використовується для зміни розмірів і для переміщення елемента діаграми на аркуші.
2. Кнопка – [A- text] – для створення тексту – (перший зі шрифтів Arial).
3. Лінія для з'єднання елементів діаграми проводиться мишею. Вид лінії вибираються нижче кн. Line Ends – Кінці ліній.

Контрольні запитання:

1. Що таке алгоритм? У чому полягає суть побудови алгоритмів?
2. Що таке алгоритмічний процес?
3. Які існують способи опису алгоритмів?
4. Що таке блок-схема?
5. Основні графічні елементи блок-схем, їх призначення.
6. Правила оформлення блок-схем.

Комп'ютерний практикум №3. Робота в інтегрованому середовищі розробки. Типи даних. Поняття змінної

Мета роботи: Знайомство з середовищем IDLE, збереження коду. Розробка найпростіших програм. Дані та їх типи. Цілі числа, числа з плаваючою точкою, рядки. Зміна типу даних. Поняття змінної. Операції.

Завдання

1. Змінній `var_int` надайте значення 10, `var_float` – значення 8.4, `var_str` – "No".
2. Змініть значення, збережене в змінній `var_int`, збільшивши його в 3.5 рази, результат зв'яжіть зі змінною `big_int`.
3. Змініть значення, збережене в змінній `var_float`, зменшивши його на одиницю, результат зв'яжіть з тією ж змінною.
4. Розділіть `var_int` на `var_float`, а потім `big_int` на `var_float`. Результат даних виразів не прив'яжуйте до жодних змінних.
5. Змініть значення змінної `var_str` на "NoNoYesYesYes". При формуванні нового значення використовуйте операції конкатенації (+) і повторення рядка (*).
6. Виведіть значення всіх змінних.

Варіанти завдань:

№	Вхідні дані	Вихідні дані	Формула
1	4 невід'ємних цілих числа a, b, c та x	дійсне число - результат обчислення формули	$Y = \sqrt{x^3 + ax^2 + bx + c};$
2	2 невід'ємних цілих числа c та x , 2 від'ємних цілих числа a та b	дійсне число - результат обчислення формули	$Y = 5\sqrt{cx^3 + 1} + a^3 - b^2;$
3	3 невід'ємних цілих числа a, z та x	дійсне число - результат обчислення формули	$Y = (1 + z) \frac{\sqrt{x}}{a - \sqrt{\frac{1}{1+x}}};$
4	3 невід'ємних цілих числа a, b та x	дійсне число - результат обчислення формули	$Z = \frac{\sqrt{x^4 + ax + b}}{\sqrt{x^4 + ax + b}};$
5	3 невід'ємних цілих числа a, b та x	дійсне число - результат обчислення формули	$Z = \frac{\sqrt{x^2 + b}}{\sqrt{x^2 + ax}};$
6	4 невід'ємних дійсних числа	дійсне число - результат обчислення формули	$Y = (1 + z) \frac{x + \frac{y}{x}}{a - \frac{1}{1+x}};$
7	5 невід'ємних цілих числа	дійсне число - результат обчислення формули	$Y = \frac{(a + b)^n}{1 + \frac{x}{x^m + b^{m-n}}};$

№	Вхідні дані	Вихідні дані	Формула
8	5 невід'ємних цілих числа	дійсне число - результат обчислення формули	$A = \frac{\alpha_1 \beta_1 - \alpha_2 \beta_2}{m \alpha_1 - \alpha_2^2};$
9	5 невід'ємних дійсних числа	ціле число - результат обчислення формули	$S = K_1 a_1 + K_2 a_2 - \beta;$
10	4 невід'ємних цілих числа	дійсне число - результат обчислення формули	$Z = (a\sqrt{b} - c\sqrt{d})^2 \frac{5,6}{a + b + c};$
11	3 від'ємних цілих числа	дійсне число - результат обчислення формули	$Y = (1 + z) \frac{x}{a - \frac{1}{1 + x}};$
12	3 невід'ємних цілих числа a, b та x	дійсне число - результат обчислення формули	$Z = \frac{\sqrt{x^2 + ax + b}}{\sqrt{x^4 + ax + b}};$
13	5 невід'ємних цілих числа	дійсне число - результат обчислення формули	$Y = \frac{x + 3a - K_1 x}{K_2 x + K_3 x};$
14	3 від'ємних цілих числа	дійсне число - результат обчислення формули	$Y = \frac{x^4 - x^2 - b}{x - b} \frac{x^3 - x - a}{x - a}$
15	4 невід'ємних цілих числа	дійсне число - результат обчислення формули	$Y = (1 + z^2) \frac{x + \frac{y}{x}}{a + \frac{1}{1 + x}};$
16	3 від'ємних цілих числа	дійсне число - результат обчислення формули	$Z = \frac{\sqrt{x^4 + ax + b}}{\sqrt{x^4 + ax + b}};$
17	4 невід'ємних цілих числа	дійсне число - результат обчислення формули	$Y = (1 + z) \frac{x + \frac{y}{x}}{a - \frac{1}{1 + x}};$
18	5 невід'ємних цілих числа	дійсне число - результат обчислення формули	$A = \frac{\alpha_1 \beta_1 - \alpha_2 \beta_2}{m \alpha_1 - \alpha_2^2};$
19	4 невід'ємних цілих числа	дійсне число - результат обчислення формули	$Z = (a\sqrt{b} - c\sqrt{d})^2 \frac{5,6}{a + b + c};$
20	5 невід'ємних дійсних числа	дійсне число - результат обчислення формули	$A = \frac{\alpha_1 \beta_1 - \alpha_2 \beta_2}{m \alpha_1 - \alpha_2^2};$
21	4 невід'ємних дійсних числа	дійсне число - результат обчислення формули	$Y = \sqrt{x^3 + ax^2 + bx + c};$
22	4 невід'ємних цілих числа	дійсне число - результат обчислення формули	$Y = \frac{(a + b)^n}{1 + \frac{x}{x + b}};$

№	Вхідні дані	Вихідні дані	Формула
23	5 невід'ємних цілих числа	дійсне число - результат обчислення формули	$A = \frac{\alpha_2 \beta_2 - \alpha_1 \beta_1}{\alpha_1 - m \alpha_2^2};$
24	5 невід'ємних цілих числа	дійсне число - результат обчислення формули	$Y = \frac{(a+b)^n}{1 + \frac{x}{x^m + b^{m-n}}};$
25	3 від'ємних цілих числа	ціле число - результат обчислення формули	$S = Ka + K^2 - \beta^3;$

Теоретичні відомості

УСТАНОВКА PYTHON НА ОПЕРАЦІЙНІЙ СИСТЕМІ WINDOWS

Для запуску програм на мові Python необхідна програма-інтерпретатор (віртуальна машина) Python. Дана програма приховує від Python-програміста все особливості операційної системи, тому, написавши програму на Python в системі Windows, її можна запустити, наприклад, в GNU/Linux і отримати такий же результат.

Завантажити та встановити інтерпретатор Python можна абсолютно безкоштовно з офіційного сайту: <https://python.org/downloads/windows/>. Для роботи необхідний інтерпретатор Python версії 3 або вище.

Завантажити Python. Відкрити файл *Python Software Foundation*. Для установки обирати будь-яку папку (можна залишити за замовчуванням).

В установник Python для Windows вбудоване середовище розробки IDLE.

Після установки програми запустити інтерактивне графічне середовище IDLE і дочекайтеся появи запрошення для введення команд:

Type "copyright", "credits" or "license()" for more information.

>>>

ІНТЕРАКТИВНИЙ РЕЖИМ

Один з варіантів виконання команд у Python – у вікні командного інтерпретатора.

В основному інтерпретатор виконує команди порядково: пишеться рядок, натискається Enter, інтерпретатор виконує їх, відображуючи результат.

Запустіть командний інтерпретатор.

Оскільки ніяких команд ми поки не знаємо, то будемо використовувати Python як калькулятор (можливості мови це дозволяють).

Знайти значення наступних математичних виразів. Після завершення набору виразу натисніть клавішу Enter для завершення введення і виведення результату на екран.

>>> 3.0 + 6

9.0

>>> 4 + 9

13

>>> 1 – 5

-4

>>> _ + 6

2

```
>>> 3 * (5 - 8)
-9
>>> 2.4 + 3.0 / 2
3.9
>>>
```

Нижнім підкресленням позначається останній отриманий результат. Якщо зробити помилку при введенні команди, то Python повідомить про це:

```
>>> a
Traceback (most recent call last):
  File "<pyshell#0> ", line 1, in <module>
    a
NameError: name 'a' is not defined
>>>
```

Інший варіант роботи в інтерактивному режимі - це робота в середовищі розробки IDLE, у якій є інтерактивний режим роботи. На відміну від консольного варіанту тут можна спостерігати підсвічування синтаксису (в залежності від значення синтаксичної одиниці вона виділяється певним кольором). Прокручувати список раніше введених команд можна за допомогою комбінацій Alt + N, Alt + P.

Запустіть IDLE. Спробуйте вирішити математичні вирази тут.

СТВОРЕННЯ СКРИПТІВ

Незважаючи на зручності інтерактивного режиму роботи при написанні програм на Python, звичайно потрібно зберігати вихідний програмний код для подальшого використання. В такому випадку створюються файли, які передаються потім інтерпретатору на виконання. Стосовно інтерпретованих мов програмування часто вихідний код називають скриптом. Файли з кодом на Python зазвичай мають розширення *py*.

Підготувати скрипт можна в тому ж середовищі IDLE. Для цього, після запуску програми в меню слід вибрати команду **File → New Window (Ctrl + N)**, відкриється нове вікно. Потім бажано відразу зберегти файл (з розширенням *.py*). Після того як код буде підготовлений, знову збережіть файл (щоб оновити збереження). Тепер, можна запустити скрипт, виконавши команду меню **Run → Run Module (F5)**. Після цього в першому вікні з'явиться результат виконання коду. (Примітка: якщо набирати код, ще не збереженому файлі на початку, то підсвічування синтаксису буде відсутнім).

Підготуйте скрипт (з прикладами). Запустіть його на виконання.

Насправді скрипти можна готувати в будь-якому текстовому редакторі (бажано, щоб він підтримував підсвічування синтаксису мови Python). Крім того, існують спеціальні програми для розробки.

Запускати підготовлені файли можна не тільки в IDLE, але і в консолі за допомогою команди **python адреса / ім'я файлу**.

В консолі передайте інтерпретатору Python на виконання підготовлений файл.

Крім того, існує можливість налаштувати виконання скриптів за допомогою подвійного кліка по файлу.

ДАНІ ТА ЇХ ТИПИ

Можна помітити, що все, що ми робимо, ми робимо над чимось - якимись предметами або об'єктами. Ми міняємо властивості об'єктів та їх можливості.

Програми для комп'ютерів також маніпулюють якимись об'єктами (назвемо їх поки даними).

Очевидно, дані бувають різними. Дуже часто комп'ютерній програмі доводиться працювати з числами і рядками. Наприклад, використовували числа, виконуючи над ними арифметичні операції. Можна сказати, що операція додавання виконувала зміну першого числа на величину другого, або множення збільшувало одне число в кількість разів, відповідну другому.

Числа в свою чергу також бувають різними: цілими, дробовими, можуть мати величезне значення або дуже довгу дробову частину. Найпростішими типами даних є:

- **цілі числа (int)** - додатні і від'ємні цілі числа, а також 0 (наприклад, 4, 687, -45, 0).

- **число з плаваючою точкою (float)** - дробові числа (наприклад, 1.45, -3.789654, 0.00453). Примітка: роздільником цілої і дробової частини служить точка, а не кома.

- **рядки (str)** - набір символів, укладених в лапки (наприклад, "ball", "What is your name?", 'dkfjUUv', '6589').

Примітка: лапки в Python можуть бути одинарними або подвійними.

ОПЕРАЦІЇ

Операція - це виконання якихось дій над даними (операндами). Для виконання конкретних дій потрібні спеціальні інструменти – **оператори**.

10	+	20
↑	↑	↑
Операнд1	Оператор	Операнд2

Математичні оператори, доступні над числами в Python:

Оператор	Опис
+	Додавання
-	Віднімання
/	Ділення (в результаті дійсне число)
//	Розподіл з округленням вниз
**	Зведення в ступінь
%	Залишок від ділення

```
>>> 5/3
1.6666666666666667
>>> 5//3
1
>>> 5%3
2
>>> 5**67
67762635780344027125465800054371356964111328125
>>>
```

Якщо один з операндів є дійсним числом, то в результаті вийде дійсне число.

Знайдіть значення виразу $2 + 56 * 5.0 - 45.5 + 5$.

При обчисленні математичних виразів Python дотримується пріоритету операцій:

```
>>> -2**4
-16
>>> -(2**4)
-16
>>> (-2)**4
16
```

>>>

Не зайвим буде позначити пріоритет у вигляді круглих дужок.

ЗМІНА ТИПУ ДАНИХ

Що буде, якщо спробувати виконати в одному виразі операцію над різними типами даними? Наприклад, скласти ціле і дробове число, число і рядок. Однозначну відповідь дати не можна: так, при складанні цілого числа і числа з плаваючою точкою, виходить число з плаваючою точкою, а якщо спробувати скласти будь-яке число і рядок, то інтерпретатор Python видасть помилку.

Вираз	Результат виконання
1 + 0.65	1.6499999999999999
"Hi, " + 15	Помилка

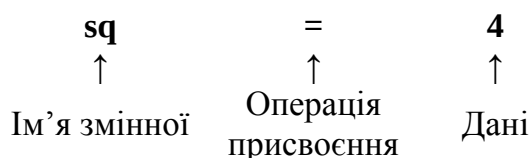
Однак, бувають випадки, коли програма отримує дані у вигляді рядків, а оперувати повинна числами (або навпаки). У такому випадку використовуються спеціальні функції (особливі оператори), що дозволяють перетворити один тип даних в інший. Так функція **int()** перетворює переданий їй рядок (або число з плаваючою точкою) в ціле, функція **str()** перетворює переданий їй аргумент в рядок, **float()** - в дробове число.

Вираз	Результат виконання
int("56")	56
int(4.03)	4
int("comp 486")	Помилка
str(56)	'56'
str(4.03)	'4.03'
float(56)	56.0
float("56")	56.0

ЗМІННІ В PYTHON

Дані зберігаються в комітках пам'яті комп'ютера. Коли вводиться число, воно розміщується в пам'яті. Але як дізнатися, куди саме? Як надалі звертатися до цих даних? Раніше, під час написання програм на машинній мові, звернення до осередків пам'яті здійснювали за допомогою вказівки регістрів. Але вже з появою асемблерів, при зверненні до даних стали використовувати так звані змінні. Механізм зв'язку між змінними і даними може відрізнятися в залежності від мови програмування і типу даних. Поки досить запам'ятати, що дані зв'язуються з будь-яким ім'ям і надалі звернення до них можливо за цим іменем.

У програмі на мові Python зв'язок між даними і змінними встановлюється за допомогою знака **=**. Така операція називається присвоєнням. Наприклад, вираз **sq = 4** означає, що на об'єкт (дані) в певній області пам'яті посилається ім'я **sq** і звертатися до них тепер слід за цим іменем.



Розглянемо вираз **y=x+3*6**, де **y** і **x** є змінними, які можуть містити значення числового типу. Мовою Python обчислити значення **y** при **x** дорівнює 1 можна наступним чином:

```
>>> x = 1
>>> y = x + 3 * 6
>>> y
```

19

>>>

У виразі можна використовувати змінну, якщо раніше їй не було присвоєно значення - для Python такі змінні не визначені.

Вміст змінної у можна побачити, якщо в інтерактивному режимі набрати її ім'я.

Імена змінних можуть бути будь-якими. Однак є кілька загальних правил їх написання:

1. Бажано давати змінним осмислені імена, що відображають про призначення даних, на які вони посилаються.

2. Ім'я змінної не повинно збігатися з командами мови (зарезервованими ключовими словами), які для Python мають певний сенс (ці слова підсвічуються в IDLE помаранчевим кольором):

and	as	assert	break	class	continue
def	del	elif	else	except	exec
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with
yield	True	False	None		

3. Ім'я змінної має починатися з букви або символу підкреслення (_).

Щоб дізнатися значення, на яке посилається змінна, перебуваючи в режимі інтерпретатора, досить її викликати (написати ім'я і натиснути Enter).

Приклад роботи зі змінними в інтерактивному режимі:

```
>>> apples = 100
>>> eat_day = 5
>>> day = 7
>>> apples = apples - eat_day * day
>>> apples
65
>>>
```

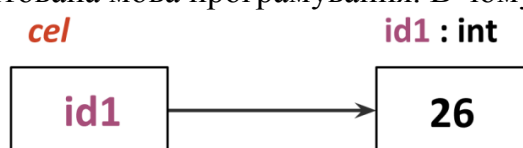
Приклад. Формула перекладу з градусів за Цельсієм (T_C) в градуси за Фаренгейтом (T_F) має вигляд:

$$T_F = 9/5 * T_C + 32$$

Знайдемо значення T_F при T_C рівному 26. Створимо змінну з ім'ям *cel*, що містить значення цілочисельного типу 26.

```
>>> cel = 26
>>> cel
26
>>> 9/5 * cel + 32
78.80000000000001
>>>
```

Є істотна особливість, яка відрізняє Python від інших мов програмування. Python - повністю об'єктно-орієнтована мова програмування. В чому це виражається?



У момент виконання присвоювання `cel = 26` в пам'яті комп'ютера створюється об'єкт, розташований за деякою адресою (умовно позначимо його як `id1`), що має значення 26 цілочисленного типу `int`. Потім створюється змінна з ім'ям `cel`, якій присвоюється адреса об'єкта `id1`. Змінні в Python містять адреси об'єктів або можна сказати, що змінні посилаються на об'єкти. Постійно зберігаючи в голові цю модель, для спрощення ми будемо говорити, що змінна містить значення.

Обчислення наступного виразу в результаті призведе до привласнення змінної `cel` значення 72, тобто спочатку обчислюється права частина, потім результат присвоюється лівій частині.

У момент виконання присвоювання `cel = 26` в пам'яті комп'ютера створюється об'єкт, розташований за деякою адресою (умовно позначимо його як `id1`), що має значення 26 цілочисельного типу `int`. Потім створюється змінна з ім'ям `cel`, якій присвоюється адреса об'єкта `id1`. Змінні в Python містять адреси об'єктів, тобто, змінні посилаються на об'єкти.

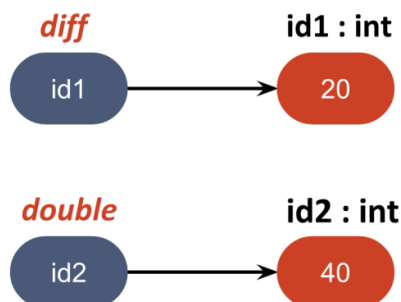
Обчислення наступного виразу в результаті призведе до привласнення змінної `cel` значення 72, тобто спочатку обчислюється права частина, потім результат присвоюється лівій частині.

```
>>> cel = 26 + 46
>>> cel
72
>>>
```

Приклад. Замість змінної `diff` підставити цілочисельне значення 20:

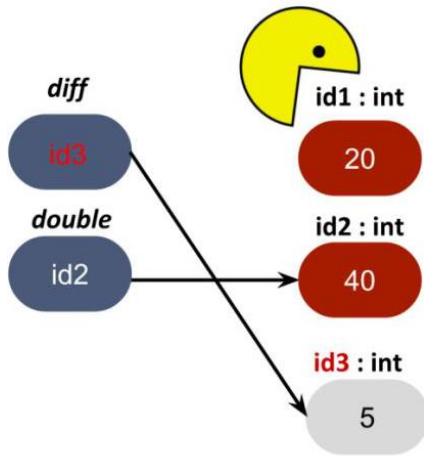
```
>>> diff = 20
>>> double = 2 * diff
>>> double
40
>>>
```

Після закінчення обчислень пам'ять для Python матиме такий вигляд:



Продовжимо обчислення. Дамо змінній `diff` значення 5 і подивимося вміст змінних `double` і `diff`.

```
>>> diff = 5
>>> double
40
>>> diff
5
>>>
```



У момент присвоювання змінній `diff` значення 5 в пам'яті створиться об'єкт за адресою `id3`, що містить цілочисельне значення 5. Після цього зміниться вміст змінної `diff`, замість адреси `id1` туди запишеться адреса `id3`. Також Python побачить, що на об'єкт за адресою `id1` більше ніхто не посилається і тому видалить його з пам'яті.

Python не змінює існуючі числові об'єкти, а створює нові. Це особливість числового типу даних - об'єкти цього типу є незмінними.

Контрольні запитання:

1. Які типи даних ви знаєте? Опишіть їх.
2. Чи можна перетворити дробове число в ціле? Ціле в дробове? У яких випадках можна рядок перетворити в число?
3. Наведіть приклади операцій. Для чого призначена операція присвоєння?
4. Які існують правила і рекомендації для іменування змінних?

Комп'ютерний практикум №4. Логічні вирази і логічний тип даних. Умовний оператор. Інструкція if

Мета роботи: Особливості організації умовних операторів і множинних розгалужень. Розробка програм.

Завдання

1.1. Присвоїти двом змінним будь-які числові значення.

1.2. Складіть чотири складних логічних вирази за допомогою оператора and, два з яких повинні давати істину, а два інших - хибність.

1.3. Аналогічно виконайте п. 2, використовуючи оператор or.

1.4. Спробуйте використати в складних логічних виразах роботу зі змінними рядкового типу.

2.1. Напишіть програмний код, в якому в разі, якщо значення якоїсь змінної більше 0, виводилося б спеціальне повідомлення (використовуйте функцію print). Один раз виконайте програму при значенні змінної більше 0, вдруге - менше 0.

2.2. Вдосконалить попередній код за допомогою гілки else так, щоб в залежності від значення змінної, виводилася або 1, або -1.

3.1. Напишіть програму за наступним описом:

a. двом змінним присвоюються числові значення;

b. якщо значення першої змінної більше другої, то знайти різницю значень змінних (відняти від першої другу), результат зв'язати з третьою змінною;

c. якщо перша змінна має менше значення, ніж друга, то третю змінну зв'язати з результатом суми значень двох перших змінних;

d. у всіх інших випадках, привласнити третій змінній значення першої змінної;

e. вивести значення третьої змінної на екран.

4. Напишіть програму у відповідності до варіанту та побудуйте блок-схему:

Варіанти завдань:

№	Завдання
1	Напишіть програму, яка в залежності від характеру вітру видає повідомлення про його швидкість: від 1 до 4 м / с - слабкий; від 5-10 м/с - помірний; від 9-18 м/с - сильний; більше 19 м / с - ураганний.
2	Напишіть програму, яка обчислює значення функції (на вхід подається дійсне число): $Y = \begin{cases} x^2 & -2,4 \leq x \leq 5,7 \\ 4 & \text{в іншому випадку} \end{cases}$
3	Напишіть програму, яка обчислює значення функції (на вхід подається дійсне число): $Y = \begin{cases} 2x^2 + a & x \leq 0; a=3,5bx; b=-0,9 \\ (x + 3a)c & x > 0; c=1,35x+b\sqrt{x} \end{cases}$
4	За кодом пакету мобільного зв'язку вивести назву оператора. Наприклад, (0)50, (0)66 – Vodafone.
5	За першою буквою з назви міста вивести країну. Перелік міст: Київ, Тула, Харків, Париж, Рівно, Одеса.
6	За першою буквою з назви області України вивести в якій стороні світу відносно Києва вона знаходиться. Перелік областей: Львівська, Житомирська, Крим, Луганська, Донецька, Рівненська, Харківська, Одеська.

№	Завдання
7	За номером кольору в веселці вивести повідомлення, чи відноситься він до теплих чи холодних кольорів(теплі – червоний, жовтий і т.п., холодні – синій, голубий і т.п.).
8	Обчислити площі фігур трикутника, прямокутника, паралелограма, трапеції. Вибір фігури здійснює користувач.
9	По введеному року, визначити до якого знаку зодіаку він відноситься (2016-2020).
10	Визначити до якої пори року відноситься і як називається місяць, номер якого ввів користувач.
11	По введеному року визначити чи є він високосним чи ні (2016-2020).
12	Скласти програму введення значення температури повітря t і видачі тексту "Хорошая погода!". Якщо $t > 10$ градусів і тексту „Плохая погода!", Якщо $t \leq 10$ градусів.
13	Скласти програму введення оцінки P , отриманої студентами, і видачі тексту „молодець!", Якщо $P = 5$, „Добре!", Якщо $P = 4$ і „Лентяй!", Якщо $P \leq 3$.
14	Дано натуральне число. Визначити, чи буде це число: парним, кратним 4.
15	Є коробка зі сторонами: $A \times B \times C$. Визначити, чи пройде вона в двері з розмірами $M \times K$.
16	Дано номер місця в плацкартному вагоні. Визначити, яке це місце: верхнє або нижнє, в купе або бічне.
17	Дано дійсне число A . Обчислити $f(A)$, якщо $f(x) = x^2 + 4x + 5$, при $x \leq 2$; в іншому випадку $f(x) = 1 / (x^2 + 4x + 5)$.
18	Дано дійсне число. Визначити, яке це число: додатнє, від'ємне, нуль.
19	Для заданого року визначити значення століття (наприклад, 1900 рік - 19 століття, 1901 рік - 20 століття).
20	Написати програму, яка перевіряє, чи є ціле число n , введене з клавіатури, кратним 5.
21	Скласти програму, яка змінній d присвоює найбільше з трьох чисел, а змінній s найменше з трьох чисел.
22	Дано від'ємні числа a , b і c . Знайти найбільше з трьох чисел і обчислити його куб.
23	Напишіть програму, яка обчислює значення функції (на вхід подається дійсне число): $F = \begin{cases} -ax^2 + b & \text{при } x < 0 \text{ та } b \neq 0 \\ \frac{x}{x-c} + 5.5 & x > 0 \text{ та } b = 0 \\ \frac{x}{-c} & \text{в інших випадках} \end{cases}$
24	Напишіть програму, яка обчислює значення функції (на вхід подається дійсне число): $F = \begin{cases} -ax^2 + b & \text{при } x = 0 \text{ та } b \neq 0 \\ \frac{x}{x-c} + 5.5 & x = 0 \text{ та } b = 0 \\ \frac{x}{-c} & \text{в інших випадках} \end{cases}$

№	Завдання
25	Напишіть програму, яка обчислює значення функції (на вхід подається дійсне число): $F = \begin{cases} -ax^2 + b & \text{при } x + 10 < 0 \text{ та } b \neq 0 \\ \frac{x}{x-c} + 5.5 & x + 10 > 0 \text{ та } b = 0 \\ \frac{x}{-c} & \text{в інших випадках} \end{cases}$

Теоретичні відомості

ЛОГІЧНІ ВИРАЗИ І ЛОГІЧНИЙ ТИП ДАНИХ

Числа можна порівнювати. В Python для цього є наступні операції порівняння:

> більше
 < менше
 >= більше чи рівне
 <= менше чи рівне
 == рівне
 != не рівне

В інтерактивному режимі порівняти два числа:

```
>>> 6>5
True
>>> 7<1
False
>>> 7==7
True
>>> 7 != 7
False
>>>
```

Python повертає True (Істина == 1), коли порівняння вірне і False (Хибність == 0) - в іншому випадку. True і False відносяться до логічного (булевого) типу даних bool.

```
>>> type(True)
<class 'bool'>
>>>
```

Не рідко використовуються більш складні вирази. Може знадобитися отримати відповідь "Так" або "Ні" в залежності від результату виконання двох простих виразів. Наприклад, "на вулиці йде сніг або дощ", "змінна new більше 12 і менше 20" і т.п.

У таких випадках використовуються спеціальні оператори, що об'єднують два і більше простих логічних вирази. Широко використовуються два способи об'єднання: через, так звані, логічні **I (and)** та **АБО (or)**.

Щоб отримати істину (True) при використанні оператора and, необхідно, щоб результати обох простих виразів, які пов'язує цей оператор, були

істинними. Якщо хоча б в одному випадку результатом буде False (брехня), то і весь складний вираз буде хибним.

Щоб отримати істину (True) при використанні оператора or, необхідно, щоб результати хоча б одного простого виразу, що входить до складу складного, був істинними. У разі оператора or складний вираз стає хибним лише тоді, коли хибні всі прості вирази, що входять.

Приклад:

x = 8

y = 13

x == 8 and y < 15 # x рівне 8 та y менше 15

x > 8 and y < 15 # x більше 8 та y менше 15

x != 0 or y > 15 # x не рівне 0 або y менше 15

x < 0 or y > 15 # x менше 0 або y менше 15

Визначте усно результати виконання операцій, наведених в прикладі вище. Перевірте правильність припущень, виконавши дані вирази за допомогою інтерпретатора мови Python.

Все це можна об'єднати і представити у вигляді таблиці, де 0 – False, а 1 – True.

XY	and	or
00	0	0
01	0	1
10	0	1
11	1	1

Для Python істинним або хибним може бути не тільки логічне висловлювання, а й об'єкт.

В Python будь-яке число, не рівне нулю, або непорожній об'єкт інтерпретується як істина.

Числа, рівні нулю, порожні об'єкти і спеціальний об'єкт None інтерпретуються як хибність.

Приклад:

```
>>> " and 2 # False and True
"
```

```
>>> " or 2 # False or True
2
>>>
```

У Python є три логічних оператора and, or, not.

```
>>> y = 6>8
```

```
>>> y
```

```
False
```

```
>>> not y
```

```
True
```

```
>>> not None
```

```
True
```

```
>>> not 2
```

```
False
```

```
>>>
```

Результатом застосування логічного оператора `not` (НЕ) відбудеться заперечення операнда, тобто якщо операнд істинний, то `not` поверне - хибність, якщо хибних, то - істину.

Логічний оператор `and` (І) поверне `True` (істину) або `False` (брехня), якщо його операндами є логічні висловлювання.

```
>>> 2>4 and 45>3 # комбінація False and (И) True поверне значення False
False
>>>
```

Якщо операндами оператора `and` є об'єкти, то в результаті Python поверне об'єкт:

```
>>> " and 2 # False and True
"
>>>
```

Для обчислення оператора `and` Python обчислює операнди зліва направо і повертає перший об'єкт, який має помилкове значення.

```
>>> 0 and 3 # поверне перший помилковий об'єкт-операнд
0
>>> 5 and 4 # поверне крайній правий об'єкт-операнд
4
>>>
```

Якщо Python не вдається знайти помилковий об'єкт-операнд, то він повертає крайній правий операнд.

Логічний оператор `or` діє схожим чином, але для об'єктів-операндів Python повертає перший об'єкт, який має істинне значення. Python припинить подальші обчислення, як тільки буде знайдений перший об'єкт, який має істинного значення.

```
>>> 2 or 3 # поверне перший справжній об'єкт-операнд
2
>>> None or 5 # поверне другий об'єкт-операнд, тому що перший завжди
хибний
5
>>> None or 0 # поверне об'єкт-операнд, що залишився
0
>>>
```

Логічні вирази можна комбінувати:

```
>>> 1+3 > 7 # пріоритет + вище, ніж >
False
>>> (1+3) > 7 # дужки сприяють наочності і позбавляють від помилок
False
>>> 1+(3>7)
1
>>>
```

В Python можна перевіряти приналежність інтервалу:

```
>>> x=0
```

```
>>> -5<x<10 # еквівалентно: x > -5 and x<10
True
>>>
```

Рядки в Python теж можна порівнювати за аналогією з числами. Символи, як і все інше, представлено в комп'ютері у вигляді чисел. Є спеціальна таблиця, яка ставить у відповідність кожному символу деяке число.

Визначити, яке число відповідає символу можна за допомогою функції `ord()`:

```
>>> ord('L')
76
>>> ord('Ф')
1060
>>> ord('A')
65
>>>
```

Тепер порівняння символів зводиться до порівняння чисел, які їм відповідають:

```
>>> 'A' > 'L'
False
>>>
```

Для порівняння рядків Python їх порівнює посимвольно:

```
>>> 'Aa' > 'Ll'
False
>>>
```

Оператор `in` перевіряє наявність підрядка в рядку:

```
>>> 'a' in 'abc'
True
>>> 'A' in 'abc' # великої літери А немає
False
>>> "" in 'abc' # порожній рядок є в будь-якому рядку
True
>>> " in "
True
>>>
```

УМОВНА КОНСТРУКЦІЯ IF

Найбільш часто логічні вирази використовуються всередині умовної інструкції `if`:

```
if<<умова>>:
    <<блок виразів>>
```

Блок виразів виконується тільки в тому випадку, якщо вираз, який знаходиться в умові, є істинним (`True`).

Умовний вираз може включати безліч перевірок. Загальний синтаксис у нього наступний:

```
if<<умова>>:
    <<блок виразів>>
```

```

elif<<умова>>:
    <<блок виразів>>
else:
    <<блок виразів>>

```

Блок виразів, що відноситься до else, виконується, коли всі вищі умови повернули значення False.

Приклад:

```

pH = float(input("Введіть рН: ")) # рядок перетворили до дійсного типу
if pH == 7.0:
    print (pH, "Вода")
elif 7.36 < pH < 7.44:
    print (pH, "Кров")
else:
    print ("Щось інше")

```

Контрольні запитання:

1. Як описується та виконується оператор розгалуження?
2. Як описується та виконується оператор множинного розгалуження?
3. Що називається логічним виразом?
4. Які 3 можливих варіанти представлення умови в інструкції *if*?

Комп'ютерний практикум №5. Універсальний організатор циклу – інструкція while

Мета роботи: Особливості організації циклічної інструкції while.

Завдання

1. Напишіть скрипт на мові програмування Python, що виводить ряд чисел Фібоначчі (див. приклад). Запустіть його на виконання. Потім змініть код так, щоб виводився ряд чисел Фібоначчі, починаючи з п'ятого члена ряду і закінчуючи двадцятим.

2. Напишіть цикл, що виводить ряд парних чисел від 0 до 20. Потім, кожне третє число в ряді від -1 до -21.

3. Напишіть скрипт у відповідності до варіанту та побудуйте блок-схему:

Варіанти завдань:

№	Обчислити значення функції на відрізку $[x_0; x_k]$ з кроком dx за допомогою циклу з while	
1	$y = 10^{-2}bc/x + \sqrt{a^3x}$	$x_0=-1.5; x_k=3.5; dx=0.5;$ $a=-1.25; b=-1.5; c=0.75$
2	$y = 1.2(a-b)^3 a^{x^2} + x$	$x_0=-0.75; x_k=-1.5; dx=-0.05;$ $a=1.5; b=1.2$
3	$y = 10^{-1}ax^3(a-bx)$	$x_0=-0.5; x_k=2.5; dx=0.05;$ $a=10.2; b=1.25$
4	$y = ax^3 + a^2(x^3 - b)$	$x_0=5.3; x_k=10.3; dx=0.25;$ $a=1.35; b=-6.25$
5	$y = x^4 + (2 + x^3 - d)$	$x_0=4.6; x_k=5.8; dx=0.2; d=1.3$
6	$y = x^2 + (5x + b/x)$	$x_0=-1.5; x_k=-2.5; dx=-0.5; b=-0.8$
7	$y = 9(x + 15\sqrt{x^3 + b^3})$	$x_0=-2.4; x_k=1; dx=0.2; b=2.5$
8	$y = 9x^4 + (57.2 + x)$	$x_0=-0.75; x_k=-2.05; dx=-0.2$
9	$y = 10.0025bx^3 + \sqrt{x + b^{0.82}}$	$x_0=-1; x_k=4; dx=0.5; b=2.3$
10	$y = x(\sqrt{x + b - 0.84})$	$x_0=-2.5; x_k=-3.5; dx=-0.1; b=3.9$
11	$y = x + \sqrt{x^3 + a - b^x}$	$x_0=-4; x_k=-6.2; dx=-0.2; a=0.1; b=1.25$
12	$y = 9(x^3 + b^3)x$	$x_0=1; x_k=2.2; dx=0.2; b=3.2$
13	$y = (x^{5/2} - b)\ln(x^2 + 12.7)$	$x_0=0.25; x_k=5.2; dx=0.3; b=0.8$
14	$y = 10^{-3}x^{5/2} + x + b$	$x_0=1.76; x_k=-2.5; dx=-0.25; b=35.4$
15	$y = 5.2x^{3/2} + x + b$	$x_0=1.2; x_k=-2.5; dx=-0.3; b=12.6$
16	$y = \frac{0.0084(x^{5/4} + b)}{(x^2 + 3.62)}$	$x_0=-2.25; x_k=-2; dx=0.05;$ $b=74.2$
17	$y = 0.8 * 10^{-5} (x^3 + b^3)^{7/6}$	$x_0=-0.5; x_k=0.5; dx=0.1; b=6.74$
18	$y = \frac{((x^3 + 0.25) + 1)^{3/2}}{0.8x * 10^{-3}}$	$x_0=0.12; x_k=0.64; dx=0.2$

№	Обчислити значення функції на відрізку $[x_0; x_k]$ з кроком dx за допомогою циклу while	
19	$y = x^{b^b} + (x^{3/2} + b^{3/4})$	$x_0=13.7; x_k=19.1; dx=0.4; b=2$
20	$y = 10^{-2}(a + bx) - a^{x^3+b}$	$x_0=-3.4; x_k=-1.4; dx=0.1; a=5; b=4$
21	$y = ax^3 + b^{5/4}xa^{-x}$	$x_0=2.51; x_k=10.59; dx=1.01; a=4; b=2$
22	$y = ax^{5/2} + (\sqrt{a^x})$	$x_0=-0.31; x_k=0.61; dx=0.3; a=8$
23	$y = 9(x + 15\sqrt{x^2 + b^2})$	$x_0=-2.4; x_k=1; dx=0.2; b=2.5$
24	$y = ax^2 + b^{4/3}xa^{-x}$	$x_0=2.51; x_k=10.59; dx=1.01; a=4; b=3$
25	$y = ax^{5/2} + (\sqrt{x^a})$	$x_0=-0.31; x_k=0.61; dx=0.3; a=4$

Теоретичні відомості ЦИКЛ WHILE

Алгоритм, в якому передбачено неодноразове виконання однієї і тієї ж послідовності дій, називається алгоритмом циклічної структури або циклом. Цикли - це інструкції, які виконують одну й ту ж послідовність дій, поки діє задана умова. Цикл дозволяє істотно скоротити розмір запису алгоритму, представити його компактно шляхом відповідної організації пропонованих дій. Повторювати які-небудь дії має сенс при різних значеннях параметрів, змінюваних при кожному новому виході на повторення. Такі параметри, що змінюються, називаються параметрами циклу. Блок повторюваних операторів називають тілом циклу.

Універсальним організатором циклу в мові програмування Python (як і в багатьох інших мовах) є конструкція **while**. Слово "while" з англійської мови перекладається як "поки" ("поки логічне вираження повертає істину, виконувати певні операції"). Конструкцію **while** мовою Python можна описати наступною схемою:

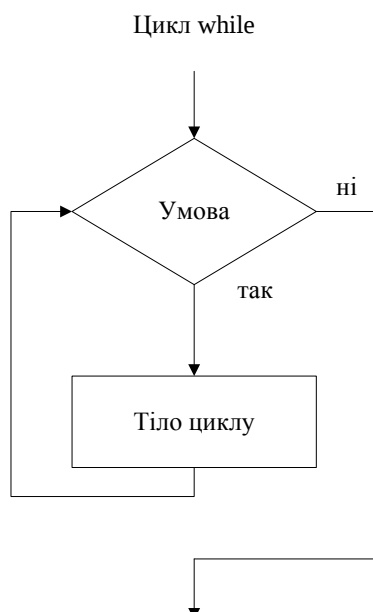
while *а логічний_оператор* *b*:

дія(і)

змінення а

Ця схема є приблизною, оскільки логічний вираз в заголовку циклу **while** може бути більш складним, а змінюватися може змінна (або вираз) *b*.

Може виникнути питання: "Навіщо змінювати *a* або *b*?". Коли виконання програмного коду доходить до циклу **while**, виконується логічний вираз в заголовку, і, якщо було отримано True (істина), виконуються вкладені вирази. Після потік виконання програми знову повертається в заголовок циклу **while**, і знову перевіряється умова. Якщо умова ніколи не буде хибною, то не буде причин зупинки циклу і програма зациклиться. Щоб цього не сталося, необхідно передбачити можливість виходу з циклу - хибність виразу в заголовку. Таким чином, змінюючи значення змінної в тілі циклу, можна довести логічний вираз до хибності.



Цю змінювану змінну, яка використовується в заголовку циклу `while`, зазвичай називають лічильником. Як і будь-якій змінній їй можна давати довільні імена, однак дуже часто використовують букви `i` та `j`. Найпростіший цикл на мові програмування Python може виглядати так:

```
str1 = "+"
i = 0
while i < 10:
    print (str1)
    i = i + 1
```

В останньому рядку коду відбувається збільшення значення змінної `i` на одиницю, тому з кожним оборотом циклу її значення збільшується. Коли буде досягнуто число 10, логічний вираз `i < 10` дасть хибний результат, виконання тіла циклу буде припинено, а потік виконання програми перейде на команди наступні за всією конструкцією циклу. Результатом виконання скрипта наведеного вище є виведення на екран десяти знаків `+` в стовпчик. Якщо збільшувати лічильник в тілі циклу не на одиницю, а на 2, то буде виведено лише п'ять знаків, оскільки цикл зробить лише п'ять оборотів.

Більш складний приклад з використанням циклу:

```
fib1 = 0
fib2 = 1
print (fib1)
print (fib2)
n = 10
i = 0
while i < n:
    fib_sum = fib1 + fib2
    print (fib_sum)
    fib1 = fib2
    fib2 = fib_sum
    i = i + 1
```

Цей приклад виводить числа Фібоначчі - ряд чисел, в якому кожне наступне число дорівнює сумі двох попередніх: 0, 1, 1, 2, 3, 5, 8, 13 і т.д. Скрипт виводить дванадцять членів ряду: два (0 і 1) виводяться поза циклом і десять виводяться в результаті виконання циклу.

Як це відбувається? Вводяться дві змінні (fib1 і fib2), яким присвоюються початкові значення. Присвоюються значення змінної n та лічильнику i, між якими ті чи інші математичні відношення формують бажане число витків циклу. В середині циклу створюється змінна fib_sum, якій присвоюється сума двох попередніх членів ряду, і її ж значення виводиться на екран. Далі змінюються значення fib1 і fib2 (першому присвоюється друге, а другому - сума), а також збільшується значення лічильника.

Приклад: Обчислити значення функції $y = \frac{x^3 - 4x + 1}{x + 1}$, користуючись оператором циклу **WHILE**, при x, що змінюється в діапазоні $x_{нач} \leq x \leq x_{кон}$ з кроком Δx .

```
xn = float(input('введіть xn = '))
xk = float(input('введіть xk = '))
dx = float(input('введіть dx = '))
x=xn
while x<=xk:
    y=(x**3-4*x+1)/(x+1)
    print ('x=', x)
    print ('y=', y)
    x+=dx
```

Контрольні запитання:

1. Що таке цикл? Навіщо вони потрібні?
2. Як описується та виконується циклічна інструкція while?
3. Як можна організувати нескінченні цикли? Наведіть декілька варіантів і поясніть їх.
4. Як можна вийти з нескінченних циклів?
5. Що відбувається при запуску нескінченного циклу?
6. Чи може оператор циклу не мати тіла? Чому?
7. Для чого служать оператори переривання break та continue? Наведіть приклад.

Комп'ютерний практикум №6. Списки. Інструкція for

Мета роботи: Особливості організації умовних операторів і множинних розгалужень. Розробка програм.

Завдання

1. Створіть два будь-яких списки та зв'яжіть їх зі змінними.
2. Вийміть з першого списку другий елемент.
3. Змініть в другому списку останній об'єкт. Виведіть список на екран.
4. З'єднайте обидва списки в один, привласнивши результат новій змінній.

Виведіть список, що вийшов на екран.

5. Створіть список $L = [3, 'hello', 7, 4, 'привіт', 4, 3, -1]$

Досліджуйте кілька прикладів використання зрізів. >>> $L[:3]$

>>> $L[:]$

>>> $L[:2]$

>>> $L[::2]$

>>> $L[::1]$

>>> $L[-1]$

>>> $L[-1:]$

6. $L = [3, 6, 7, 4, -5, 4, 3, -1]$

6.1. Визначте суму елементів списку L . ЯКЩО сума перевищує значення 2, ТО вивести на екран число елементів списку.

6.2. Визначити різницю між мінімальним і максимальним елементами списку. ЯКЩО абсолютне значення різниці більше 10, ТО вивести на екран відсортоване за зростанням список, ІНАКШЕ вивести на екран фразу «Різниця менше 10».

7. $L = [3, 'hello', 7, 4, 'привіт', 4, 3, -1]$

Визначте наявність рядка «привіт» в списку. ЯКЩО такий рядок в списку присутній, ТО вивести її на екран, повторивши 10 разів.

8. $L = [3, 'hello', 7, 4, 'привіт', 4, 3, -1]$

Визначте наявність рядка «привіт» в списку. ЯКЩО такий рядок в списку присутній, ТО видалити її зі списку, ІНАКШЕ додати рядок в список.

Підрахувати, скільки разів в списку зустрічається число 4, ЯКЩО більше одного разу, ТО очистити список.

$$S = \sum_{i=1}^n (\sin i + \cos i)$$

9. Скласти програму обчислення значення суми

10. Напишіть програму у відповідності до варіанту та побудуйте блок-схему:

Завдання 1 Знайти суму n членів ряду, заданого за варіантом. (Розрахувати число $n = \text{варіант} + 10$).

Варіанти завдань:

№	Ряд	№	Ряд
1	$\sum_{i=1}^n \frac{\sqrt{2i}}{\sqrt{i+5} - \sqrt{i}}$	14	$\sum_{i=1}^n \frac{e^{-\sqrt{i}}}{\sqrt{i}}$

№	Ряд	№	Ряд
2	$\sum_{i=1}^n \frac{\sqrt[5]{i}}{(i+3)\sqrt{i}}$	15	$\sum_{i=1}^n \ln \frac{i+1}{2i+5}$
3	$\sum_{i=1}^n \sqrt{\frac{i+2}{2i+9}}$	16	$\sum_{i=1}^n \frac{i+1}{\sqrt{i^4+1}}$
4	$\sum_{i=1}^n \sqrt[3]{\frac{i-2}{i+900}}$	17	$\sum_{i=1}^n \frac{3^i}{i}$
5	$\sum_{i=1}^n \frac{\sqrt{3i+7}}{\sqrt{2i}-\sqrt{2i+3}}$	18	$\sum_{i=1}^n \frac{2^i}{\left(\frac{i+1}{i}\right)^{i^2}}$
6	$\sum_{i=1}^n \sqrt[3]{\frac{i+1}{2i+1}}$	19	$\sum_{i=1}^n \frac{i}{(i^2+5)+\ln i}$
7	$\sum_{i=1}^n (\sqrt{i^2+3i}-i)$	20	$\sum_{i=1}^n (-1)^{i+1} \frac{1}{\sqrt{2i+1}}$
8	$\sum_{i=1}^n \operatorname{tg}(\pi i + 0.5)$	21	$\sum_{i=1}^n \frac{(-1)^{i+1}}{(i+1)^i}$
9	$\sum_{i=1}^n \frac{\ln i}{\cos \frac{\pi}{i}}$	22	$\sum_{i=1}^n \frac{1}{\sqrt{i+3}}$
10	$\sum_{i=1}^n \frac{i \ln(1+i^2)}{1+i^2}$	23	$\sum_{i=1}^n \frac{3i-1}{i+20}$
11	$\sum_{i=1}^n \frac{\sqrt{i}}{\sqrt{i+1}-\sqrt{i}}$	24	$\sum_{i=1}^n \frac{1}{i(i+1)(i+3)}$
12	$\sum_{i=1}^n \ln\left(\frac{i}{2i+1}\right)$	25	$\sum_{i=1}^n \frac{1}{i(i+3)}$
13	$\sum_{i=1}^n \frac{1}{i} \sin \frac{\pi}{i}$	26	$\sum_{i=1}^n \frac{1}{i^3+4i}$

Варіанти завдань 2:

№	Завдання
1	Знайти суму цифр заданого натурального числа n.
2	Дано натуральне число. Підрахувати загальну кількість його дільників.
3	Написати програму, яка виводить таблицю квадратів перших десяти цілих додатних чисел.
4	Дано натуральне число. Вивести на екран всі прості числа до заданого включно.
5	Дано натуральне число n. Вивести на екран n перших простих чисел. Наприклад, при введенні числа 10 програма повинна вивести відповідь: 2 3 5 7 11 13 17 19 23 29

№	Завдання
6	Дано натуральне N . З'ясуйте, скільки цифр воно містить.
7	Знайти кількість тризначних чисел, сума простих дільників яких кратна 5
8	Дано натуральні числа x і n (яке також може бути 0). Обчислити x^n
9	Написати програму, яка виводить таблицю квадратів перших п'яти цілих додатних непарних чисел.
10	Дано натуральне число n (яке також може бути нуль). Обчислити $n!$
11	Дана послідовність з 20 цілих чисел. Визначити зі скількох простих чисел вона складається.
12	Написати програму, яка обчислює суму перших n цілих додатних чисел. Кількість чисел, що додаються, має задаватись під час роботи програми.
13	Необхідно ввести з клавіатури N чисел, знайти з них найбільше і вивести його.
14	Введіть з клавіатури 6 чисел і визначте їх середнє арифметичне.
15	Знайдіть мінімальне з N чисел.
16	Напишіть програму, яка вводить цілі числа з клавіатури і складає їх, поки не буде введено число 0.
17	Дано натуральне число N . Визначити всі прості числа не перевищують N .
18	Написати програму, яка обчислює суму і середнє арифметичне послідовності додатних чисел, які вводяться з клавіатури.
19	Написати програму, яка генерує десять випадкових чисел в діапазоні від 1 до 10, виводить ці числа на екран і обчислює їх середнє арифметичне.
20	Написати програму, яка виводить на екран таблицю вартості (наприклад, яблук), в діапазоні від 100 г до 1 кг з кроком 100.
21	Нехай дано шість цілих чисел, які вводяться по одному. Отримати суму тих з них, які кратні 5.
22	Написати програму, яка вводить з клавіатури послідовність з п'яти дробових чисел і після введення кожного числа виводить середнє арифметичне одержаної частини послідовності.
23	Написати програму, яка вводить з клавіатури 5 дробових чисел і обчислює їх середнє арифметичне.
24	Написати програму, яка виводить таблицю степенів двійки (від нульової до десятої).
25	Написати програму, яка виводить на екран таблицю множення на 7.

Контрольні запитання:

1. Як працює оператор `for`?
2. Для організації яких циклів застосовується оператор `for`?
3. Які методи списків ви знаєте?
4. Які функції списків ви знаєте?

Теоретичні відомості

СПИСКИ

Списки в мові програмування Python є впорядкованими послідовностями. Списки складаються з різних об'єктів (значень, даних), і заключені в квадратних дужках []. Об'єкти відокремлюються один від одного за допомогою коми.

Списки можуть складатися з різних об'єктів: чисел, рядків і навіть інших списків. В останньому випадку, списки називають вкладеними.

```
[23, 656, -20, 67, -45] # список цілих чисел
```

```
[4.15, 5.93, 6.45, 9.3, 10.0, 11.6] # список з дробових чисел
```

```
["Katy", "Sergei", "Oleg", "Dasha"] # список з рядків
```

```
["Москва", "Титова", 12, 148] # змішаний список
```

```
[[0, 0, 0], [0, 0, 1], [0, 1, 0]] # список, що складається зі списків
```

Над списками можна виконувати операції з'єднання і повторення:

```
>>> [45, -12, 'april'] + [21, 48.5, 33]
```

```
[45, -12, 'april', 21, 48.5, 33]
```

```
>>> [[0,0], [0,1], [1,1]] * 2
```

```
[[0, 0], [0, 1], [1, 1], [0, 0], [0, 1], [1, 1]]
```

```
>>>
```

Можна отримувати доступ до об'єктів списку по їх індексам, витягувати зрізи, вимірювати довжину списку:

```
>>> li = ['a', 'b', 'c', 'd', 'e', 'f']
```

```
>>> len(li)
```

```
6
```

```
>>> li[0]
```

```
'a'
```

```
>>> li[4]
```

```
'e'
```

```
>>> li[0:3]
```

```
['a', 'b', 'c']
```

```
>>> li[3:]
```

```
['d', 'e', 'f']
```

```
>>>
```

Списки – це змінні послідовності. Якщо уявити рядок як об'єкт в пам'яті, то коли над ним виконуються операції конкатенації і повторення, то цей рядок не змінюється, а в результаті операції створюється інший рядок в іншому місці пам'яті. У рядок можна додати новий символ або видалити існуючий, не створивши при цьому нового рядка. Зі списком інша справа. При виконанні операцій інші списки можуть не створюватися, а змінювати безпосередньо оригінал. Зі списків можна видаляти елементи, додавати нові. При цьому слід пам'ятати, багато що залежить від того, як ви розпоряджаєтеся змінними. Бувають ситуації, коли списки все-таки копіюються. Наприклад, результат операції присвоюється іншій змінній.

Символ в рядку змінити не можна, елемент списку - можна:

```
>>> mystr = 'abracadabra'
```

```
>>> mylist = ['ab', 'ra', 'ka', 'da', 'bra']
>>> mystr [3] = '0'
Traceback (most recent call last):
File "<pyshell # 11>", line 1, in <module>
mystr [3] = '0'
TypeError: 'str' object does not support item assignment
>>> mylist [1] = 'ro'
>>> mylist
['ab', 'ro', 'ka', 'da', 'bra']
>>>
```

У списку можна замінити цілий зріз:

```
>>> mylist [0: 2] = [10,20]
>>> mylist
[10, 20, 'ka', 'da', 'bra']
>>>
```

Більш складна ситуація:

```
>>> alist = mylist[0:2] + [100,'it is ',200] + mylist[2:] # новий список
>>> a2list = mylist # створюється друга посилання-змінна на перший
список
>>> alist
[10, 20, 100, 'it is ', 200, 'ka', 'da', 'bra']
>>> a2list
[10, 20, 'ka', 'da', 'bra']
>>> a2list[0] = '!!!' # змінюємо список
>>> a2list
['!!!', 20, 'ka', 'da', 'bra']
>>> mylist # обидві змінні зв'язані з одним списком
['!!!', 20, 'ka', 'da', 'bra']
>>>
```

ЦИКЛ FOR

У програмах, написаних на Python, широко застосовується цикл `for`, який представляє собою цикл обходу заданої множини елементів (символів рядка, об'єктів списку або словника) и виконання в своєму тілі різних операцій над ними.

Цикл ***for*** дозволяє перебрати всі елементи зазначеного списку. Цикл спрацює рівно стільки разів, скільки елементів знаходиться в списку. Ім'я змінної, в яку на кожному кроці буде розміщуватися елемент списку, вибирає програміст. У нашому прикладі це змінна з ім'ям *i*.

На першому кроці змінній *i* буде присвоєно перший елемент списку `num`, рівний 0.8. Потім програма переходить в тіло циклу `for`, відокремлене відступами. У тілі циклу міститься виклик функції `print ()`, якій передається змінна *i*.

На наступному кроці змінній *i* присвоїться другий елемент списку, рівний 7.0. Відбудеться виклик функції `print ()` для відображення вмісту змінної *i* на екрані і т.д. до тих пір, поки не закінчатся елементи в списку.

У загальному вигляді цикл for для перебору всіх елементів заданого списку виглядає наступним чином:

```
for<<змінна>>in<<список>>
```

```
    <<тіло циклу>>
```

```
>>> for i in [1, 2, 'hi']:
```

```
    print(i)
```

```
1
```

```
2
```

```
hi
```

```
>>>
```

Цикл for дозволяє не тільки виводити елементи рядка або списку на екран, але і чинити між ними певні операції:

```
>>> num=[0.8, 7.0, 6.8, -6]
```

```
>>> for i in num:
```

```
    if i == 7.0:
```

```
        print (i, '- число 7.0')
```

```
7.0 - число 7.0
```

```
>>>
```

Наприклад, можемо вивести на екран тільки задане значення зі списку, виконавши порівняння на кожному кроці циклу.

ФУНКЦІЯ RANGE()

Досить часто при розробці програм необхідно отримати послідовність (діапазон) цілих чисел:

0	1	2	3	4	5	6	7	8	9
↑				↑				↑	
початок				крок 1				закінчення	

Для вирішення цього завдання в Python передбачена функція range (), що створює послідовність (діапазон) чисел. В якості аргументів функція приймає: початкове значення діапазону (за замовчуванням 0), кінцеве значення (НЕ включно) і крок (за замовчуванням 1). Якщо викликати функцію, то результату ми не побачимо:

```
>>> range(0,10,1)
```

```
range(0, 10)
```

```
>>> range(10)
```

```
range(0, 10)
```

```
>>>
```

Справа в тому, що для створення діапазону чисел необхідно використовувати цикл for:

```
>>> for i in range(0, 10, 1):
```

```
    print (i, end=' ')
```

```
0 1 2 3 4 5 6 7 8 9
```

```
>>> for i in range(10):
```

```
    print (i, end=' ')
```

```
0 1 2 3 4 5 6 7 8 9
```

```
>>> for i in range(2, 20, 2):
    print(i, end=' ')
2 4 6 8 10 12 14 16 18
>>>
```

Таким чином, в змінну i на кожному кроці циклу буде записуватися значення з діапазону, який створюється функцією `range()`.

При бажанні можна отримати діапазон в зворотному порядку проходження (зверніть увагу на аргументи функції `range()`):

```
>>> for i in range(20, 2, -2):
    print(i, end=' ')
20 18 16 14 12 10 8 6 4
>>>
```

Тепер за допомогою діапазону знайдемо суму чисел на інтервалі від 1 до 100:

```
>>> total=0
>>> for i in range(1, 101):
    total=total+i
>>> total
5050
>>>
```

Змінній i на кожному кроці циклу буде присвоюватися значення з діапазону від 1 до 100 (крайні значення не включаємо). У циклі ми накопичуємо лічильник. На першому кроці циклу спочатку обчислюється права частина виразу, тобто $total + i$.

Змінна $total$ на першому кроці дорівнює 0 (присвоїли їй значення 0 перед початком циклу), змінна i на першому кроці містить значення 1 (перше значення з діапазону), таким чином, права частина буде дорівнює значенню 1 і це значення присвоїти лівій частині виразу, тобто. змінній $total$.

На другому кроці $total$ вже буде дорівнює значенню 1, i - містить значення 2, тобто права частина виразу дорівнюватиме 3, це значення присвоїти знову $total$ і т.д. поки не дійдемо до кінця діапазону. У підсумку в $total$ після виходу з циклу буде міститися шукана сума.

В Python є краще рішення даної задачі:

```
>>> sum(list(range(1, 101)))
5050
>>>
```

Діапазони можна використовувати при створенні списків:

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(2, 10, 2))
[2, 4, 6, 8]
>>>
```

Приклад: Скласти програму обчислення значення суми $s = \sum_{i=1}^n (\sin i + \cos i)$

```
import math
```

```
n=int(input('Введіть значення n='))
summa=0
for i in range(n):
    summa=summa+math.sin(i)+math.cos(i)
print('summa=',summa)
```

Контрольні запитання:

1. Що таке цикл? Навіщо вони потрібні?
2. Як описується та виконується циклічна інструкція while?
3. Як можна організувати нескінченні цикли? Наведіть декілька варіантів і поясніть їх.
4. Як можна вийти з нескінченних циклів?
5. Що відбувається при запуску нескінченного циклу?
6. Чи може оператор циклу не мати тіла? Чому?
7. Для чого служать оператори переривання break та continue? Наведіть приклад.

Комп'ютерний практикум №7. Генерування випадкових чисел. Робота з одновимірними масивами

Мета роботи: Дослідити функції генерування послідовностей псевдовипадкових чисел і навчитись їх застосовувати. Ознайомитись з можливостями створення одновимірних масивів (списків) і навчитись обробляти їх елементи.

Завдання

1. Отримайте випадкове число між low (нижня межа) і high (верхня межа). Потім отримаєте випадкове ціле число між low (нижня межа) і high (верхня межа), включаючи граничні значення.

2. Задано список слів. Необхідно вибрати з нього випадкове слово. З обраного випадкового слова випадково вибрати букву і попросити користувача її вгадати.

Задано список слів: ['самовар', 'весна', 'літо']

Вибираємо випадкове слово: 'весна'

Вибираємо випадкову букву: 'з'

Виводимо на екран: ве?на

Користувач намагається вгадати букву.

Примітка: використовуйте метод choice () модуля random.

3. Дан список числових значень, що нараховує N елементів. Поміняйте місцями першу і другу половини списку.

4. Напишіть програму у відповідності до варіанту та побудуйте блок-схему:

Завдання а. Числа m та k ($3 \leq k \leq 10$) вводяться з клавіатури. Згенерувати та вивести на екран m цілих випадкових чисел з проміжку, вказаному у пункті а. Виведення на екран здійснювати по k чисел у рядку.

Завдання б. Числа n та k ($3 \leq k \leq 10$) вводяться з клавіатури. Згенерувати і вивести на екран n дійсних чисел (виводити на екран з вказаною точністю) з проміжку, вказаному у пункті б. Виведення на екран здійснювати по k чисел у рядку.

Варіанти завдань:

№	Завдання	
1	а) [-25, 30]	б) [1, 5] з точністю до сотих
2	а) [13, 399]	б) [-2, 2] з точністю до десятих
3	а) [-200, 100]	б) [0, 10] з точністю до тисячних
4	а) [0, 125]	б) [3, 6] з точністю до сотих
5	а) [-100, 0]	б) [-1, 3] з точністю до десятих
6	а) [-45, 45]	б) [-5, 0] з точністю до тисячних
7	а) [77, 127]	б) [-6, 10] з точністю до десятих
8	а) [-66, 666]	б) [0, 13] з точністю до сотих
9	а) [-33, 333]	б) [1, 2] з точністю до тисячних
10	а) [-444, 333]	б) [4, 8] з точністю до тисячних

№	Завдання	
11	a) $[-10, 10]$	б) $[-1, 0]$ з точністю до сотих
12	a) $[-1000, 500]$	б) $[5, 10]$ з точністю до десятих
13	a) $[13, 900]$	б) $[-1, 1]$ з точністю до сотих
14	a) $[-11, 111]$	б) $[2, 6]$ з точністю до тисячних
15	a) $[125, 500]$	б) $[10, 15]$ з точністю до десятих
16	a) $[-33, 333]$	б) $[1, 2]$ з точністю до тисячних
17	a) $[-444, 333]$	б) $[4, 8]$ з точністю до тисячних
18	a) $[-10, 10]$	б) $[-1, 0]$ з точністю до сотих
19	a) $[-1000, 500]$	б) $[5, 10]$ з точністю до десятих
20	a) $[13, 900]$	б) $[-1, 1]$ з точністю до сотих
21	a) $[-33, 333]$	б) $[1, 3]$ з точністю до сотих
22	a) $[-444, 333]$	б) $[4, 8]$ з точністю до тисячних
23	a) $[-10, 10]$	б) $[-1, 0]$ з точністю до сотих
24	a) $[-1000, 500]$	б) $[5, 10]$ з точністю до десятих
25	a) $[13, 900]$	б) $[-1, 1]$ з точністю до сотих

Завдання 2. Розробити програму, дотримуючись таких вимог:

- число n (кількість елементів списку) – іменована константа;
- елементи списку – псевдовипадкові числа, згенеровані на інтервалі $[a, b]$, де a і b вводяться з клавіатури ($a < b$);
- усі вхідні дані і також елементи списку виводяться на екран.

Варіанти завдань:

№	Завдання
1	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) суму від'ємних елементів; 2) добуток елементів списку, розташованих між максимальним і мінімальним елементами.
2	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) мінімальний за модулем елемент списку; 2) суму модулів елементів, розташованих після першого елемента, рівного нулю.
3	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) максимальний за модулем елемент списку; 2) суму елементів списку, розташованих між першим і другим додатними елементами.
4	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) номер максимального елемента списку; 2) добуток елементів списку, розташованих між першим і другим нульовими елементами.

№	Завдання
5	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) номер мінімального елемента списку; 2) суму елементів списку, розташованих між першим і другим від'ємними елементами.
6	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) максимальний елемент списку; 2) суму елементів списку, розташованих до останнього додатного елемента.
7	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) мінімальний елемент списку; 2) суму елементів списку, розташованих між першим і останнім додатними елементами.
8	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) суму додатних елементів списку; 2) добуток елементів списку, розташованих між максимальним за модулем і мінімальним за модулем елементами.
9	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) добуток елементів списку з парними номерами; 2) суму елементів списку, розташованих між першим і останнім нульовими елементами.
10	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) суму елементів списку з непарними номерами; 2) суму елементів списку, розташованих між першим і останнім від'ємними елементами.
11	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) номер мінімального за модулем елемента списку; 2) суму модулів елементів, розташованих після першого від'ємного елемента.
12	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) кількість елементів, що лежать в діапазоні від A до B (A і B – з клавіатури); 2) суму елементів списку, розташованих після максимального елемента.
13	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) кількість елементів списку, більших за C (C вводиться з клавіатури); 2) добуток елементів списку, розташованих після максимального за

№	Завдання
	модулем елемента.
14	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) номер максимального за модулем елементу списку; 2) суму елементів списку, розташованих після першого додатного елемента.
15	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) кількість елементів, що лежать в діапазоні від A до B (A і B – з клавіатури); 2) суму елементів списку, розташованих після максимального елемента.
16	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) кількість елементів списку, рівних 0; 2) суму елементів списку, розташованих після мінімального елемента.
17	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) мінімальний елемент списку; 2) суму елементів списку, розташованих між першим і останнім додатними елементами.
18	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) суму від’ємних елементів списку; 2) добуток елементів списку, розташованих між максимальним і мінімальним елементами.
19	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) суму додатних елементів списку; 2) добуток елементів списку, розташованих між максимальним за модулем і мінімальним за модулем елементами.
20	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) номер мінімального елемента списку; 2) суму елементів списку, розташованих між першим і другим від’ємними елементами.
21	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) максимальний елемент списку; 2) суму елементів списку, розташованих до останнього додатного елемента.
22	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) мінімальний елемент списку; 2) суму елементів списку, розташованих між першим і останнім

№	Завдання
	додатними елементами.
23	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) номер максимального за модулем елементу списку; 2) суму елементів списку, розташованих після першого додатного елемента.
24	В одновимірному масиві (списку), що складається з n дійсних елементів, обчислити: 1) кількість елементів, що лежать в діапазоні від A до B (A і B – з клавіатури); 2) суму елементів списку, розташованих після максимального елемента.
25	В одновимірному масиві (списку), що складається з n цілих елементів, обчислити: 1) номер мінімального елемента списку; 2) суму елементів списку, розташованих між першим і другим від'ємними елементами.

Теоретичні відомості

ГЕНЕРАЦІЯ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

В сучасних умовах для отримання випадкових чисел використовують різноманітні генератори, які поділяються на апаратні та програмні.

В апаратних генераторах джерелом випадкових чисел є шум в електронних приладах. Проте, використання апаратних генераторів вимагає наявності спеціального обладнання. В зв'язку з цим більш зручним вважається застосування програмних генераторів випадкових чисел.

Програмний генератор випадкових чисел являє собою програму, яка генерує послідовність чисел за деяким алгоритмом. Завдяки алгоритму така послідовність чисел цілком детермінована, тобто не може бути цілком випадковою. Її називають послідовністю псевдовипадкових чисел.

В Python є вбудований модуль, який дозволяє генерувати псевдовипадкові числа.

Підключити модуль можна за допомогою інструкції **import**. Наприклад, підключимо модуль **os** для отримання поточної директорії:

```
>>>
>>> import os
>>> os.getcwd()
'C:\\Python33'
```

Після ключового слова **import** вказується **назва модуля**. Однею інструкцією можна підключити декілька модулів, хоча цього не рекомендується робити, так як це знижує читаність коду. Імпортуємо модулі **time** і **random**.

```
>>>
>>> import time, random
>>> time.time()
1376047104.056417
```

```
>>> random.random()
0.9874550833306869
```

Після імпортування модуля його назва стає змінною, через яку можна отримати доступ до атрибутів модуля. Наприклад, можна звернутися до константи `e`, розташованої в модулі `math`:

```
>>>
>>> import math
>>> math.e
2.718281828459045
```

Модуль `random` включає в себе функцію `random`, яка повертає дійсне число буде в діапазоні від 0.0 до 1.0. Кожен раз при виконанні функції `random` ви отримаєте число з довгого ряду.

```
>>> import random
>>> random.random()
0.15651968855132303
```

Щоб подивитися, як працює функція, запустіть наступну програму:

```
>>> import random
>>> for i in range(10):
>>>     x = random.random()
>>>     print (x)
```

Щоб отримати випадкове число між 0.0 і верхньою межею `high`, просто помножьте `x` на `high`.

Список випадкових величин

Починає виконуватися зі списком з `n` нулів. При кожному проході через цикл замінюється один з елементів випадковим числом.

```
>>> s = [0] * n
>>> for i in range(n):
>>>     s[i] = random.random()
>>>     print (s)
```

Числа, що видаються функцією `random`, розподілені рівномірно; це означає, що все значення рівноймовірно.

Функції модулю random

`random.randint(a, b)` - повертає випадкове ціле число на відрізок від `a` до `b` включно.

```
print random.randint(10, 20)
print random.randint(1, 100500)
```

`random.choice(x)` - повертає випадковий елемент з непорожньої послідовності (списку або рядка) `x`.

```
print random.choice(['back', 'forward', 'left', 'right'])
print random.randint('abcdefghijklmnopqrstuvwxyz')
```

`random.shuffle(x)` - випадковим чином “перемішує” елементи послідовності (списку або рядка) `x`, зберігаючи результат в `x`.

```
mylist = [1, 0, 3, 5, 7, 1, 2]
random.shuffle(mylist)
print mylist
```

random.random() - повертає випадкове дійсне число на проміжку від 0.0 (включно) до 1.0 (не включаючи).

```
print random.random()
```

random.uniform(a, b) - повертає випадкове дійсне число на відрізку від a до b включно.

```
print random.uniform(0, 0.1)
```

```
print random.uniform(10, 100)
```

round(number) повертає число з плаваючою точкою, округлене до 0 цифр після коми (за замовчуванням). Може бути викликана з двома аргументами:

round (number [, ndigits]), де ndigits - число знаків після коми.

```
>>> round(4.56666)
```

```
5
```

```
>>> round(4.56666, 3)
```

```
4.567
```

```
>>>
```

ОДНОВИМІРНІ МАСИВИ

Масив – це набір змінних одного типу, що мають одне і те ж ім'я. Доступ до конкретного елемента масиву здійснюється за допомогою індексу.

Одновимірний масив може бути набором чисел, сукупністю символьних даних чи елементів іншої природи (навіть масив масивів). Так само, як і в послідовності, в одновимірному масиві можна вказати елемент з конкретним номером або записати загальний вигляд елемента, використовуючи як індекс змінну i , вказуючи діапазон її зміни: $a[i]$, $i=1, 2, \dots, n$.

Найпростіша форма – це одновимірний масив (лінійна таблиця). Він аналогічний одновимірному числовому вектору і має індивідуальне ім'я, а для позначення окремої компоненти до імені масиву додається індекс, який і виділяє потрібну компоненту.

Компоненти масиву називаються змінними з індексами. У звичайній математичній символіці записують: $X_1, X_2, \dots, X_n$; або $a_1, \dots, a_{50}$ і т. д.

Найменший індекс називається **нижньою межею**, найбільший – **верхньою межею**, а число елементів – **розміром масиву**.

У мові Python масиви задаються у вигляді списків.

Підходи до створення списку

Розглянемо різні способи створення списків. Найбільш очевидний спосіб:

```
>>> a = []
```

```
>>> for i in range(1,15):
```

```
    a.append(i)
```

```
>>> a
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

```
>>>
```

У циклі з діапазону від 1 до 14 вибираємо числа і за допомогою спискового методу **append()** додаємо їх до списку a.

Зі створенням списку з діапазону ми вже зустрічалися:

```
>>> a=list (range (1, 15))
```

```
>>> a
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

```
>>>
```

Можна також використовувати «спискові включення» (іноді називають «генератором списку»):

```
>>> a = [ i for i in range(1,15) ]
```

```
>>> a
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

```
>>>
```

Правила роботи для спискового включення:

$$a = [i \text{ for } i \text{ in } \text{range}(1, 15)]$$

У наступному прикладі вибираємо з діапазону все числа від 1 до 14, зводимо їх в квадрат і відразу формуємо з них новий список:

```
>>> a = [ i**2 for i in range(1,15) ]
```

```
>>> a
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196]
```

```
>>>
```

Спискові включення дозволяє задавати умова для вибору значення з діапазону (в прикладі виключили значення 4):

```
>>> a = [ i**2 for i in range(1,15) if i!=4 ]
```

```
>>> a
```

```
[1, 4, 9, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196]
```

```
>>>
```

Замість діапазонів спискове включення дозволяє вказувати існуючий список:

```
>>> a = [2, -2, 4, -4, 7, 5]
```

```
>>> b = [i**2 for i in a]
```

```
>>> b
```

```
[4, 4, 16, 16, 49, 25]
```

```
>>>
```

У прикладі ми вибираємо послідовно значення зі списку a, зводимо в квадрат кожен з його елементів і відразу додаємо отримані значення в новий список.

За аналогією можна перебирати символи з рядка і формувати з них список:

```
>>> c = [c*3 for c in 'list' if c != 'i']
```

```
>>> c
```

```
['lll', 'sss', 'ttt']
```

```
>>>
```

Розглянемо ще приклад, як отримати список, що складається з випадкових цілих чисел:

```
>>> from random import randint
```

```
>>> A = [ randint(1, 9) for i in range(5) ]
```

```
>>> A
```

```
[2, 1, 1, 7, 8]
```

```
>>>
```

В даному прикладі функція `range()` виступає як лічильник числа повторень (цикл `for` спрацює рівно 5 разів). Зверніть увагу, що при формуванні нового списку змінна `i` не використовується. В результаті п'ять разів буде проведений виклик функції `randint()`, яка згенерує ціле випадкове число з інтервалу, і вже це число додасться в новий список.

Перейдемо до ручного введення значень для списку. Задамо довжину списку і введемо з клавіатури все його значення:

```
a = [] # оголошуємо порожній список
n = int(input()) # зчитуємо кількість елемент в списку
for i in range(n):
    new_element = int(input()) # зчитуємо черговий елемент
    a.append(new_element) # додаємо його в список
# Останні два рядки можна було замінити однією:
# a.append(int(input()))
print(a)
```

В результаті запуску програми:

```
>>>
```

```
===== RESTART: C:\Python35-32\myprog.py =====
```

```
3
```

```
4
```

```
2
```

```
1
```

```
[4, 2, 1]
```

```
>>>
```

У цьому прикладі `range()` знову виступає як лічильник числа повторень, а саме – задає довжину списку.

Тепер запишемо рішення цього завдання через спискові включення в один рядок:

```
>>> A = [ int(input()) for i in range(int(input())) ]
```

```
3
```

```
4
```

```
2
```

```
1
```

```
>>> A
```

```
[4, 2, 1]
```

```
>>>
```

Функції списків

Для списків доступні основні вбудовані функції, а також методи списків (Табл.2).

Таблиця 2

Метод	Призначення
list.append(x)	Додає елемент в кінець списку
list.extend(l)	розширює список list, додаючи в кінець всі елементи списку l

Метод	Призначення
list.insert(i, x)	Вставляє на i-ий елемент значення x
list.remove(x)	Видаляє перший елемент у списку, який має значення x. ValueError, якщо такого елемента не існує
list.pop([i])	Видаляє i-ий елемент і повертає його. Якщо індекс не вказано, видаляється останній елемент
list.index(x, [start [, end]])	Повертає положення першого елемента зі значенням x (при цьому пошук ведеться від start до end)
list.count(x)	Повертає кількість елементів зі значенням x
list.sort([key = функція])	Сортує список на основі функції
list.reverse()	Розвертає список
list.copy()	Поверхнева копія списку
list.clear()	Очищає список

Приклади роботи зі списками

```
>>> a = [66.25, 333, 333, 1, 1234.5]
>>> print(a.count(333), a.count(66.25), a.count('x'))
2 1 0
>>> a.insert(2, -1)
>>> a.append(333)
>>> a
[66.25, 333, -1, 333, 1, 1234.5, 333]
>>> a.index(333)
1
>>> a.remove(333)
>>> a
[66.25, -1, 333, 1, 1234.5, 333]
>>> a.reverse()
>>> a
[333, 1234.5, 1, 333, -1, 66.25]
>>> print (a.sort())
[-1, 1, 66.25, 333, 333, 1234.5]
```

Контрольні запитання:

1. Яким чином можна згенерувати випадкове число?
2. Для чого існує функція random()?
3. Яким чином генеруються цілі випадкові числа на певному інтервалі?
4. Як згенерувати дійсні випадкові числа на певному інтервалі?
5. Що таке масиви? Як розташовуються елементи масивів у пам'яті?
6. Як звернутись до першого та останнього елемента масиву?

Комп'ютерний практикум №8. Розробка програм з використанням двовимірних масивів

Мета роботи: Засвоїти техніку використання вкладених списків.

Завдання

1. Нехай задано список дійсних випадкових чисел (додатних та від'ємних) $[[a_{11}, \dots, a_{1n}] \dots [a_{m1}, \dots, a_{mn}]]$. Написати програму, яка визначить список мінімальних елементів кожного рядка (результат записати в інший список). Вивести на екран початкову матрицю та отриманий список.

```
import random
n=3 # кількість стовпчиків
m=4 # кількість рядків
x=[0]*m
for i in range(m):
    x[i]=[0]*n
y=[0]*m; i=0
while i<m :
    j=0
    k=0
    while j < n :
        x[i][j] = random.randint(-5,5)
        j+=1
    y[i] = min(x[i][0:m])
    k+=1
    i+= 1
print('x= ', x)
print('min = ', y)
```

2. Напишіть програму у відповідності до варіанту та побудуйте блок-схему:

Завдання Розробити програму, дотримуючись таких вимог:

- розміри масиву n і m – ввести з клавіатури;
- елементи масиву – псевдовипадкові числа, згенеровані на інтервалі $[a, b]$, де a і b ($a < b$) вводяться з клавіатури;
- усі вхідні та вихідні дані і також елементи початкової матриці та отриманої виводити на екран.

Варіанти завдань:

№	Завдання
1	Реалізувати програму, яка міняє місцями перший і останній стовпці квадратної матриці.
2	Реалізувати програму, яка додає перший і останній рядки квадратної матриці і записує результат у останній стовпець.
3	Реалізувати програму, яка міняє значення елементів квадратної матриці ні значення відповідних елементів заданого одновимірного масиву.
4	Реалізувати програму, яка додає відповідні елементи двох заданих

№	Завдання
	масивів і заносить результат у третій масив. Усі три масиви мають однакові розмірності (nхm).
5	Реалізувати програму, яка міняє місцями перший рядок і останній стовпець квадратної матриці.
6	Реалізувати програму, яка міняє місцями діагоналі квадратної матриці.
7	Реалізувати програму, яка сумує елементи рядків двовимірного масиву і заносить результат в одновимірний масив, розмірність якого дорівнює числу рядків двовимірного масиву.
8	Реалізувати програму, яка знаходить максимальний за модулем елемент заданого двовимірного масиву.
9	Реалізувати програму, яка міняє місцями останній рядок і перший стовпець квадратної матриці.
10	Реалізувати програму, яка додає перший і останній стовпці квадратної матриці і записує результат на місце першого рядка.
11	Реалізувати програму, яка міняє елементи заданого стовпця на значення відповідних елементів одновимірного масиву.
12	Реалізувати програму, яка перемножує відповідні елементи двох заданих масивів і заносить результат у третій масив. Розмірності усіх масивів однакові.
13	Реалізувати програму, яка міняє місцями останній рядок і перший стовпець квадратної матриці.
14	Реалізувати програму, яка сумує елементи стовпців двовимірного масиву і заносить результат в одновимірний масив, розмірність якого дорівнює числу стовпців двовимірного масиву.
15	Реалізувати програму, яка знаходить номер рядка заданого двовимірного масиву, що має максимальну за модулем суму елементів.
16	Реалізувати програму, яка змінить місцями два стовпчики: стовпчик, який містить максимальний від'ємний елемент, і стовпчик, який містить мінімальний додатний елемент матриці.
17	Реалізувати програму, яка змінить місцями перший рядок з рядком, що містить максимальний елемент матриці.
18	Реалізувати програму, яка змінить місцями останній рядок з рядком, який містить мінімальний додатний елемент матриці.
19	Реалізувати програму, яка визначить, чи є задана квадратна матриця симетричною відносно головної діагоналі.
20	Реалізувати програму, яка визначить добуток від'ємних елементів другого рядка та кількість елементів в другому стовпчику, які не кратні «5».
21	Реалізувати програму, яка змінить місцями останній стовпчик і стовпчик, який містить мінімальний додатний елемент матриці.
22	Реалізувати програму, яка визначить максимальний елемент третього стовпчика та суму непарних елементів першого рядка.
23	Реалізувати програму, яка визначить рядок, сума елементів якого мінімальна.

№	Завдання
24	Реалізувати програму, яка визначить суму елементів кожного стовпчика (результат записати в інший список).
25	Реалізувати програму, яка сформує новий список, в якому всі елементи матриці вище побічної діагоналі є нульовими.
26	Реалізувати програму, яка змінить місцями перший стовпчик і стовпчик, який містить мінімальний за абсолютною величиною елемент матриці.
27	Реалізувати програму, яка визначить суму елементів парних рядків, записати результат у новий список.
28	Реалізувати програму, яка визначить суму елементів в кожному стовпчику, записати результат у новий список.

Теоретичні відомості ДВОВИМІРНІ МАСИВИ

Масив – це структура даних, яку можна розглядати як набір змінних однакового типу, що мають загальне ім'я. Доступ до будь-якого елементу масиву здійснюється за його номером.

У масиві дані різняться своїм порядковим номером (індексом). Якщо кожний елемент масиву визначається за допомогою одного номера, то такий масив називається *одновимірним*, якщо за двома — то *двовимірним*.

Двовимірний масив — це таблиця з рядків і стовпчиків. У таблицях перший номер вказує на рядок, а другий — на положення елемента в рядку. Усі рядки таблиці мають однакову довжину.

У мові програмування Python двовимірний масив можна представити у вигляді списку рядків, кожен елемент якого є в свою чергу списком.

Приклад. Створення числової таблиці з двох рядків і трьох стовпців, над якими здійснюються різні дії: `a = [[1, 2, 3], [4, 5, 6]]`.

```
print(a[0])
print(a[1])
```

Перший рядок списку `a[0]` є списком з чисел `[1, 2, 3]`. Тобто `a[0][0]=1`, значення `a[0][1]=2`, `a[0][2]=3`, `a[1][0]=4`, `a[1][1]=5`, `a[1][2]=6`.

Для обробки і виведення списку, як правило, використовують два вкладених цикли. Перший цикл перебирає номер рядка, другий цикл проходить за елементами всередині рядка.

Приклад. Виведення двовимірного числового списку на екран порядково, розділяючи числа пробілами всередині одного рядка:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
for i in range(len(a)):
    for j in range(len(a[i])):
        print(a[i][j], end=' ')
    print()
```

Змінна циклу **for** в Python може перебирати не тільки діапазон, який створюється за допомогою функції **range()**, але і взагалі перебирати будь-які елементи будь-якої послідовності. Послідовностями в Python є списки, рядки, а також деякі інші об'єкти.

Приклад. Виведення двовимірного масиву, використовуючи цикл *for*: $a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]$.

```
for row in a:
    for elem in row:
        print(elem, end=' ')
    print()
```

Для виведення одного рядка можна використовувати метод ***join()***:

```
for row in a:
    print(' '.join([str(elem) for elem in row]))
```

Приклад. Підрахунок суми всіх чисел в списку:

Рішення 1.

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for i in range(len(a)):
    for j in range(len(a[i])):
        s += a[i][j]
print(s)
```

Рішення 2.

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for row in a:
    for elem in row:
        s += elem
print(s)
```

СТВОРЕННЯ ВКЛАДЕНОГО СПИСКУ

Нехай дано два числа: кількість рядків n і кількість стовпців m . Необхідно створити список розміром $n \times m$, заповнений нулями.

Перший спосіб: спочатку створимо список з n елементів (для початку просто з n нулів). Потім зробимо кожен елемент списку посиланням на інший одновимірний список з m елементів:

```
n = 3
m = 4
a = [0] * n
for i in range(n):
    a[i] = [0] * m
```

Другий спосіб: створити порожній список, потім n раз додати в нього новий елемент, який є списком-рядком:

```
n = 3
m = 4
a = []
for i in range(n):
    a.append([0] * m)
```

Але ще простіше скористатися генератором: створити список з n елементів, кожен з яких буде списком, що складається з m нулів:

```
n = 3
```

```
m = 4
a = [[0] * m for i in range(n)]
```

У цьому випадку кожен елемент створюється незалежно від інших (заново конструюється список $[0]*m$ для заповнення чергового елемента списку), а не копіюються посилання на один і той же список.

ВВЕДЕННЯ ДВОВИМІРНОГО МАСИВУ

Нехай програма отримує на вхід двовимірний масив у вигляді n рядків, кожен з яких містить m чисел, розділених пробілами. Як їх рахувати? Наприклад, так:

```
# В першому рядку введення йде кількість рядків масиву
n = int(input())
a = []
for i in range(n):
    a.append([int(j) for j in input().split()])
```

Або, без використання складних вкладених викликів функцій:

```
# В першому рядку введення йде кількість рядків масиву
n = int(input())
a = []
for i in range(n):
    row = input().split()
    for i in range(len(row)):
        row[i] = int(row[i])
    a.append(row)
```

Можна зробити те ж саме і за допомогою генератора:

```
# В першому рядку введення йде кількість рядків масиву
n = int(input())
a = [[int(j) for j in input().split()] for i in range(n)]
```

ПРИКЛАД ОБРОБКИ ДВОВИМІРНОГО МАСИВУ

Приклад. Нехай дано квадратну матрицю з n рядків і n стовпців. Необхідно елементам, що знаходяться на головній діагоналі, що проходить з лівого верхнього кута в правий нижній (тобто тих елементів $a[i][j]$, для яких $i=j$) присвоїти значення 1, елементам, що знаходяться вище головної діагоналі - значення 0, елементам, що знаходяться нижче головної діагоналі - значення 2, тобто необхідно отримати такий масив (наприклад $n=4$):

```
1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1
```

Рішення 1. Елементи, які лежать вище головної діагоналі - це елементи $a[i][j]$, для яких $i < j$, а для елементів нижче головної діагоналі $i > j$. Таким чином, ми можемо порівнювати значення i та j та за ними визначати значення $a[i][j]$.

```
n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
    for j in range(n):
```

```

    if i < j:
        a[i][j] = 0
    elif i > j:
        a[i][j] = 2
    else:
        a[i][j] = 1
for row in a:
    print(' '.join([str(elem) for elem in row]))

```

Рішення 2. Спочатку заповнимо головну діагональ, для чого нам знадобиться один цикл:

```

for i in range(n):
    a[i][i] = 1

```

Потім заповнимо значеннями 0 всі елементи вище головної діагоналі, для цього нам знадобиться в кожному з рядків з номером i привласнити значення елементам $a[i][j]$ для $j=i+1, \dots, n-1$. Тут знадобляться вкладені цикли:

```

for i in range(n):
    for j in range(i + 1, n):
        a[i][j] = 0

```

Аналогічно присвоюємо значення 2 елементам $a[i][j]$ для $j=0, \dots, i-1$:

```

for i in range(n):
    for j in range(0, i):
        a[i][j] = 2

```

Рішення 3. Можна також зовнішні цикли об'єднати в один:

```

n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
    for j in range(0, i):
        a[i][j] = 2
    a[i][i] = 1
    for j in range(i + 1, n):
        a[i][j] = 0
for row in a:
    print(' '.join([str(elem) for elem in row]))

```

Рішення 4. Використовуємо операцію повторення списків для побудови чергового рядка списку. i -й рядок списку складається з i чисел 2, потім йде одне число 1, потім йде $n-i-1$ число 0:

```

n = 4
a = [0] * n
for i in range(n):
    a[i] = [2] * i + [1] + [0] * (n - i - 1)
for row in a:
    print(' '.join([str(elem) for elem in row]))

```

Можна замінити цикл на генератор:

```

n = 4
a = [0] * n

```

```
a = [[2] * i + [1] + [0] * (n - i - 1) for i in range(n)]
for row in a:
```

```
    print(' '.join([str(elem) for elem in row]))
```

ВКЛАДЕНІ ГЕНЕРАТОРИ ДВОВИМІРНИХ МАСИВІВ

Для створення двовимірних масивів можна використовувати **вкладені генератори**, розмістивши генератор списку, що є рядком, всередині генератора всіх рядків. Зробити список з n рядків і m стовпців можна за допомогою генератора, що створює список з n елементів, кожен елемент якого є списком з m нулів:

```
[[0] * m for i in range(n)]
```

Але при цьому внутрішній список також можна створити за допомогою, наприклад, такого генератора: `[0 for j in range(m)]`. Вклавши один генератор в інший, отримаємо вкладені генератори:

```
[[0 for j in range(m)] for i in range(n)]
```

Але якщо число 0 замінити на деякий вираз, залежний від i (номер рядка) та j (номер стовпчика), то можна отримати список, заповнений за деякою формулою.

Наприклад, нехай потрібно задати наступний масив:

```
0 0 0 0 0 0
0 1 2 3 4 5
0 2 4 6 8 10
0 3 6 9 12 15
0 4 8 12 16 20
```

У цьому масиві $n=5$ рядків, $m=6$ стовпців, та елемент в рядку i та стовпці j обчислюється за формулою: $a[i][j]=i*j$.

Для створення такого масиву можна використовувати генератор:

```
[[i * j for j in range(m)] for i in range(n)]
```

Контрольні запитання:

1. Як оголосити одновимірний динамічний масив?
2. Як оголосити двовимірний масив?
3. Скільки індексів характеризують конкретний елемент двовимірного масиву?
4. Як в програмі використовувати значення конкретного елемента двовимірного масиву?
5. Наведіть фрагмент коду для заповнення масиву цілими (дійсними) випадковими числами.
6. Наведіть фрагмент коду для обчислення суми (добутку) елементів одновимірного масиву?
7. Наведіть фрагмент коду для підрахунку суми (добутку) елементів певного рядка (стовпця) двовимірного масиву?

Комп'ютерний практикум №9. Рядки як послідовності символів. Кортежі. Словники

Мета роботи: Дослідити основні функції роботи з рядками, кортежами та словниками і реалізовувати найпростіші операції з ними.

Завдання

1. Зв'яжіть змінну з будь-яким рядком, що складається не менше ніж з 8 символів. Виведіть з рядка перший символ, потім останній, третій з початку і третій з кінця. Виміряйте довжину вашої рядка.

2. Привласніть довільний рядок довжиною 10-15 символів змінній і витягніть з неї такі зрізи:

- перші вісім символів;
- чотири символи з центру рядка;
- символи з індексами кратними трьома.

3. Створіть словник, зв'язавши його зі змінною `university`, і наповніть його даними, які б відображали кількість студентів в 8 різних групах (наприклад, 61бс 62бс, 51бс, 52бс і т.д.).

4. Дізнайтеся скільки студентів в якій групі.

5. Уявіть, що на факультеті відбулися зміни, внесіть їх в словник:

- в трьох групах змінилася кількість учнів;
- на факультеті з'явилося дві нових групи;
- на факультеті розформували одну з груп.

6. Виведіть вміст словника на екран.

7. Створити базу даних з 6-ти студентів, в якій вказується ПІБ, оцінка за іспити з трьох предметів. Вибрати з них студента з найнижчим балом і відрахувати його (видалити з бази). Оновлену базу даних вивести на екран.

8. Скласти журнал успішності з кількох предметів, в якому вказуються всі отримані оцінки по кожному предмету. Залежно від введеного за запитом номера предмета в журналі вивести на екран його назву та середній бал.

9. Напишіть програму у відповідності до варіанту та побудуйте блок-схему:

Завдання 3 клавіатури вводиться текстовий рядок. Розробити програму, яка реалізує вказані дії.

Варіанти завдань:

№	Завдання
1	а) підраховує кількість слів, які мають непарну довжину; б) виводить на екран частоту входження кожної літери; в) видаляє текст, що розміщено в круглих дужках.
2	а) перевіряє, чи співпадає кількість відкритих і закритих круглих дужок у введеному рядку; б) виводить на екран найдовше слово; в) видаляє всі слова, що складаються тільки з латинських літер.
3	а) підраховує кількість різних слів, що входять до заданого тексту; б) виводить на екран кількість використаних символів;

№	Завдання
	в) видаляє всі слова, що мають подвоєні літери.
4	а) підраховує кількість слів у тексті; б) виводить на екран слово, що містить найбільшу кількість голосних літер; в) видаляє з тексту всі непотрібні пробіли.
5	а) підраховує кількість розділових знаків у тексті; б) виводить всі слова, що мають парну кількість літер; в) міняє місцями першу і останню літери кожного слова.
6	а) підраховує кількість великих літер у тексті; б) виводить на екран слова, що мають найменшу кількість літер; в) видаляє всі слова, що починаються з малої літери.
7	а) підраховує кількість чисел у тексті (не цифр, а саме чисел); б) виводить на екран всі слова, що складаються тільки з латинських літер; в) видаляє кожне друге слово.
8	а) підраховує кількість цифр у тексті; б) виводить на екран слова, що починаються з приголосних літер; в) знищує всі слова, які починаються і закінчуються за одну й ту ж літеру.
9	а) підраховує кількість слів у тексті, які закінчуються на голосну літеру; б) виводить на екран всі слова, довжина яких менша п'яти символів; в) видаляє всі слова, які містять хоча б одну латинську літеру.
10	а) підраховує кількість слів у тексті, які починаються з голосної літери; б) виводить на екран всі слова, що мають непарну кількість приголосних літер; в) видаляє всі числа з тексту.
11	а) замінює всі великі літери, що входять до тексту на відповідні малі; б) виводить на екран найдовше слово; в) видаляє всі слова, що містять непарну кількість приголосних літер.
12	а) кількість слів, які містять однакову кількість голосних і приголосних літер; б) виводить на екран найдовше слово; в) видаляє з тексту всі слова-паліндроми.
13	а) виводить всі символи, які розташовані після першого символу ":"; б) підраховує кількість речень, що містять непарну кількість слів; в) видаляє з тексту всі слова, які розташовані після ком.
14	а) підраховує кількість слів у кожному реченні; б) виводить на екран найдовше речення; в) видаляє всі слова, передостання літера яких голосна.
15	а) інвертує рядок, подаючи його у зворотному вигляді; б) підраховує кількість чисел у тексті; в) видаляє всі слова, що починаються з голосних літер.
16	а) перевіряє, чи співпадає кількість відкритих і закритих лапок у введеному рядку (перевірити для круглих та квадратних дужок);

№	Завдання
	б) виводить на екран найдовше слово; в) видаляє всі слова, остання літера яких голосна.
17	а) підраховує кількість слів у тексті, які починаються з голосної літери; б) виводить на екран всі слова, що мають непарну кількість приголосних літер; в) видаляє всі числа з тексту.
18	а) замінює всі маленькі літери, що входять до тексту на відповідні великі; б) підраховує кількість чисел у тексті; в) видаляє останнє число з тексту.
19	а) підраховує кількість однакових слів, що входять до заданого тексту; б) виводить на екран кількість використаних символів; в) видаляє всі слова, які містять хоча б одну латинську літеру.
20	а) підраховує кількість цифр у тексті; б) виводить всі слова, що мають непарну кількість літер; в) міняє місцями останню і першу літери кожного слова.
21	а) підраховує кількість пробілів у тексті; б) виводить всі слова, що мають парну кількість літер; в) видаляє всі числа з тексту.
22	а) підраховує кількість маленьких літер у тексті; б) виводить на екран слова, що мають найбільшу кількість літер; в) видаляє всі слова, що починаються з великої літери.
23	а) підраховує кількість слів у тексті, які починаються з приголосної літери; б) виводить на екран всі слова, що мають парну кількість приголосних літер; в) видаляє кожне третє слово.
24	а) виводить всі символи, які розташовані після першого символу "!"; б) виводить на екран всі слова, що мають непарну кількість приголосних літер; в) видаляє всі символи з тексту крім цифр.
25	а) підраховує кількість слів, які мають парну довжину; б) виводить на екран найдовше слово; в) видаляє текст, що розміщено в квадратних дужках.
26	а) виводить всі символи, які розташовані після першого символу ":"; б) підраховує кількість речень, що містять непарну кількість слів; в) видаляє з тексту всі слова, які розташовані після крапок.
27	а) перевіряє, чи співпадає кількість відкритих і закритих квадратних дужок у введеному рядку; б) підраховує кількість речень, що містять парну кількість слів; в) видаляє всі слова, перша літера яких голосна.
28	а) перевіряє, чи співпадає кількість відкритих і закритих лапок у введеному рядку (перевірити для круглих та квадратних дужок); б) виводить на екран найкоротше слово;

№	Завдання
	в) видаляє всі слова, остання літера яких приголосна.

Теоретичні відомості РЯДКИ І ОПЕРАЦІЇ НАД НИМИ

Python часто використовують для обробки текстів: пошуку в тексті, заміни окремих частин тексту і т.д. Для роботи з текстом в Python передбачений спеціальний строковий тип даних **str**.

Рядок - це складний тип даних, який представляє собою послідовність символів.

Рядки в мові програмування Python можуть полягати як в поодинокі, так і подвійні лапки. Однак, початок і кінець рядка повинні обрамлятися однаковим типом лапок.

```
>>> 'hello'
'hello'
>>> "Hello"
'Hello'
>>>
```

Існує спеціальна функція **len()**, що дозволяє виміряти довжину рядка. Ця функція визначає довжину рядка, яка передається їй як аргумент.

```
>>> help(len)
Help on built-in function len in module builtins:
len(obj, /)
    Return the number of items in a container.
```

```
>>> len('Привет!')
7
>>>
```

Також для рядків існують операції конкатенації (+) і дублювання (*).

```
>>> len('It is a long string')
19
>>> '!!!' + ' Hello World ' + '!!!'
'!!! Hello World !!!'
>>> '-' * 20
'-----'
>>>
```

Якщо хочемо помістити різні види лапок в рядок, то зробити це можна кількома способами:

```
>>> "Hello's"
"Hello's"
>>> 'Hello\'s'
"Hello's"
>>>
```

Перший – взяти в лапки різних типів, щоб Python зрозумів, де закінчується рядок.

Другий – використовувати спеціальні символи (керуючі escape-послідовності), які записуються, як два символи, але Python бачить їх як один:

```
>>> len("\n")
```

```
1
```

```
>>>
```

Спеціальні символи в роботі з рядками:

\n - перехід на новий рядок

\t - знак табуляції

\\ - похила риса вліво

\' - символ одиночної лапки

\\" - символ подвійних лапок

У послідовності важливий порядок символів, у кожного символу в рядку є унікальний порядковий номер – *індекс*. Можна звертатися до конкретного символу в рядку і витягувати його з допомогою оператора індексування, який представляє собою квадратні дужки з номером символу в них.

```
>>> 'morning, afternoon, night'[1]
```

```
'o'
```

```
>>> tday = 'morning, afternoon, night'
```

```
>>> tday[4]
```

```
'i'
```

```
>>>
```

У прикладі, вираз 'morning, afternoon, night' [1] призвело до вилучення другого символу. Справа в тому, що індексація починається не з одиниці, а з нуля. Тому, коли потрібно витягти перший символ, то оператор індексування повинен виглядати так: [0]. Також дозволено видобувати символи, починаючи відлік з кінця.

У цьому випадку відлік починається з -1 (останній символ).

```
>>> tday_ru = 'ранок, день, ніч'
```

```
>>> tday_ru[0]
```

```
'р'
```

```
>>> tday_ru[-1]
```

```
'ч'
```

```
>>> tday_ru[-3]
```

```
'н'
```

```
>>>
```

Зручніше працювати не з самими рядками, а зі змінними, які на них посилаються. Результат виконання вираження індексування можна привласнити іншій змінній.

```
>>> a = "very big string"
```

```
>>> a[6]
```

```
'i'
```

```
>>> b = a[0]
```

```
>>> b
```

```
'v'
>>>
```

Можна витягати з рядка не один символ, а кілька, тобто отримувати зріз (підрядок). Оператор вилучення зрізу з рядка виглядає так: [X: Y]. X - це індекс початку зрізу, а Y - його закінчення; причому символ з номером Y в зріз вже не входить. Якщо відсутній перший індекс, то зріз береться від початку до другого індексу; при відсутності другого індексу, зріз береться від першого індексу до кінця рядка.

```
>>> tday = 'morning, afternoon, night'
>>> tday[0:7]
'morning'
>>> tday[9:-7]
'afternoon'
>>> tday[-5:]
'night'
>>>
```

Крім того, можна витягати символи не підряд, а через певну кількість символів. У такому випадку оператор індексування виглядає так: [X: Y: Z]; Z - це крок, через який здійснюється вибір елементів.

```
>>> str4 = "Full Ball Fill Pack Ring"
>>> str4[::5]
'FBFPR'
>>> str4[0:15:2]
'Fl alFl '
>>>
```

Таблиця 3

Функція або метод	Призначення
S = 'str'; S = "str"; S = '''str'''; S = """str"""	Літерали рядків
S = "s\np\ta\nbbb"	Екрановані послідовності
S = r"C:\temp\new"	Неформатовані рядки (пригнічують екранування)
S = b"byte"	Рядок байтів
S1 + S2	Конкатенація (додавання рядків)
S1 * 3	Повторення рядка
S[i]	Звернення за індексом
S[i:j:step]	Витяг зрізу
len(S)	Довжина рядка
S.find(str, [start],[end])	Пошук підрядка в рядку. Повертає номер першого входження або -1
S.rfind(str, [start],[end])	Пошук підрядка в рядку. Повертає номер останнього входження або -1
S.index(str, [start],[end])	Пошук підрядка в рядку. Повертає номер першого входження або викликає ValueError
S.rindex(str, [start],[end])	Пошук підрядка в рядку. Повертає номер останнього входження або викликає ValueError

S.replace (шаблон, замена)	Заміна шаблону
S.split (символ)	Розбиття рядка по разделителю
S.isdigit ()	Чи складається рядок з цифр
S.isalpha ()	Чи складається рядок з літер
S.isalnum ()	Чи складається рядок з цифр або букв
S.islower ()	Чи складається рядок із символів в нижньому регістрі
S.isupper ()	Чи складається рядок із символів у верхньому регістрі
S.isspace ()	Чи складається рядок з символів, що не відображаються (пробіл, символ перекладу сторінки ('\f'), "новий рядок" ('\n'), "переклад каретки" ('\r'), "горизонтальна табуляція" ('\t') і "вертикальна табуляція" ('\v'))
S.istitle ()	Чи починаються слова в рядку з великої літери
S.upper ()	Перетворення рядка до верхнього регістру
S.lower ()	Перетворення рядка до нижнього регістру
S.startswith (str)	Чи починається рядок S з шаблону str
S.endswith (str)	Чи закінчується рядок S шаблоном str
S.join (список)	Збірка рядка зі списку з роздільником S
ord (символ)	Символ в його код ASCII
chr (число)	Код ASCII в символ
S.capitalize ()	Перекладає перший символ рядка в верхній регістр, а всі інші в нижній
S.center (width, [fill])	Повертає відцентрований рядок, по краях якого стоїть символ fill (пробіл за замовчуванням)
S.count (str, [start],[end])	Повертає кількість непересічних входжень підрядка в діапазоні [початок, кінець] (0 і довжина рядка за замовчуванням)
S.expandtabs ([tabsize])	Повертає копію рядка, в якій всі символи табуляції замінюються одним або декількома пропусками, в залежності від поточного стовпця. Якщо TabSize не вказано, розмір табуляції вважається рівним 8 пробілам
S.lstrip ([chars])	Видалення символів пробілів на початку рядка
S.rstrip ([chars])	Видалення символів пробілів в кінці рядка
S.strip ([chars])	Видалення символів пробілів на початку і в кінці рядка
S.partition (шаблон)	Повертає кортеж, що містить частину перед першим шаблоном, сам шаблон, і частину після шаблону. Якщо шаблон не знайдений, повертається кортеж, що містить сам рядок, а потім два порожніх рядки
S.rpartition (sep)	Повертає кортеж, що містить частину перед останнім шаблоном, сам шаблон, і частину після шаблону. Якщо шаблон не знайдений, повертається кортеж, що містить два порожні рядки, а потім сам рядок

S.swapcase()	Перекладає символи нижнього регістра в верхній, а верхнього - в нижній
S.title()	Першу букву кожного слова переводить в верхній регістр, а всі інші в нижній
S.zfill(width)	Робить довжину рядка не меншою width, в разі потреби заповнюючи перші символи нулями
S.ljust(width, fillchar=" ")	Робить довжину рядка не меншою width, в разі потреби заповнюючи останні символи символом fillchar
S.rjust(width, fillchar=" ")	Робить довжину рядка не меншою width, в разі потреби заповнюючи перші символи символом fillchar
S.format(*args, **kwargs)	форматування рядка

КОРТЕЖІ

Кортеж (tuple) умовно можна назвати незмінним «списком», тому що до нього застосовні багато спискові функції, крім зміни. Кортежі використовуються, коли ми хочемо бути впевнені, що елементи структури даних не будуть змінені в процесі роботи програми.

Деякі операції над кортежами:

```
>>> () # Створення порожнього кортежу
()
>>> (4) # це не кортеж, а цілочисельний об'єкт!
4
>>> (4,) # а ось це - кортеж, що складається з одного елемента!
(4,)
>>> B = ('1', 2, '4') # створюємо кортеж
>>> b
('1', 2, '4')
>>> Len(b) # визначаємо довжину кортежу
3
>>> T = tuple(range(10)) # створення кортежу з допомогою функції range
()
>>> T + b # злиття кортежів
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, '1', 2, '4')
>>> R = tuple([1,5,6,7,8, '1']) # кортеж зі списку
>>> r
(1, 5, 6, 7, 8, '1')
>>>
```

За допомогою кортежів можна присвоювати значення одночасно двом змінним:

```
>>> (x,y)=(10,5)
>>> x
10
>>> y
5
```

```
>>> x, y = 1,3 # якщо прибрати круглі дужки, то результат не зміниться
>>> x
1
>>> y
3
>>>
Поміняти місцями вміст двох змінних:
>>> x,y=y,x
>>> x
3
>>> y
1
>>>
Можна змінити, наприклад, список, що входить в кортеж:
>>> t=(1,[1,3],'3')
>>> t[1]
[1, 3]
>>> t[1][0]='1'
>>> t
(1, ['1', 3], '3')
>>>
```

СЛОВНИКИ

Одним зі складних типів даних (поряд з рядками і списками) в мові програмування Python є словники (dict). Словник - це змінний (як список) неупорядкований (на відміну від рядків і списків) набір пар "ключ: значення".

Щоб уявлення про словнику стало більш зрозумілим, можна провести аналогію зі звичайним словником, наприклад, англо-російським. На кожне англійське слово в такому словнику є російське слово-переклад: cat - кішка, dog - собака, table - стіл і т.д. Якщо англо-російський словник описувати за допомогою Python, то англійські слова будуть ключами, а російські - їх значеннями:

```
{ 'Cat': 'кішка', 'dog': 'собака', 'bird': 'птиця', 'mouse': 'миша' }
```

За допомогою фігурних дужок визначається словник.

Синтаксис словника на Python можна описати такою схемою:

Якщо створити словник в інтерпретаторі Python, то після натискання Enter можна спостерігати, що послідовність виведення пар "ключ: значення" не збігається з тим, як було введено:

```
>>> { 'Cat': 'кішка', 'dog': 'собака', 'bird': 'птиця', 'mouse': 'миша' }
{ 'Bird': 'птиця', 'mouse': 'миша', 'dog': 'собака', 'cat': 'кішка' }
>>>
```

У словнику абсолютно не важливий порядок пар, і інтерпретатор виводить їх у випадковому порядку. У словнику доступ до значень здійснюється за ключам, які полягають в квадратні дужки (по аналогії з індексами рядків і списків).

```
>>> dic = { 'cat': 'кішка', 'dog': 'собака', 'bird': 'птиця', 'mouse': 'миша' }
```

```
>>> dic['cat']
'кішка'
>>> dic['bird']
'птах'
>>>
```

Словники, як і списки, є змінним типом даних: можна змінювати, додавати і видаляти елементи (пари "ключ: значення"). Спочатку словник можна створити порожнім (наприклад, `d = {}`) і лише потім заповнити його елементами. Додавання і зміна має однаковий синтаксис: словник [ключ] = значення. Ключ може бути як вже існуючим (тоді відбувається зміна значення), так і новим (відбувається додавання елемента словника). Видалення елемента словника здійснюється за допомогою функції `del` ().

```
>>> dic = {'cat': 'кішка', 'dog': 'собака', 'bird': 'птах', 'mouse': 'миша'}
>>> dic['elephant'] = 'бегемот'
>>> dic['fox'] = 'лис'
>>> dic
{'fox': 'лис', 'dog': 'собака', 'cat': 'кішка', 'elephant': 'бегемот', 'mouse': 'миша',
'bird': 'птаха'}
>>> dic['elephant'] = 'слон'
>>> del(dic['bird'])
>>> dic
{'fox': 'лис', 'dog': 'собака', 'cat': 'кішка', 'elephant': 'слон', 'mouse': 'миша'}
>>>
```

Тип даних ключів і значень словників не обов'язково повинні бути рядками.

Значення словників можуть бути більш складними (містити структури даних, наприклад, інші словники або списки).

```
>>> d = {1:'one',2:'two',3:'three'}
>>> d
{1: 'one', 2: 'two', 3: 'three'}
>>> d = {10:[3,2,8], 100:[1,10,5], 1000:[23,1,5]}
>>> d
{1000: [23, 1, 5], 10: [3, 2, 8], 100: [1, 10, 5]}
>>> d = {1.1:2, 1.2:0, 1.3:8}
>>> d
{1.3: 8, 1.2: 0, 1.1: 2}
>>> d = {1.1:2, 10:'apple', 'box':100}
>>> d
{'box': 100, 10: 'apple', 1.1: 2}
>>>
```

Словники – це широко використовуваний тип даних мови Python. Для роботи з ними існує ряд вбудованих функцій.

Методи словників

dict.clear () - очищає словник.

dict.copy () - повертає копію словника.

classmethod dict.fromkeys (seq [, value]) - створює словник з ключами з seq і значенням value (за замовчуванням None).

dict.get (key [, default]) - повертає значення ключа, але якщо його немає, не кидає виняток, а повертає default (за замовчуванням None).

dict.items () - повертає пари (ключ, значення).

dict.keys () - повертає ключі в словнику.

dict.pop (key [, default]) - видаляє ключ і повертає значення. Якщо ключа немає, повертає default (за замовчуванням кидає виняток).

dict.popitem () - видаляє і повертає пару (ключ, значення). Якщо словник порожній, кидає виняток KeyError. Пам'ятайте, що словники не впорядковані.

dict.setdefault (key [, default]) - повертає значення ключа, але якщо його немає, не кидає виняток, а створює ключ з значенням default (за замовчуванням None).

dict.update ([other]) - оновлює словник, додаючи пари (ключ, значення) з other. Існуючі ключі перезаписувати. Повертає None (не нова словник!).

dict.values () - повертає значення в словнику.

Контрольні запитання:

1. Що таке рядок?
2. Наведіть приклади використання функцій обробки символів.
3. Які функції існують для введення і виведення символів?
4. Як отримати довжину рядка?
5. Що таке кортеж?
6. Що таке словник?

Комп'ютерний практикум №10. Функції. Модулі

Мета роботи: Дослідити принципи побудови функцій користувача, основні складові функцій, оголошення та опис функцій.

Завдання

1. Напишіть функцію, яка обчислює суму трьох чисел і повертає результат в основну програму.

2. Напишіть програму, яка визначить максимальне з двох введених чисел. Реалізувати у вигляді виклику власної функції, що повертає більше з двох переданих їй чисел.

3. Створіть функцію:

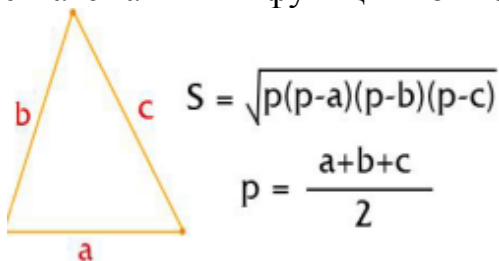
```
def func1(num):
    n = num * 5
    print (n)
```

Викличте її, передавши в якості аргументу значення глобальної змінної, потім будь-яке число і будь-який рядок.

4. Виконайте за допомогою інтерпретатора Python скрипт, попередньо виправивши код функції так, щоб вона повертала значення змінної n:

```
>>> def func(n):
        if n < 3:
            n = n*10
>>> a = 2
>>> b = func(a)
>>> a
2
>>> b # Чому зі змінною не пов'язане ніяке значення?
>>>
```

5. Знайдіть площу трикутника за допомогою формули Герона. Сторони задаються з клавіатури. Реалізувати обчислення площі у вигляді функції, на вхід якої подаються три числа, на виході - площа. Функція знаходиться в окремому модулі, де відбувається розділення між запуском та імпортуванням. Опис математичних функцій можна знайти в документації.



6. Вивести число Пі з точністю до сотих.

7. Напишіть програму у відповідності до варіанту та побудуйте блок-схему:

Завдання Розробити програму для реалізації індивідуального завдання. До всіх функцій додати документаційні рядки.

Варіанти завдань:

№	Завдання																
1	Написати функцію, яка виводить на екран суцільний прямокутник, створений із заданих символів С (наприклад, '#'), сторони прямокутника визначаються змінними W та H. Наприклад, якщо C='*', W=5, H=5, то функція виведе показане зображення. ***** ***** ***** ***** *****																
2	Функція отримує час у вигляді трьох цілих аргументів (години, хвилини, секунди) і повертає кількість секунд з моменту, коли на годиннику було 0 годин. Використайте цю функцію для підрахунку часу в секундах між двома моментами часу.																
3	Напишіть функцію, яка виводить ціле число між 1 та 32767 і друкує це число як послідовність цифр, між якими стоять два пробіли. Наприклад, ціле число 4562 повинно бути надруковане так: 4 5 6 2.																
4	Ціле число називається досконалим (рос. совершенным), якщо сума його дільників, включаючи 1 (але не саме число) , дорівнює цьому числу. Наприклад, число 6 = 1+2+3 є досконалим. Напишіть програму perfect, яка визначає, чи є число досконалим. Використайте цю функцію у програмі, яка знаходить і друкує усі досконалі числа з діапазону від 1 до 1000.																
5	Ціле число називається простим, якщо воно ділиться на 1 і на самого себе. Наприклад, числа 2,3,5 і 7 є простими, а 2,6,8 і 9 – ні. Напишіть функцію, яка визначає, чи є число простим. Використайте цю функцію у програмі, яка знаходить і друкує усі прості числа у діапазоні від 1 до 10000.																
6	Напишіть функцію, що отримує ціле значення і повертає число з оберненим порядком цифр. Наприклад, для 7631 функція повинна повернути 1367.																
7	Напишіть функцію, яка вводить бали, зароблені студентом, а повертає оцінку за шкалою ECTS: <table><tr><td>Сума балів</td><td>95-100</td><td>85-94</td><td>75-84</td><td>65-74</td><td>64-60</td><td>30-60</td><td>0-30</td></tr><tr><td>Оцінка ECTS</td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>FX</td><td>F</td></tr></table>	Сума балів	95-100	85-94	75-84	65-74	64-60	30-60	0-30	Оцінка ECTS	A	B	C	D	E	FX	F
Сума балів	95-100	85-94	75-84	65-74	64-60	30-60	0-30										
Оцінка ECTS	A	B	C	D	E	FX	F										
8	Напишіть функцію distance, яка обраховує відстань між двома точками з координатами (x1, y1) і (x2, y2). Усі числа і повернене значення повинні бути дійсними. Використайте цю функцію у програмі, яка обчислює площу трикутника за формулою Герона.																
9	Напишіть програму, яка допоможе школяру вивчити таблицю множення. Згенеруйте два додатних однорозрядних числа (окрема функція). Програма виводить, наприклад, питання: "Скільки буде 5 на 6?". Школяр повинен відповісти. Якщо відповідь правильна, програма друкує "Молодець! Дуже добре!" і після цього задає наступне питання на множення. Якщо відповідь неправильна, програма друкує: "Невірно! Спробуйте знову ...", до тих пір, поки відповідь не буде правильною.																

№	Завдання
	Розроблювана функція повинна отримувати три числа: два множники та число-віповідь школяра, і друкувати потрібну фразу.
10	Напишіть функцію <code>multiply</code> для двох цілих, яка визначатиме, чи кратне друге число першому. Функція повинна отримувати два цілих аргументи і повертати 1 (<code>true</code>), якщо друге число кратне першому, і 0 (<code>false</code>) в іншому випадку. Використайте цю функцію у програмі, яка вводить серію пар цілих чисел.
11	Напишіть програму, яка грає у гру "Вгадай число" наступним чином: ваша програма "задумає" число (випадкове число у діапазоні від 1 до 1000), яке треба вгадати. Далі програма друкує: <i>У мене є число між 1 та 1000. Відгадайте і введіть ваше число...</i> Далі гравець вводить перше число. Програма відповідає однією з фраз: <i>Чудово! Ви вгадали число! Будете грати далі?</i> <i>Занадто мале. Спробуйте ще раз.</i> <i>Занадто велике. Спробуйте ще раз.</i> При реалізації гри необхідно написати функцію, яка приймає два числа: "задумане" і відповідь гравця, а після аналізу друкувати одну з фраз.
12	Написати функцію, яка виводить на екран трикутник, створений із заданих символів <code>C</code> (наприклад, <code>#</code>), основа трикутника визначається змінною <code>W</code>. Наприклад, якщо <code>C='#'</code>, <code>W=5</code>, то функція виведе показане зображення. <pre>***** **** *** ** *</pre>
13	Напишіть програму, яка буде використовувати функцію, що обчислює скалярний добуток двох векторів дійсних чисел, що мають однакову розмірність.
14	Розробити функцію, яка змінює значення двох заданих дійсних змінних. Перша змінна повинна отримати значення суми початкових даних, а друга – значення їх різниці.
15	Напишіть програму, яка вводить довільні натуральні числа і визначає, чи є введене число паліндромом. Для визначення, чи є число паліндромом, написати окрему функцію, яка приймає число і повертає <code>true</code> або <code>false</code> . Число називається паліндромом, якщо воно сліва направо і справа наліво читається однаково. Наприклад, числа 232, 7667 і 34543 є паліндромами, а числа 123 і 45 – ні.
16	Напишіть програму, яка отримує рядок і виводить його посимвольно, вставляючи між символами знаки підкреслювання.
17	Введіть з клавіатури 6 чисел і визначте їх середнє арифметичне.
18	Написати програму, яка обчислює суму і середнє арифметичне послідовності додатних чисел, які вводяться з клавіатури.
19	Написати програму, яка генерує десять випадкових чисел в діапазоні від 1

№	Завдання
	до 10, виводить ці числа на екран і обчислює їх середнє арифметичне.
20	Написати програму, яка вводить з клавіатури послідовність з п'яти дробових чисел і після введення кожного числа виводить середнє арифметичне одержаної частини послідовності.
21	У масиві всі елементи, які стоять після непарних, замінити на 0. Приклад: з масиву A[5]: 1 3 4 5 6 повинен вийти масив 1 0 4 5 0.
22	У масиві всі парні елементи обнулити. Приклад: з масиву A[5]: 1 3 4 5 6 повинен вийти масив 1 3 0 5 0.
23	У масиві всі елементи, які стоять після мінімального, замінити на 0. Приклад: з масиву A[5]: 3 2 1 5 6 повинен вийти масив 3 2 1 0 0.
24	У масиві всі непарні елементи, які стоять після максимального, замінити на 0. Приклад: з масиву A[5]: 3 7 1 5 4 повинен вийти масив 3 7 0 0 4.
25	Обчислити добуток елементів масиву B[1..10] з непарними коефіцієнтами.
26	Сформувати масив C, елементами якого є сума відповідних елементів масивів A і B.
27	Сформувати масив C, елементами якого є ціла частина від ділення відповідних елементів масивів A і B.
28	З масиву видалити останній з непарних елементів. Приклад: з масиву A [5]: 1 3 4 5 6 повинен вийти масив A [4] 1 3 4 6.

Теоретичні відомості

ФУНКЦІЇ

Функція в Python є основою при написанні програм. До сих пір ми використовували вбудовані функції і функції з модулів, які входять в комплект його постачання. Але міць структурних мов програмування полягає в тому, що ми можемо створювати свої власні функції, причому робиться це досить просто. У структурному програмуванні функція являє собою іменовану послідовність виразів, що виконують необхідну операцію.

Розглянемо функцію з ім'ям **abs ()**, вона приймає на вхід один аргумент – об'єкт числового типу і повертає абсолютне значення для цього об'єкта.

Приклад виклику функції **abs ()** з аргументом -9 має вигляд:

```
>>> abs (-9)
9
>>> d = 1
>>> n = 3
>>> abs (d - n)
2
>>> abs (-9) + abs (5.6)
14.6
```

Результат виклику функції можна присвоїти змінній, використовувати його в якості операндів математичних виразів, тобто складати більш складні вирази.

pow(x, y) повертає значення x в степені y . Еквівалентно запису $x^{**}y$.

```
>>> pow(4,5)
```

1024

round(number) повертає число з плаваючою точкою, округлене до 0 цифр після коми (за замовчуванням). Може бути викликана з двома аргументами:

round (number [, ndigits]), де **ndigits** - число знаків після коми.

```
>>> round(4.56666)
```

```
5
```

```
>>> round(4.56666, 3)
```

```
4.567
```

```
>>>
```

Крім складання складних математичних виразів Python дозволяє передавати результати виконання функцій в якості аргументів інших функцій без використання додаткових змінних:

```
pow (abs (-2), round(4.3))
```

На малюнку представлений приклад виклику і порядок обчислення виразів. У цьому прикладі на місці числових об'єктів (-2, 4.3) можуть перебувати виклики функцій або їх комбінації, тому вони теж обчислюються.

Функція **int ()** повертає цілочисельний об'єкт, побудований з числа або рядка, або 0, якщо аргументи не передані.

Функція **float ()** повертає число з плаваючою точкою, побудоване з числа або рядка.

```
>>> int(5.6)
```

```
5
```

```
>>> int()
```

```
0
```

```
>>> float(5)
```

```
5.0
```

```
>>> float()
```

```
0.0
```

```
>>>
```

Завдання. Знайдіть значення наступних виразів:

```
pow (abs(-5) + abs(-3), round(5.8))
```

```
int (round (pow(round(5.777, 2), abs(-2)), 1))
```

Функція в Python - об'єкт, який приймає аргументи і повертає значення. Оголошення функції описує її прототип (іноді кажуть "сигнатура"). Зазвичай функція визначається за допомогою інструкції **def**, що має такий синтаксис:

def ІМ'Я_ФУНКЦІЇ(СПИСОК_ПАРАМЕТРІВ):

ПОСЛІДОВНІСТЬ_ВИРАЗІВ

Правила вибору імен функцій повністю аналогічні правилам вибору імен змінних. *Список параметрів* визначає набір значень, які можуть бути передані функції в якості вихідних даних - параметри перераховуються через кому. Перший рядок визначення зазвичай називається заголовком функції, позначених ключовим словом **def** (від англ. «Define» - «визначити»). Заголовок

функції в Python завершується двокрапкою. Після нього може слідувати будь-яка кількість виразів, але вони повинні бути записані зі зміщенням щодо початку рядка.

```
>>> def printAnything(object):
...     print (object)
>>> printAnything("Some string")
Some string
>>> number = 15
>>> printAnything(number)
15
```

У перших двох рядках прикладу ми визначили функцію PrintAnything (), яка друкує об'єкт, переданий як параметр. Ми можемо передати цю функцію рядок або число, що ми і спробували зробити в четвертому і сьомому рядках прикладу.

ПАРАМЕТРИ І АРГУМЕНТИ ФУНКЦІЙ

Часто функція використовується для обробки даних, отриманих із зовнішнього для неї середовища (з основної програми). Дані передаються функції при її виклику в дужках і називаються *аргументами*. Однак, щоб функція могла "взяти" передані їй дані, необхідно при її створенні описати *параметри* (в дужках після імені функції), що представляють собою змінні.

Параметрами функції називають список змінних, яким присвоюються при виклику цієї функції значення, що передаються. А самі передані значення називають *аргументами*. У своєму тілі функція оперує параметрами, але не аргументами.

Коли функція викликається, конкретні аргументи підставляються замість параметрів-змінних. У більшості випадків кількість аргументів і параметрів має збігатися (хоча можна запрограмувати змінну кількість прийнятих аргументів). В якості аргументів можуть виступати як безпосередньо значення, так і змінні, що посилаються на них.

```
>>> def printAnything(object):
...     print object
>>> printAnything("Some string")
Some string
>>> number = 15
>>> printAnything(number)
15
```

Тут змінна *object* є параметром, а рядок "Some string" і змінна *number* - аргументами. Таким чином, правильно буде сказати «передати аргумент» або «передати значення як параметр», але не «передати функції параметр».

Локальні і глобальні змінні

Якщо записати в IDLE наведену нижче функцію, і потім спробувати вивести значення змінних, то виявиться, що деяких з них чомусь не існує:

```
>>> def mathem(a,b):
...     a = a/2
...     b = b+10
```

```

    print(a+b)
>>> num1 = 100
>>> num2 = 12
>>> mathem(num1,num2)
72.0
>>> num1
100
>>> num2
12
>>> a
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    a
NameError: name 'a' is not defined
>>> b
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    b
NameError: name 'b' is not defined

```

Змінні *num1* і *num2* не змінили своїх початкових значень. У функцію передаються копії значень. Колишні значення з основної гілки програми залишилися як і раніше пов'язані з їх змінними.

А ось змінних *a* і *b* виявляється немає (помилка "name 'b' is not defined" перекладається як "змінна *b* не визначена"). Ці змінні існують лише в момент виконання функції і називаються **локальними**. На противагу їм, змінні *num1* і *num2* видно не тільки в зовнішній гілці, але і всередині функції:

```

>>> def mathem2():
    print(num1+num2)
>>> mathem2()
112

```

Змінні, визначені в основний гілці програми, є **глобальними**.

Функції, які повертають результат

Неважко уявити собі ситуацію, коли результати роботи функції можуть знадобитися для подальшої роботи програми. Але після завершення роботи функції всі змінні, які були ініційовані в її тілі, знищуються, тому що вони є локальними. Тому при необхідності **повернути результат роботи функції** в підпрограму, з якої вона викликала, для його подальшої обробки застосовується команда *return*. Виглядає це приблизно так:

```

>>> def getSum(x,y):
...     z = x + y
...     return z
>>> print getSum(1,3)
4

```

Отже, значення, що повертається функцією, сприймається інтерпретатором, як результат виразу, що викликає цю функцію.

У Python функції здатні повертати кілька значень одночасно. Для прикладу візьмемо функцію обчислення коренів квадратного рівняння:

```
>>> def PrintRoots(a, b, c):
...     D = b**2 - 4 * a * c
...     import math
...     x1 = (-b + math.sqrt(D)) / 2 * a
...     x2 = (-b - math.sqrt(D)) / 2 * a
...     return x1, x2
>>> print PrintRoots(1.0, 0, -1.0)
(1.0, -1.0)
```

Крім того, результати виконання функції можна привласнювати відразу декільком змінним:

```
>>> x1, x2 = PrintRoots(1.0, 0, -1.0)
>>> print "x1 =", x1, "\nx2 =", x2
x1 = 1.0
x2 = -1.0
```

Функція може бути будь-якої складності і повертати будь-які об'єкти (списки, кортежі, і навіть функції):

```
>>> def newfunc(n):
...     def myfunc(x):
...         return x + n
...     return myfunc
>>> new = newfunc(100) # new - це функція
>>> new(200)
300
```

РЯДКИ ДОКУМЕНТАЦІЇ

Для опису роботи функцій використовують документацію. В Python документація для функції може бути викликана за допомогою функції **help ()**, на вхід якої передається ім'я функції:

```
>>> help(abs)
Help on built-in function abs in module builtins:
abs(x, /)
    Return the absolute value of the argument.
>>>
```

Напишіть власну функцію, яка нічого не буде робити (в тілі функції для цього вказується слово **pass**), але яка виведе інформацію, що вона нічого не робить.

```
В окремому файлі наберіть і запустіть:
def my_function():
    """Не робимо нічого, але документуємо.
    Ця функція нічого не робить.
    """
    pass
help (my_function)
```

В `"""` потрібні подвійні лапки в тілі функції поміщається інформація, яку виводить на екран функція **help ()**.

МОДУЛІ У PYTHON

Наприклад, ви написали кілька корисних функцій, які часто використовуєте в своїх програмах. Щоб до них швидко звертатися, зручно всі ці функції помістити в окремий файл і завантажувати їх звідти. В Python такі файли з набором функцій називаються *модулями*. Для того щоб скористатися функціями, які знаходяться в цьому модулі, його необхідно імпортувати за допомогою команди **import**:

```
>>> import math
>>>
```

Ми завантажили в пам'ять стандартний модуль **math** (містить набір математичних функцій), тепер можна звертатися до функцій всередині цього модуля.

Сила Python у величезній кількості стандартних і корисних модулів. Звернутися до функції модуля (в даному випадку для знаходження квадратного кореня з 9) можна наступним чином:

```
>>> math.sqrt(9)
3.0
>>>
```

Вказується ім'я модуля, точка та ім'я функції з аргументами. Дізнатися про функції, які містить модуль, можна через довідку:

```
>>> help(math)
Help on built-in module math:
NAME
    math
DESCRIPTION
    This module is always available. It provides access to the
    mathematical functions defined by the C standard.
FUNCTIONS
    acos(...)
        acos(x)
```

Return the arc cosine (measured in radians) of x.

```
>>>
```

Якщо хочемо подивитися опис конкретної функції модуля, то викликаємо довідку окремо для неї:

```
>>> help(math.sqrt)
Help on built-in function sqrt in module math:
sqrt(...)
    sqrt(x)
```

Return the square root of x.

```
>>>
```

У момент виклику функції ***sqrt()*** Python знаходить змінну ***math*** (модуль повинен бути попередньо імпортований), переглядає модульний об'єкт, знаходить функцію ***sqrt()*** всередині цього модуля і потім виконує її.

В Python можна імпортувати окрему функцію з модуля:

```
>>> from math import sqrt
>>> sqrt(9)
3.0
>>>
```

Таким чином, Python не створюватиме змінну ***math***, а завантажить в пам'ять тільки функцію ***sqrt()***. Тепер виклик функції можна виконувати, не звертаючись до імені модуля ***math***.

```
>>> def sqrt(x):
    return x*x
>>> sqrt(5)
25
>>> from math import sqrt
>>> sqrt(9)
3.0
>>>
```

Було створено власну функцію з ім'ям ***sqrt***. Після цього імпортовано функцію ***sqrt()*** з модуля ***math***. Викликавши ***sqrt()*** бачимо, що це не наша функція.

Інший приклад:

```
>>> def sqrt(x):
    return x*x
>>> sqrt(6)
36
>>> import math
>>> math.sqrt(9)
3.0
>>> sqrt(7)
49
>>>
```

Знову створюємо власну функцію з ім'ям ***sqrt*** і викликаємо її. Імпортуємо модуль ***math*** і через нього викликаємо стандартну функцію ***sqrt()***. Тепер корінь квадратний вираховується і функція залишилася як повинно бути.

Функція ***abs()*** використовується для знаходження модуля числа. Насправді, ця функція теж знаходиться в модулі, який Python завантажує в пам'ять в момент початку роботи. Цей модуль називається ***__builtins__*** (два нижніх підкреслення до і після імені модуля).

Якщо викликати довідку для даного модуля, то там є величезна кількість функцій і змінних:

```
>>> help (__builtins__)
Help on built-in module builtins:
NAME
```

builtins - Built-in functions, exceptions, and other objects.

DESCRIPTION

Noteworthy: None is the 'nil' object; Ellipsis represents '...' in slices.

...

>>>

В Python є функція **dir()**, яка повертає перелік імен всіх функцій і змінних, що містяться в модулі:

```
>>> dir (__builtins__)
```

...

>>>

СТВОРЕННЯ ВЛАСНИХ МОДУЛІВ

Створіть файл з ім'ям **mm.py** (для модулів обов'язково вказується розширення **.py**), що містить код (вміст модуля):

```
def f():
```

```
    return 4
```

Тепер потрібно сказати Python, де шукати модуль. З'ясуємо через звернення до змінної **path** модуля **sys**, де Python за замовчуванням зберігає власні модулі (список каталогів може відрізнятись):

```
>>> import sys
```

```
>>> sys.path
```

```
['', 'C:\\Python35-32\\Lib\\idlelib', 'C:\\Python35-32\\python35.zip', 'C:\\Python35-32\\DLLs', 'C:\\Python35-32\\lib', 'C:\\Python35-32', 'C:\\Python35-32\\lib\\site-packages']
```

>>>

Далі помістимо модуль в один з перерахованих каталогів, наприклад, в 'C:\\Python35-32'.

Якщо все правильно зроблено, то імпортуємо модуль, вказавши тільки його ім'я (без розширення):

```
>>> import mm
```

```
>>> mm.f()
```

```
4
```

>>>

Тепер через крапку можемо викликати функцію, яка знаходиться в модулі **mm**.

Створимо ще один модуль (по аналогії з попереднім), вкажемо для нього інше ім'я - **mtest.py**:

```
print('test')
```

Новий модуль буде містити виклик функції **print()**. Імпортуємо його кілька разів поспіль:

```
>>> import mtest
```

```
test
```

```
>>> import mtest
```

```
>>>
```

Отже, імпортування модуля виконує команди що містяться в ньому. А повторне імпортування не приводить до виконання модуля, тобто він повторно не імпортується. Пояснюється це тим, що імпортування модулів в пам'ять – ресурсномісткий процес, тому зайвий раз Python його не виконує цього. Але якщо було змінено модуль і необхідно імпортувати його повторно:

```
>>> import imp
>>> imp.reload(mtest)
test
<module 'mtest' from 'C:\\Python35-32\\mtest.py'>
>>>
```

Отже примусово вказали Python, що модуль вимагає перезавантаження. Після виклику функції **reload()** із зазначенням в якості аргументу імені модуля, оновлений модуль завантажиться повторно.

Створити ще один модуль з ім'ям **mypr.py**:

```
def func(x):
    return x**2+7
x=int(input("Введіть значення: "))
print(func(x))
Імпорт модуля призводить до виконання всієї програми:
>>> import mypr
Введіть значення: 111
12328
>>>
```

При необхідності тільки імпортувати функцію **func()** з модуля для використання її в іншій програмі, для того щоб відокремити виконання модуля (Run -> Run Module) від його імпортування (import mypr) в Python є спеціальна змінна **__name__** (Python починає назви спеціальних функцій і змінних з двох нижніх підкреслень):

Якщо запускаємо модуль, то вміст змінної **__name__** дорівнюватиме рядку **__main__**, а в разі імпортування - змінна **__name__** буде містити ім'я модуля.

Створити модуль з ім'ям **prog3.py** і змістом:

```
def func(x):
    return x**2+7
if __name__ == "__main__":
    x=int(input("Введіть значення: "))
    print(func(x))
```

Тепер Python зрозуміє, коли необхідно виконати модуль, а коли - імпортувати. Якщо модуль виконати (Run -> Run Module), то виконається весь файл, тому що спрацює умова **if**. При імпортуванні модуля (import prog3) умова не виконається і Python завантажить в пам'ять тільки функцію **func()**.

Контрольні запитання:

1. Що називають функцією?
2. Що таке прототип функції? Де його розміщують у програмі?

3. Як відбувається звернення до функції?
4. Чи кожна функція повинна мати оператор повернення?
5. Що таке локальні змінні?
6. Що таке глобальні змінні?
7. Що таке фактичні параметри функції?
8. Що таке формальні параметри?
9. Чи можуть ідентифікатори фактичних і формальних параметрів співпадати?
10. Чи обов'язково кількість фактичних і формальних параметрів повинні співпадати?
11. Чи може глобальна змінна бути розташована у тілі програми?
12. Чи можна у середині однієї функції оголошувати іншу функцію?
13. Що таке вбудовані функції? Коли вони використовуються?
14. Що таке документаційні рядки?

Додаток А. Оформлення звіту за результатами виконання комп'ютерного практикуму

Звіт по виконанню комп'ютерного практикуму оформляється у вигляді звітнього документу.

Вимоги до звітнього документу:

- усі поля параметрів сторінки – 2 см;
- верхній та нижній колонтитули – 0 см;
- текст матеріалів набирається шрифтом Times New Roman, кегль 12, міжрядковий інтервал 1,0; абзацний відступ – 1 см; вирівнювання тексту за шириною сторінки.

Звіт по комп'ютерному практикуму (Приклад 1) формується на основі протоколу, який ведеться під час виконання поточної роботи та результатів домашньої (теоретичної та практичної) підготовки.

Звіт повинен містити наступні розділи:

1. Вступна частина (назва: ВУЗу, факультету, кафедри, дисципліни; номером поточної лабораторної роботи та тема; номер групи та П.І.Б. виконавця; П.І.Б. викладача, що перевірятиме роботу);

2. Основна частина:

- мета роботи;
- завдання до роботи;
- лістинг програми та результати поставленого завдання у вигляді скріншоту виконання коду та блок-схеми;

3. Відповіді на контрольні запитання.

Факультет Біомедичної інженерії
Національного технічного університету України «КПІ ім. І. Сікорського»
Кафедра біомедичної кібернетики

Дисципліна «Основи програмування»

Комп'ютерний практикум № ____

Тема: _____

Виконав (ла):
студент (ка) групи ____,
(ПІБ) _____

Перевірив:
викладач (ПІБ) _____

дата _____ підпис _____

Мета роботи: _____.

Завдання до роботи: _____.

Лістинг програми та результати

Відповіді на контрольні запитання

Література

Базова

1. Лутц М. Изучаем Python, 4-е издание – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с, ил.
2. Доусон М. Програмуємо на Python. – СПб.: Питер, 2014. – 416 с.: ил.
3. Мусин Д. Самоучитель Python. Выпуск 0.2, 2015. – 136 с.

Допоміжна

4. Дональд Кнут Искусство программирования, том 1. Основные алгоритмы – The Art of Computer Programming, vol.1. Fundamental Algorithms. – 3-е изд. – М.: «Вильямс», 2006. – С. 720. – ISBN 0-201-89683-4