

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Спеціальні розділи програмування Лабораторний практикум

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітніми програмами «Системи, технології та математичні методи
кібербезпеки» та «Системи технічного захисту інформації»
спеціальності 125 «Кібербезпека»*

Київ
КПІ ім. Ігоря Сікорського
2021

Спеціальні розділи програмування. Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 125 «Кибербезпека» / А. Ю. Шелестов, Н. М. Куcssуль; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1086 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 31 с.

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 4 від 10.12.2020 р.)
за поданням Вченої ради Фізико-технічного інституту Національного технічного університету
України «Київський політехнічний інститут імені Ігоря Сікорського» (протокол № 11 від
26.11.2020 р.)*

Електронне мережне навчальне видання

Спеціальні розділи програмування Лабораторний практикум

Укладачі:

*Шелестов Андрій Юрійович, д. техн. наук, проф.
Куcssуль Наталія Миколаївна, д. техн. наук, проф.*

Відповідальний
редактор

Смирнов С.А., к.ф.-м.н., доц.

Рецензент

*Лавренюк А.М., канд. техн. наук, доц. кафедри інформаційної
безпеки ФТІ, Національного технічного університету України
"Київський політехнічний інститут імені Ігоря Сікорського"*

Навчальний посібник «Спеціальні розділи програмування. Лабораторний практикум» є базовим курсом для вивчення спеціальних розділів програмування та сучасних інформаційних технологій за спеціальністю 125 «Кибербезпека». Метою є оволодіння сучасними підходами та технологіями аналізу та обробки даних при розв’язанні складних науково-прикладних задач у багатьох предметних галузях. Посібник містить необхідний теоретичний матеріал, приклади програм, а також завдання для виконання лабораторного практикуму.

© КПІ ім. Ігоря Сікорського, 2021

ЗМІСТ

| | |
|--|-----------|
| ЛАБОРАТОРНА РОБОТА №1 НАУКА ПРО ДАНІ: ПІДГОТОВЧИЙ ЕТАП | 4 |
| 1.1. ТЕОРЕТИЧНІ ВІДОМОСТІ..... | 4 |
| <i>Підготовка до роботи з даними. Налаштування робочого середовища..</i> | <i>4</i> |
| <i>Робота з пакетним менеджером PIP (основи).....</i> | <i>6</i> |
| <i>Робота з Anaconda.....</i> | <i>7</i> |
| 1.2. ПРИКЛАД | 8 |
| 1.3. ПОРЯДОК ВИКОНАННЯ РОБОТИ..... | 10 |
| 1.4. КОНТРОЛЬНІ ЗАПИТАННЯ | 11 |
| 1.5. ДОДАТКОВІ ДЖЕРЕЛА ІНФОРМАЦІЇ | 12 |
| ЛАБОРАТОРНА РОБОТА №2 НАУКА ПРО ДАНІ: ОБМІН РЕЗУЛЬТАТАМИ ТА ПОЧАТКОВИЙ АНАЛІЗ | 13 |
| 2.1. ТЕОРЕТИЧНІ ВІДОМОСТІ..... | 13 |
| <i>Вступ до роботи з GitHub</i> | <i>13</i> |
| <i>Git vs GitHub</i> | <i>13</i> |
| 2.2. ПРИКЛАД | 15 |
| <i>Демонстрація результатів аналізу у Web.....</i> | <i>15</i> |
| 2.3. ПОРЯДОК ВИКОНАННЯ РОБОТИ..... | 16 |
| 2.4. КОНТРОЛЬНІ ЗАПИТАННЯ | 16 |
| 2.5. ДОДАТКОВІ ДЖЕРЕЛА ІНФОРМАЦІЇ | 16 |
| ЛАБОРАТОРНА РОБОТА №3 СТРУКТУРИ ДЛЯ РОБОТИ З ВЕЛИКИМИ ОБСЯГАМИ ДАНИХ В PYTHON | 17 |
| 3.1. ТЕОРЕТИЧНІ ВІДОМОСТІ..... | 17 |
| 3.2. ПРИКЛАД | 17 |
| 3.3. ПОРЯДОК ВИКОНАННЯ РОБОТИ..... | 19 |
| 3.4. ЗАВДАННЯ..... | 19 |
| 3.5. ДОДАТКОВІ ДЖЕРЕЛА ІНФОРМАЦІЇ | 20 |
| ЛАБОРАТОРНА РОБОТА № 4 СЦЕНАРІЇ ОБРОБКИ БАГАТОСПЕКТРАЛЬНИХ СУПУТНИКОВИХ ЗОБРАЖЕНЬ..... | 21 |
| 4.1. ТЕОРЕТИЧНІ ВІДОМОСТІ..... | 21 |
| <i>Налаштування робочого середовища</i> | <i>21</i> |
| <i>Коротко про супутникові зображення (довідкова інформація)</i> | <i>23</i> |
| <i>Робота з растровими даними через gdal.....</i> | <i>26</i> |
| 4.2. ПРИКЛАД | 28 |
| 4.3. ПОРЯДОК ВИКОНАННЯ РОБОТИ..... | 30 |
| 4.4. КОНТРОЛЬНІ ЗАПИТАННЯ | 30 |
| 4.5. ДОДАТКОВІ ДЖЕРЕЛА ІНФОРМАЦІЇ | 31 |

Лабораторна робота №1

Наука про дані: підготовчий етап

Мета роботи: ознайомитися з основними кроками по роботі з даними – workflow від постановки задачі до написання пояснювальної записки, зрозуміти постановку задачі та природу даних, над якими виконується аналітичні операції.

1.1. Теоретичні відомості

Наука про дані. З часом обсяг даних стає все більшим (Big Data), а тому все відчутнішою стає потреба в спеціалістам в області роботи з даними (Data Science Specialization). В межах робіт в цьому напрямку фахівцями виконуються роботи з пошуку даних, їх попередньої підготовки, фільтрування, початкового аналізу, виявлення закономірностей та пояснення отриманих результатів.

Для успішного виконання робіт в цій області потрібно не тільки володіти відповідним математичним апаратом, розуміти семантику області, в якій виконується дослідження, але й володіти ефективними засобами для автоматизації процесу аналізу даних. Таким засобом автоматизації може виступати мова програмування, яка підтримує роботу з численними типами даних, відзначається широким спектром бібліотек для розширення функціоналу та відповідає сучасним вимогам до мов програмування.

Чому ж для вирішення цього кола задач доволі часто використовується мова Python? Вона є досить простою та зрозумілою, не залежить від платформи (існують інтерпретатори під більшість сучасним платформ), є мовою загального призначення (на відміну від Matlab чи R). До того ж ця мова має потужне товариство користувачів та все зростаючу популярність.

Підготовка до роботи з даними. Налаштування робочого середовища

Щодо складу та структури середовища Python досить взяти до уваги наступне:

Вбудовані класи та функції (`__builtins__`) не потребують підключення відповідних модулів (`import`);

Вбудовані модулі (близько 40) – додаткові класи та функції (не доступні без `import`);

Python Standard Library, PSL (понад 200 модулів) – при інсталяції Python за замовчуванням додається набір модулів із класами та функціями (не потребують встановлення), які можна імпортувати в робоче середовище – може сягати;

Python Package Index, PyPI (понад 42000 станом на 2014) — популярні пакети обробки даних, framework (django) – потребують встановлення.

Порада: для виконання лабораторних робіт Ви можете використовувати будь-яке середовище для роботи на Python, однак досить корисним було б ознайомитись з [IPython notebook](#), який встановлюється разом із IDE Spyder в дистрибутиві Anaconda.

Чимало популярних пакетів (`numpy`, `scipy`, `pandas`, `matplotlib`), які часто використовуються для роботи з даними, належать саме до останньої групи (Python Package Index), а тому буде корисним (особливо для тих, хто не зміг присвятити достатньо часу курсовій роботі в минулому семестрі ☺) розглянути процедуру встановлення таких додаткових пакетів, а також самого інтерпретатора.

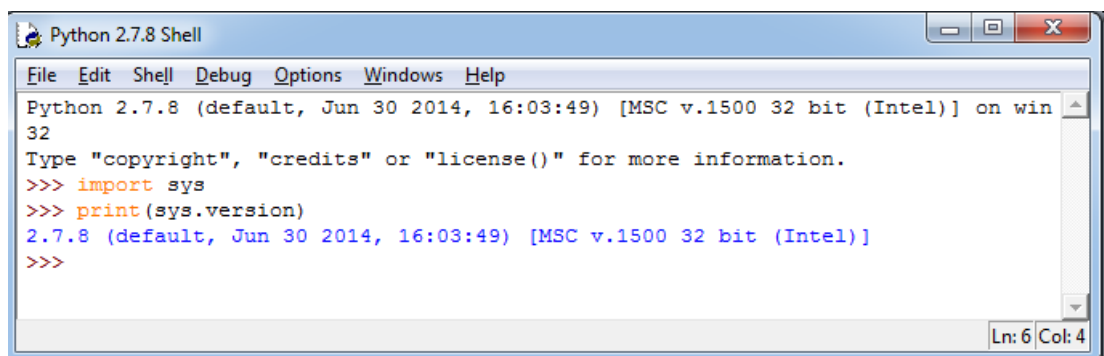
Для початку розглянемо налаштування робочого середовища на прикладі ОС Windows 7 x64. Потрібно встановити коректну версію інтерпретатора для своєї системи (ті, хто віддає перевагу Linux-системам, можуть отримати консультацію щодо встановлення у викладача, який веде практичні заняття, та у одногрупників ☺).

На даний момент має місце перехідний період від Python 2 до Python 3 (триватиме до 2020 р.), а тому будемо розглядати роботу з Python 3, який вже має достатню кількість розробників. На момент написання методичних вказівок актуальною була версія 3.7.6, MSI-пакет <https://www.python.org/downloads/>, а також для новачків рекомендовано використовувати дистриб'ютор анаконда, який можна встановити з Python актуальної версії за адресою <https://www.anaconda.com/distribution/>.

Завантажуємо Windows x86-64 MSI installer та встановлюємо (за замовчуванням - C:\Python37). Після завершення процедури перевіримо коректність встановлення та версію – запустимо Python GUI IDLE (даний shell встановлюється автоматично разом з інтерпретатором та базовими бібліотеками):

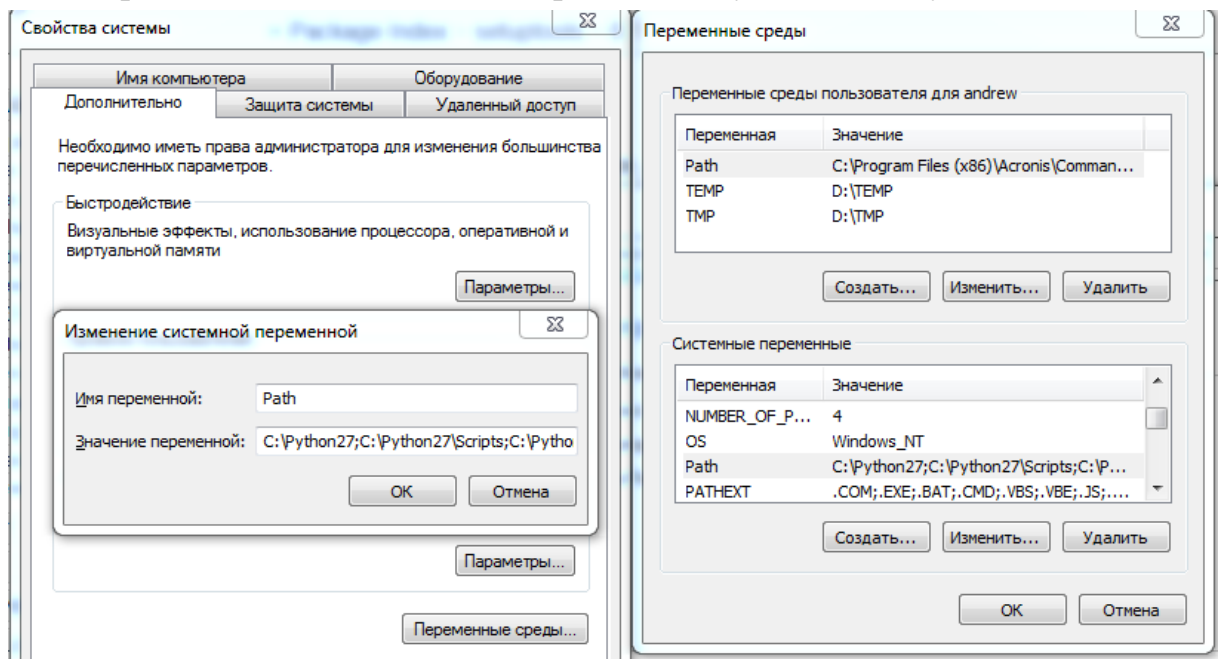
```
>>> import sys
```

```
>>> print(sys.version)
```



Для того, щоб в ОС Windows використовувати скрипти Python, в тому числі для простого встановлення бібліотек з командного рядка, необхідно визначити системні зміни, це можна зробити у наступний спосіб:

- Комп'ютер - властивості;
- Додаткові параметри системи;
- В розділі системні змінні знайти змінну PATH та додати до переліку шляхів наступне значення: C:\Python37;C:\Python37\Scripts; (за умови, що встановили Python за замовчуванням);
- Зберігаємо внесені зміни та перезавантажуємо систему.



Робота з пакетним менеджером PIP (основи)

Для того, щоб отримати довідку по роботі з Python, введіть команду:

```
pip help
```

Для встановлення наявних пакетів:

```
pip install <ім'я_пакету >
```

Для видалення пакету:

```
pip uninstall <ім'я_пакету >
```

Для пошуку пакету:

```
pip search <пошуковий_параметр>
```

Під час встановлення пакетів може знадобитися встановлення додатково програмного забезпечення (компілятор Fortran95, C та C++ тощо) – уважно читайте інструкції та помилки, які видає pip.

Для того, щоб встановити деякі з приведених нижче модулів, потрібно скачати і встановити Microsoft Visual C++ 9.0 :

<http://aka.ms/vcpython27>

Встановимо модулі matplotlib, numpy, pandas, ipython:

```
pip install numpy pandas matplotlib ipython
```

Також варто зникати до того, що при потребі Ви будете встановлювати додаткові модулі для виконання тих чи інших специфічних задач.

Робота з Anaconda

Із офіційного сайту завантажте файл для встановлення Anaconda з необхідною версією python. При встановленні на Windows в почергових стандартних виборах опцій виберіть встановлення для одного юзера, що є рекомендованим, а також додавання до PATH.

Використовуючи дистриб'ютор Anaconda можливо в командному рядку проводити схожі операції з бібліотеками, як і з PIP, проте вагомою перевагою є можливість створення віртуальних середовищ (virtual environment) який далі будемо називати env. Всередині кожного активного env можливо встановити свій список бібліотек, не переживаючи за функціональність інших env. Таким чином якщо при встановленні бібліотек ви зіпсували зв'язки і нічого більше не працює, можна створити новий env і почати заново його псувати. В таблиці наведено порівняння основних команд командного рядку/терміналу conda та pip. Для написання програмного коду в jupyter notebook необхідно в командному рядку ввести jupyter notebook і створити ipython notebook в інтерфейсі браузера.

| Task | Conda package and environment manager command | Pip package manager command |
|------------------------|--|---|
| Install a package | <code>conda install \$PACKAGE_NAME</code> | <code>pip install \$PACKAGE_NAME</code> |
| Update a package | <code>conda update --name \$ENVIRONMENT_NAME \$PACKAGE_NAME</code> | <code>pip install --upgrade \$PACKAGE_NAME</code> |
| Update package manager | <code>conda update conda</code> | Linux/macOS: <code>pip install -U pip</code> Win: <code>python -m pip install -U pip</code> |
| Uninstall a package | <code>conda remove --name \$ENVIRONMENT_NAME \$PACKAGE_NAME</code> | <code>pip uninstall \$PACKAGE_NAME</code> |

| | | |
|--------------------------------------|--|---|
| Create an environment | <code>conda create --name \$ENVIRONMENT_NAME python</code> | X |
| Activate an environment | <code>conda activate \$ENVIRONMENT_NAME</code> | X |
| Deactivate an environment | <code>conda deactivate</code> | X |
| Search available packages | <code>conda search \$SEARCH_TERM</code> | <code>pip search \$SEARCH_TERM</code> |
| Install package from specific source | <code>conda install --channel \$URL \$PACKAGE_NAME</code> | <code>pip install --index-url \$URL \$PACKAGE_NAME</code> |
| List installed packages | <code>conda list --name \$ENVIRONMENT_NAME</code> | <code>pip list</code> |
| Create requirements file | <code>conda list --export</code> | <code>pip freeze</code> |
| List all environments | <code>conda info --envs</code> | X |
| Install other package manager | <code>conda install pip</code> | <code>pip install conda</code> |
| Install Python | <code>conda install python=x.x</code> | X |
| Update Python | <code>conda update python</code> | X |

1.2. Приклад

Проаналізувати часові ряди глобальних продуктів по оцінці вегетаційного здоров'я VHI (vegetation health index) <http://www.star.nesdis.noaa.gov/smcd/emb/vci/VH/index.php>, який надається Національною адміністрацією океанів та атмосфери США NOAA (<http://www.noaa.gov/>);

Виявити особливості ходу індексу впродовж вегетаційного періоду (вересень року-попередника – липень поточного року) в розрізі областей України;

Додаткові завдання по фільтрації даних від викладача, який веде практику.

VHI – вегетаційний індекс, який базується на відбитті видимого світла рослинним покривом, яке характеризує ступінь здоров'я рослинності. Цей індекс базується на поєднанні індексу VCI (Vegetation Condition Index), який описує ступінь пригніченості рослинного покриву та індексі температурного режиму TCI (Temperature Condition Index), які були запропоновані в 1995 році Ф. Коганом:

$$VHI = 0.5 * VCI + 0.5 * TCI,$$

$$VCI = 100 * \frac{(NDVI - NDVI_{\min})}{(NDVI_{\max} - NDVI_{\min})},$$

$$TCI = 100 * (BT_{\max} - BT) / (BT_{\max} - BT_{\min}),$$

де BT , BT_{\max} , BT_{\min} - усереднені сезонні значення яскравосної температури, її абсолютний максимум та мінімум відповідно, $NDVI$, $NDVI_{\max}$, $NDVI_{\min}$ — усереднені значення нормалізованого різницевого вегетаційного індексу $NDVI$.

$VHI < 40$ – стресові умови;

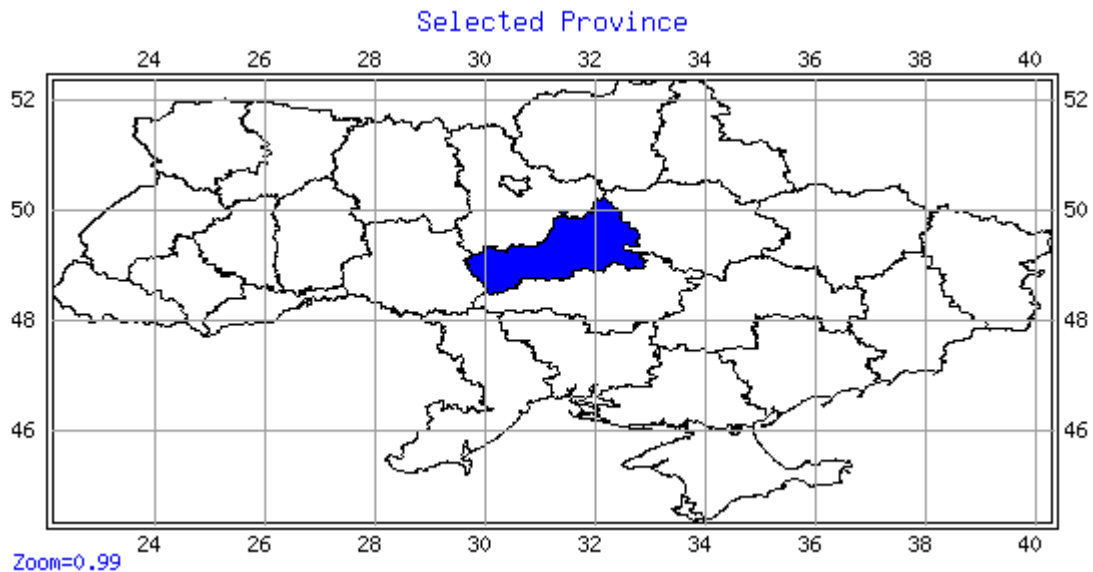
$VHI > 60$ – сприятливі умови;

$VHI < 15$ – посуха, інтенсивність якої від середньої до надзвичайної;

$VHI < 35$ – посуха, інтенсивність якої від помірної до надзвичайної.

Для розуміння фізичної природи даних, з якими Ви будете працювати, варто ознайомитися з роботами.

Для того, щоб перейти безпосередньо до роботи з даними на сторінці <http://www.star.nesdis.noaa.gov/smcd/emb/vci/VH/index.php> перейдіть за посиланням VH Info By Province, на якій зі списку країн оберіть Україну. В результаті виконання цих дій відобразиться інтерактивна карта України:



Нижче інтерактивної карти є посилання time series data, за яким можна завантажити дані з ходом вегетаційного індексу для виділеної області.

Скачування файлу за посиланням:

```
import urllib2
url="http://www.star.nesdis.noaa.gov/smcd/emb/vci/gvix/G04/ts_L1/ByProvince/
Mean/L1_Mean_UKR.R16.txt"
vhi_url = urllib2.urlopen(url)
out = open('vhi_id_16.csv','wb')
out.write(vhi_url.read())
out.close()
print "VHI is downloaded..."
```

Зчитати csv-файлу у фрейм, вивести імена стовпців та перший рядок:

```
import pandas as pd
df = pd.read_csv('vhi_id_16.csv',index_col=False, header=1)
print list(df.columns.values)
print df[1]
```

Вибрати із фрейму записи за умовою (запис із таблиці для поточної області, за 2000-й рік та 18-й тиждень):

```
df[(df['year']==2000) & (df['week']==18)]
```

1.3. Порядок виконання роботи

Створити env в якому будуть встановлені всі необхідні бібліотеки та налаштування для даної лабораторної роботи

Для кожної із адміністративних одиниць України завантажити тестові структуровані файли, що містять значення VHI-індексу. Ця процедура має бути автоматизована, параметром процедури має бути індекс (номер) області.

При зберіганні файлу до його імені потрібно додати дату та час завантаження;

Зчитати завантажені текстові файли у фрейм (детальніше про роботу із фреймами буде розказано у подальших лабораторних роботах). Імена стовбців фрейму мають бути змістовними та легкими для сприйняття (не повинно бути спеціалізованих символів, пробілів тощо). Ця задача має бути реалізована у вигляді окремої процедури, яка на вхід приймає шлях до директорії, в якій зберігаються файли;

Реалізувати процедуру, яка змінить індекси областей, які використані на порталі NOAA на наступні:

| № області | Назва | № області | Назва |
|-----------|-------------------|-----------|-----------------|
| 1 | Вінницька | 13 | Миколаївська |
| 2 | Волинська | 14 | Одеська |
| 3 | Дніпропетровська | 15 | Полтавська |
| 4 | Донецька | 16 | Рівенська |
| 5 | Житомирська | 17 | Сумська |
| 6 | Закарпатська | 18 | Тернопільська |
| 7 | Запорізька | 19 | Харківська |
| 8 | Івано-Франківська | 20 | Херсонська |
| 9 | Київська | 21 | Хмельницька |
| 10 | Кіровоградська | 22 | Черкаська |
| 11 | Луганська | 23 | Чернівецька |
| 12 | Львівська | 24 | Чернігівська |
| | | 25 | Республіка Крим |

Реалізувати процедури для формування вибірок наступного виду (включаючи елементи аналізу):

- Ряд VHI для області за рік, пошук екстремумів (min та max);
- Ряд VHI за всі роки для області, виявити роки з екстремальними посухами, які торкнулися більше вказаного відсотка області;
- Аналогічно для помірних посух

Примітка: Захист лабораторної роботи буде супроводжуватися додатковими завданнями по формуванням довільних вибірок та виконанням простого аналізу часового ряду (min, max, average, пошук підрахунок кількості спостережень, які відповідають тим чи іншим вимогам тощо). Для успішної здачі лабораторної умови необхідно чітко орієнтуватися в даних, з якими Ви працюєте.

1.4. Контрольні запитання

1. Основні кроки по вирішенню задачі аналізу даних
2. В чому переваги Python над іншими мовами програмування у сфері вирішення задач Data Science?

3. Дати визначення поняттю «ідеальний набір даних»
4. Які, на Вашу думку, існують джерела даних?
5. Які дані можна вважати «чистими» (clean data)?

1.5. Додаткові джерела інформації

1. <http://docs.python.org/tutorial/>
2. <http://www.greenteapress.com/thinkpython/>
3. <http://www.diveintopython.net/>
4. Kevin Sheppard. Introduction to Python for econometrics, statistics and data analysis. Self-published, University of Oxford, version 2.1 edition, February 2014
5. Finn Arup Nielsen. Data Mining with Python. Draft, December, 2014

Лабораторна робота №2

Наука про дані: обмін результатами та початковий аналіз

Мета роботи: ознайомитися з системою контролю версій GitHub, навчитися створювати прості веб-додатки для обміну результатами досліджень із використанням модуля `spurge`.

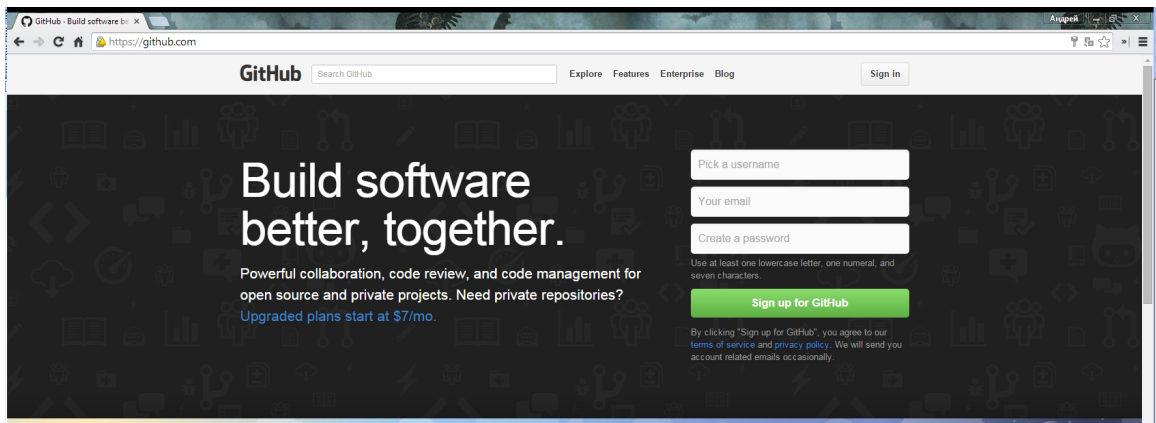
2.1. Теоретичні відомості

Вступ до роботи з GitHub

GitHub – веб-орієнтований хостінговий сервіс для проектів по розробці програмного забезпечення, побудований на базі системи контролю версій `git`. GitHub надає користувачам чимало можливостей:

- виконувати обмін файлами між локальним репозиторієм та тією версією, що зберігається на GitHub;
- Обмінюватися репозиторіями з іншими користувачами (fork repo);
- Спостерігати за роботою інших користувачів;
- Стандартні можливості системи контролю версій.

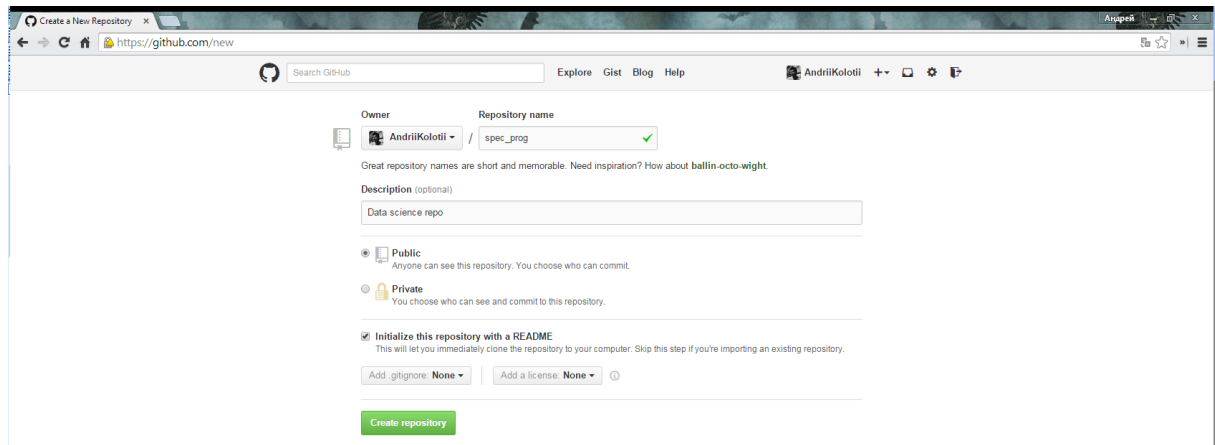
Для того, щоб користуватися GitHub потрібно створити обліковий запис (безкоштовні проекти на GitHub створюються лише з `public`-доступом) за адресою <https://github.com/>:



Git vs GitHub

Варто зазначити кілька моментів щодо використання сервісу GitHub:

- можна користуватися системою контролю версій git (локально) і без GitHub (веб);
- GitHub – зручний засіб для обміну результатами роботи з іншими розробниками та дослідниками.
- Основні способи створення такого репозиторію:
- створити новий репозиторій через веб-інтерфейс GitHub (New repository над переліком Ваших репозиторіїв);
- Дублювати собі репозиторій іншого користувача (fork).



Для початку варто використати перший спосіб, вказавши ім'я репозиторію (repository name), опис (description) рівень доступу інших користувачів (Public в безкоштовному тарифі), додавши до репозиторій файл README.

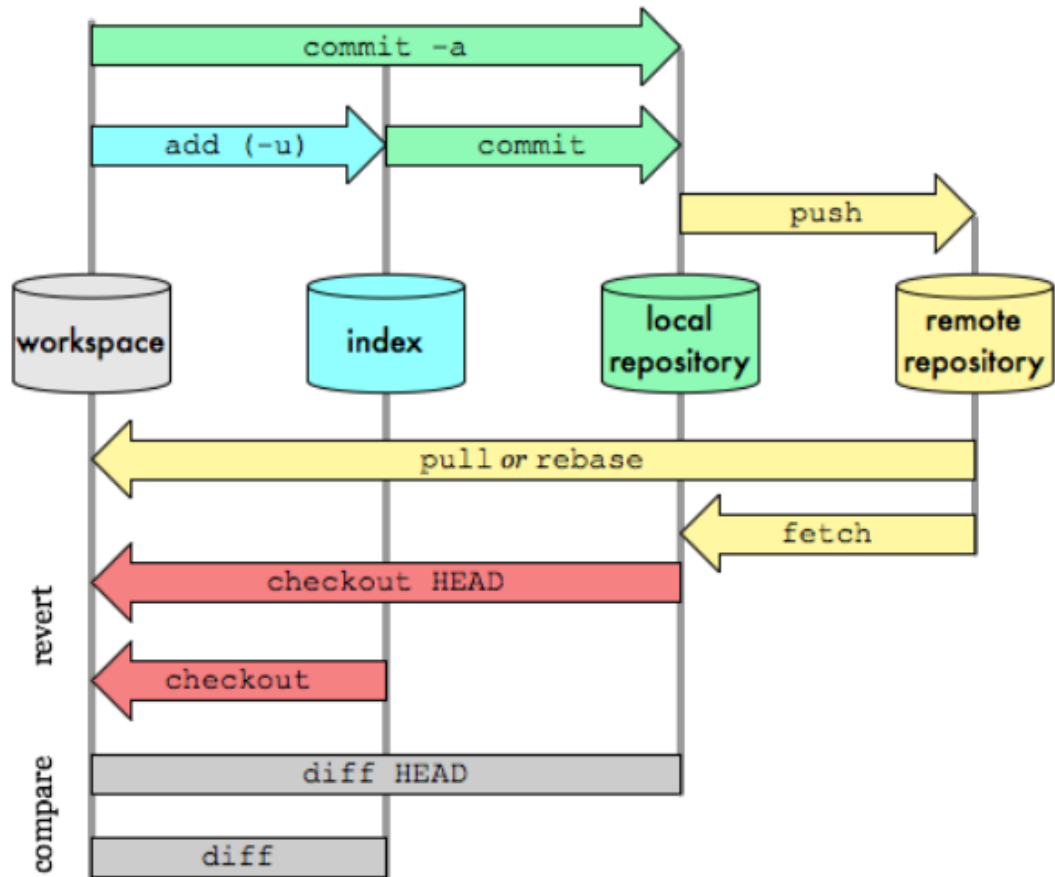
Для створення локальної версії репозиторія Вам знадобиться відповідний клієнт. Для ОС Windows - <https://windows.github.com>, для якого встановлюється як графічна оболонка, так і відповідний git shell.

Запустимо git shell та повернемося до щойно створеного репозиторію. Вам потрібно клонувати створений репозиторій до локального репозиторію. Для цього скористаємося командами:

- `git clone https://github.com/yourUserName/repoName.git`
- Тепер додамо файл до локального репозиторію.
- `git add filename`
- та відправимо зміни на сервер GitHub:
- `git commit -m "added data"`
- `git push`

Зміни були завантажені на сервер GitHub, а тому навіть працюючи за інших комп'ютером, Ви зможете отримати доступ до свого коду та даних (клонувавши репозиторій або завантаживши Ваш проект та дані і вигляді архіву).

Загалом схема роботи з системою контролю версій виглядає наступним чином:



2.2. Приклад

Демонстрація результатів аналізу у Web

Окрім власне вирішення поставленої в дослідження задачі важливим є також питання обміну результатами досліджень. Звичайно ж для цього є класичні шляхи - статті, конференції тощо. Однак буває досить корисним і просто мати можливість дати зовнішньому користувачу можливість оперувати зібраними даними та візуалізувати їх тим чи іншим способом.

Одним із таких способів в Python є використання Spyre, який дозволяє легко створювати веб-додатки із простим функціоналом.

Під час виконання першої лабораторної роботи необхідно було створити env з бібліотеками matplotlib, pandas, numpy, cherry, jinja2, dataspyre.

Детальну інформацію про роботу з модулем Spyre можна знайти за адресою <http://nbviewer.ipython.org/github/adamhajari/spyre/blob/master/tutorial/SpyrePresentation.ipynb> та на відповідній сторінці проекту на GitHub <https://github.com/adamhajari/spyre>.

2.3. Порядок виконання роботи

Зареєструватись на сайті GitHub, створити репозиторій (саме створити, а не “форкнути” результат друга чи подруги □), додати в репозиторій код та дані із лабораторної роботи №1, продемонструвати навички роботи з системою контролю версій git на роботі з проектом GitHub;

Створити веб-додаток із використанням модуля Spyre, який дозволить:

- обрати часовий ряд VCI, TCI, VHI для набору даних із лабораторної роботи 1 (випадаючий список);
- Вибрати область, для якої буде виконуватись аналіз (випадаючий список);
- Зазначити інтервал тижнів, за які відбираються дані;
- Створити кілька вкладок для відображення таблиці із даними на графіку ходу індексів;

Інші завдання на прохання викладача.

Код розробленого додатку додати до створено репозиторію

2.4. Контрольні запитання

1. Для кого призначена система git?
2. Назвати основні можливості, які надає система контролю версій та використання GitHub.
3. Які ще способи створення інтерактивних додатків в Python Ви знаєте?
Які існують аналоги в інших мовах, що застосовуються для аналізу даних (R, Matlab).

2.5. Додаткові джерела інформації

1. <http://git-scm.com/doc>
2. <http://git-scm.com/doc>
3. <https://github.com/adamhajari/spyre>
4. <http://nbviewer.ipython.org/github/adamhajari/spyre/blob/master/tutorial/SpyrePresentation.ipynb>

Лабораторна робота №3

Структури для роботи з великими обсягами даних в Python

Мета роботи: отримати навички роботи із структурами для зберігання в Python (`python`, `numpy`, `pandas`, `numpy array`, `dataframe`, `timeit`).

3.1. Теоретичні відомості

Мінімально необхідні навички роботи із масивами та фреймами вже отримано при виконанні лабораторних робіт 1 та 2.

За потреби можна скористатись офіційними (та не дуже ☺) сторінками відповідних проєктів:

<http://pandas.pydata.org/pandas-docs/version/0.15.2/index.html>

<http://scipy-lectures.github.io/intro/numpy/numpy.html>

3.2. Приклад

Скрипт приймає два параметри: вихідний вектор та csv-файл.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#to import the basemap library give the direct path to the library
import os
os.environ["PROJ_LIB"]="\"
from mpl_toolkits.basemap import Basemap
import geopandas as gpd

city=gpd.read_file("Boundary.shp")

csv=pd.read_csv("latlong_raj.csv")

lat=csv['LAT'].values
lon=csv['LONG'].values
population = city['POPULATION'].values
dist=city['DIST_NAME'].values

fig = plt.figure(figsize=(8, 8))
m = Basemap(projection='lcc', resolution='h',
```

```

        lat_0=27.0238, lon_0=74.2179,
        width=1.05E6, height=1.2E6)
m.shadedrelief()

m.drawcoastlines(color='blue',linewidth=3)
m.drawcountries(color='gray',linewidth=3)
m.drawstates(color='gray')

# scatter city data, with c reflecting population
m.scatter(lon,lat, latlon=True,
          c=population,s=700,
          cmap='YlGnBu_r', alpha=0.5)
#create colorbar
plt.colorbar(label=r'Population')
plt.clim(300000, 4000000)

dict1={}
list1=[]
list2=[]
list3=[]
n=0
#storing each value in different lists
for z in lat:
    list1.append(z)
for c in lon:
    list2.append(c)
for b in dist:
    list3.append(b)
#storing the values of lat long in a dictionary with lat as keys and long as values
while(n<len(list1)):
    dict1[list1[n]]=list2[n]
    n+=1

i=0
# Map (long, lat) to (x, y) for plotting
#naming the cities of Rajasthan with the help of their lat(z)long(c)
for z,c in dict1.items():
    x,y = m(c, z)
    plt.plot(x, y, 'ok', markersize=5)
    plt.text(x, y,list3[i], fontsize=10);
    i+=1

```

3.3. Порядок виконання роботи

В ході виконання лабораторної роботи необхідно оцінити час виконання поставленого завдання із використанням масивів (`numpy array`) та фреймів (`pandas dataframe`).

Для кожної із структур даних потрібно виконати профілювання часу виконання (використайте `timeit` із однойменного модуля).

Наука про дані (*Data science*) дозволяє вирішувати широке коло задач, до яких належать не тільки суто наукові задачі теоретичного плану, але й цілком практичні, які є частиною нашого звичайного життя. Оскільки наразі доволі важливою є проблема економії енергоресурсів, то розглянемо задачу, пов'язану із споживанням електричної енергії домогосподарством.

Вашій увазі пропонується набір даних *Individual household electric power consumption Data Set*, який можна завантажити із UCI-репозиторію (за посиланням **Data folder**): <https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption>

Детальний опис набору даних можна знайти за наведеним вище посиланням, якщо ж “*коротко і по суті*”, то це відомості щодо основних витрат електричної енергії домогосподарствами, зібрані впродовж 47 місяців (12.2006 – 11.2010).

Перелік атрибутивної інформації:

date: дата виміру у форматі dd/mm/yyyy

time: час у форматі hh:mm:ss

global_active_power: активна потужність, яку споживає домогосподарство за хвилину (усереднено) [кВт]

global_reactive_power: реактивна потужність, яку споживає домогосподарство за хвилину (усереднено) [кВт]

voltage: напруга, усереднена за хвилину спостереження [В]

global_intensity: усереднена силу струму для домогосподарства [А]

sub_metering_1: набір споживачів енергії №1 [Вт-годин активної енергії], відповідає кухні, на якій є машина для миття посуду на мікрохвильовка (електричної плити немає, використовується газова).

sub_metering_2: набір споживачів енергії №2 [Вт-годин активної енергії], відповідає пральні, в якій працює пральна машина, сушарка, холодильних та ввімкнено світло.

sub_metering_3: набір споживачів енергії №3 [Вт-годин активної енергії], відповідає бойлеру та кондиціонеру.

3.4. Завдання

Виконати всі завдання, використовуючи як `numpy array`, так і `dataframe`, проаналізувати часові витрати на виконання процедур (профілювання часу виконання), зробити висновки щодо ситуацій, в яких має сенс віддати перевагу тій чи іншій структурі даних. Висновки оформити звітом із зазначеним часом виконання та оцінкою по 5-бальній шкалі зручності виконання операцій відбору).

Також варто звернути увагу на те, що дані, як і практично все в реальному житті, можуть потребувати Вашої уваги ☺ - потрібно залишити лише ті спостереження, в яких немає порожніх спостережень (порожні значення – пусті поля між роздільником – ? – 28.04.2007, як приклад).

1. Обрати всі домогосподарства, у яких загальна активна споживана потужність перевищує 5 кВт.
2. Обрати всі домогосподарства, у яких вольтаж перевищує 235 В.
3. Обрати всі домогосподарства, у яких сила струму лежить в межах 19-20 А, для них виявити ті, у яких пральна машина та холодильних споживають більше, ніж бойлер та кондиціонер.
4. Обрати випадковим чином 500000 домогосподарств (без повторів елементів вибірки), для них обчислити середні величини усіх 3-х груп споживання електричної енергії, а також
5. Обрати ті домогосподарства, які після 18-00 споживають понад 6 кВт за хвилину в середньому, серед відібраних визначити ті, у яких основне споживання електроенергії у вказаний проміжок часу припадає на пральну машину, сушарку, холодильник та освітлення (група 2 є найбільшою), а потім обрати кожен третій результат із першої половини та кожен четвертий результат із другої половини.

3.5. Додаткові джерела інформації

1. <https://dystosvita.gnomio.com/mod/page/view.php?id=4928>
2. <https://pythonguide.rozh2sch.org.ua/>
3. <http://programming.in.ua/programming/python/219-python-structure-of-data.html>

Лабораторна робота № 4

Сценарії обробки багатоспектральних супутникових зображень

Мета роботи: ознайомитися з основними принципами обробки даних дистанційного зондування Землі (ДЗЗ) та можливостями мови Python для обробки геопросторової інформації, а саме, багатоспектральних супутникових зображень засобами бібліотеки абстракції супутникової інформації (GDAL).

4.1. Теоретичні відомості

Космічні апарати (КА) дистанційного зондування Землі (ДЗЗ) є одним з основних джерел інформації про навколишнє середовище. Неперервний розвиток космічних технологій є однією з рушійних сил в прогресі розв'язання задач глобального моніторингу океану, суші та атмосфери. Дані ДЗЗ оперативно використовуються в різноманітних соціально-економічних сферах: сільському та лісовому господарствах, екологічному моніторингу, картографуванні та земельному кадастрі, а також в задачах прогнозування та моніторингу надзвичайних ситуацій.

Одним з широко розповсюджених видів даних ДЗЗ є супутникові зображення поверхні Землі. Останні вважаються багатоспектральними (мультиспектральними), якщо отримані в результаті обробки вихідної інформації сенсору КА, який здатний проводити зйомку в видимому (Синій, Зелений, Червоний) та інфрачервоному (ближній, короткохвильовий та тепловий) спектральних діапазонах.

Для успішної тематичної обробки супутникових зображень, чи їх фотоінтерпретації, потрібно володіти не тільки знаннями в предметній області, але й вміннями оперативно оброблювати великі об'єми даних та працювати зі спеціалізованими програмними бібліотеками. Для попередньої підготовки до обробки даних ДЗЗ використовують бібліотеку абстракції геопросторової інформації GDAL [1]. Дана бібліотека є вільно розповсюджуваною та часто оновлюваною. Багато програмних продуктів використовують GDAL і надають графічний інтерфейс доступу до сценаріїв бібліотеки, зокрема, до них належать наступні: QGIS, ArcGIS, OpenEV, MapServer, Google Earth та ін.

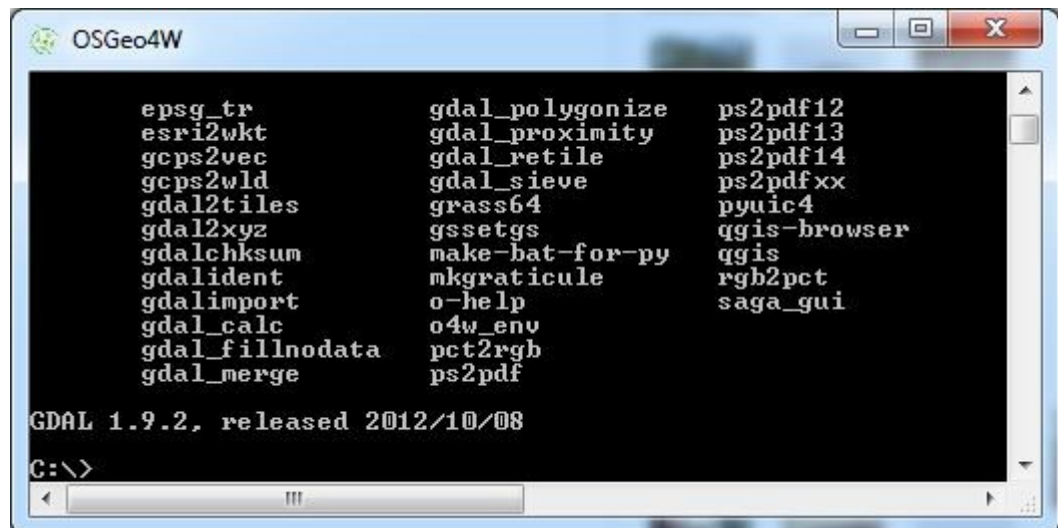
Налаштування робочого середовища

Перед тим як безпосередньо перейти до роботи з растровими зображеннями, варто сказати кілька слів про встановлення бібліотеки GDAL.

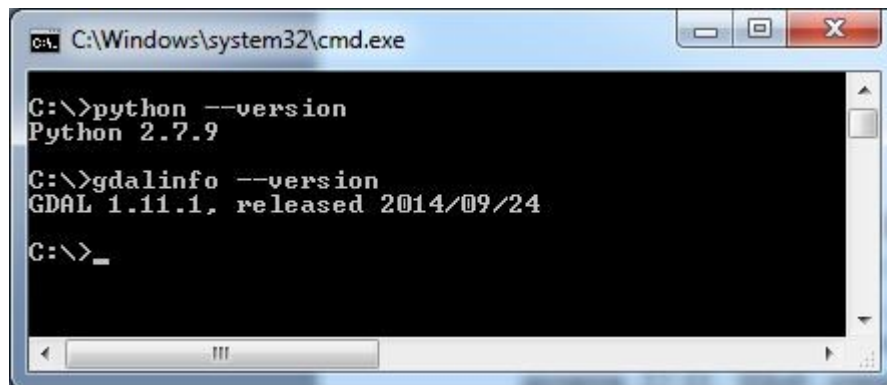
В межах виконання лабораторної роботи Ви можете встановити бібліотеку GDAL як мінімум двома різними варіантами.

Варіант перший. Встановлення програмного продукту, що містить готові бінарії (скомпільований код) бібліотеки GDAL, наприклад QGIS (Quantum GIS). Даний продукт є геоінформаційною системою та розповсюджується вільно. Завантажити його можна за електронною адресою: <http://qgis.org/downloads/> (в якості рекомендації, пропонується завантажити та використовувати версію 3.12.0, яка використовувалась при розробці методичних вказівок). У випадку, якщо Ви користуєтесь ОС Windows, потрібно також додати директорію встановленої QGIS до системних шляхів та перезавантажити систему (модифікувавши системну змінну PATH, аналогічно тому, як описано у лабораторній роботі № 1).

Виконавши всі вказані кроки, запускаємо OSGeo4W.bat – інтерфейс командного рядка Windows для роботи з сценаріями GDAL.



Варіант другий. Якщо у Вас на робочому ПК вже встановлений Python (рекомендована версія Python 3.6), то Ви маєте можливість використовувати GDAL як бібліотеку Python. Для цього потрібно приєднати шляхи до бібліотеки одним із запропонованих шляхів [2, 3]. Для отримання доступу до бінаріїв бібліотеки в перелік системних змінних обов'язково потрібно додати змінні **GDAL_DATA** зі значенням C:\Program Files (x86)\GDAL\gdal-data\ та **GDAL_DRIVER_PATH** з значенням C:\Program Files(x86)\GDAL\gdalplugins\. Майте на увазі, значення змінних можуть відрізнятися, в залежності від місця розташування файлів бібліотеки на Вашому ПК. Також не забудьте модифікувати системну змінну **PATH**, додавши до неї записи C:\Program Files (x86)\GDAL та C:\Python36. Після перезавантаження системи в командному рядку Windows Ви зможете користуватись сценаріями бібліотеки GDAL.



```

C:\Windows\system32\cmd.exe

C:\>python --version
Python 2.7.9

C:\>gdalinfo --version
GDAL 1.11.1, released 2014/09/24

C:\>_

```

Варіант третій. (самий простий). У разі, якщо Python надається в межах дистриб'юції Anaconda, то установка модуля gdal зводиться до використання наступної команди:

```
conda install -c conda-forge gdal
```

Установка sentinelhub. Для зручного завантаження продуктів Sentinel рекомендується використовувати модуль sentinelhub для Python. Інструкція з установки та використання можна знайти за посиланням на Github-проект <https://github.com/sentinel-hub/sentinelhub-py>.

Коротко про супутникові зображення (довідкова інформація)

Супутникові зображення представляються кінцевим користувачам у вигляді продуктів певного рівня обробки вихідного сигналу сенсору космічного апарата. Повний перелік рівнів обробки даних, та їх номенклатура можуть відрізнятися у різних операторів даних ДЗЗ. Але міжнародним комітетом по супутниковим спостереженням Землі з космосу CEOS за основу прийнята класифікація рівнів обробки даних NASA [4].

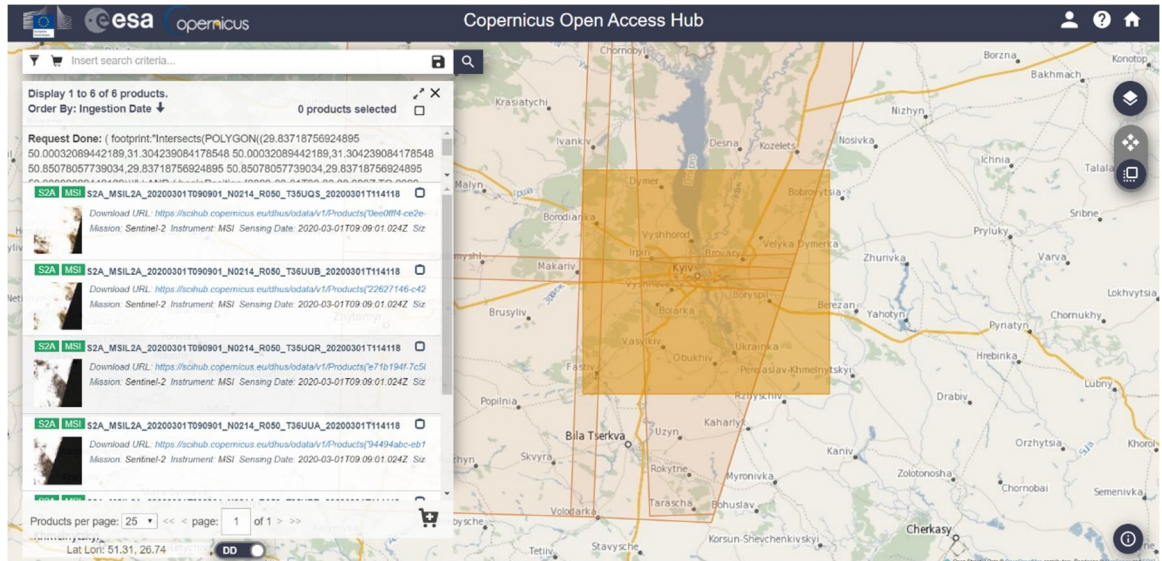
В ході виконання лабораторної роботи ви ознайомитеся з продуктами обробки даних КА Landsat-8 та Sentinel-2. Вказані продукти розповсюджуються у вигляді архівів. Останні містять канали зображень, та супутню службову інформацію.

Для завантаження даних Landsat-8 можна використовувати геопортал геологічної служби США – USGS (<http://earthexplorer.usgs.gov/>). Кожне зображення КА серії Landsat має унікальний ідентифікатор, структура якого наступна:

[Ідентифікатор сенсора та КА (3 символи)]+[Координати знімку в WRS-2 системі (Path, Row, 6 символів)]+[Year]+[DOY]+[Режим зйомки (5 символів)].

Провайдером даних місії Sentinel є програма Copernicus. Дані супутника Sentinel-2 можна отримати з ресурсу Copernicus Open Access Hub за адресою <https://scihub.copernicus.eu/>. Його можна використовувати для пошуку та

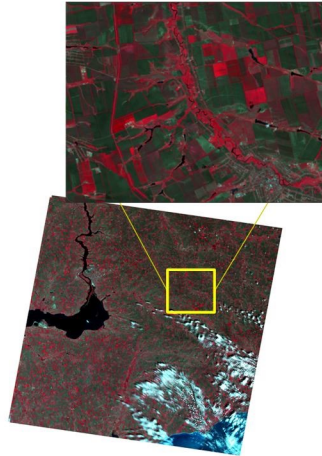
завантаження даних. Також для зручної роботи з сервісом Copernicus Scihub у середовищі Python розроблено бібліотеку `sentinelhub`.



В багатьох задачах тематичної обробки супутникових зображень часто виникають потреби в фото-інтерпретації отриманого зображення, зміні його роздільної здатності, перепроєктуванні зображення до вказаної (потрібної) системи координат, а також в обрізанні чи конкатенації зображень тощо. З деякими із вказаних задач пропонується ознайомитись в ході виконання лабораторної роботи.

Людське око здатне фіксувати навколишнє середовище в так званому RGB (true-color) композиті відбитого світлового випромінювання. В той же час багатоспектральні супутникові зображення складаються куди з більшої кількості зафіксованих спектральних каналів, ніж синій (Blue), зелений (Green) та червоний (Red). І це дає можливість проводити детальний аналіз відзнятої поверхні Землі. Наприклад на рисунку наведено приклад зображення КА Landsat-7 ETM+ в комбінації каналів Green, Red, NIR (color infrared). На зображенні території з більш насиченим червоним кольором відповідають поверхні, що відповідає рослинності, що більш вегетує.

Більш детальніше ознайомитись з фізичною інтерпретацією комбінацій каналів супутникових зображень КА Landsat можна у статті “Інтерпретація комбінацій каналів даних Landsat TM/ETM+” на ресурсі <http://gis-lab.info/> [5].



До основних характеристик супутникового зображення відносяться наступні:

- розміри зображення;
- кількість спектральних каналів зображення;
- просторова роздільна здатність зображення (розміри пікселя, вимірюються в метрах);
- система координат, в яку спроектовано зображення;
- координати крайніх правого верхнього та нижнього, а також лівого верхнього та нижнього пікселів зображення як в заданій системі координат, так і на поверхні Землі у вигляді значень широти/довготи;

Отримати зазначені характеристики зображення можна, викликавши команду **gdalinfo** *<input.tif>*, де *input.tif* — зображення, що розглядається. Нижче наведено неповний вивід команди **gdalinfo** до знімку КА Landsat-8 з ідентифікатором LC81810252013106LGN01.TIF

| | |
|----------------------------------|-----------------------------|
| Driver: GTiff/GeoTIFF | #драйвер обробки зображення |
| Files: LC81810252013106LGN01.TIF | #ідентифікатор зображення |
| Size is 7911, 7711 | #розміри зображення |
| Coordinate System is: | |
| PROJCS["WGS 84 / UTM zone 36N", | #система координат |
| ... | |

```

Origin = (220785.0000000000000000,5687415.0000000000000000)
Pixel Size = (30.0000000000000000,-30.0000000000000000)
...
Corner Coordinates:                                #координати кутових пікселів
Upper Left ( 220785.000, 5687415.000) ( 28d59'48.89"E, 51d16'10.38"N)
Lower Left ( 220785.000, 5456085.000) ( 29d10' 2.19"E, 49d11'38.05"N)
Upper Right ( 458115.000, 5687415.000) ( 32d23'55.42"E, 51d20'11.31"N)
Lower Right ( 458115.000, 5456085.000) ( 32d25'27.81"E, 49d15'21.95"N)
Center ( 339450.000, 5571750.000) ( 30d44'48.29"E, 50d16'33.70"N)
                                     #далі для кожного каналу
                                     #зображення приводитиметься
                                     #статистика спектральної
                                     #яскравості (значень пікселів)
Band 1 Block=7911x1 Type=UInt16, ColorInterp=Gray
Min=0.000 Max=22196.000
Minimum=0.000, Maximum=22196.000, Mean=6250.302,
StdDev=4752.830
Metadata:
  STATISTICS_MAXIMUM=22196
  STATISTICS_MEAN=6250.302386522
  STATISTICS_MINIMUM=0
  STATISTICS_STDDEV=4752.8300813332

```

Робота з растровими даними через gdal

```

import gdal - імпорт бібліотеки gdal
Import osr - імпорт бібліотеки osr (для роботи з координатними
системами)
Raster = gdal.Open(tif_path) - відкрити растр за шляхом tif_path через
python
Band = Raster.GetRasterBand(1).ReadAsArray() - зчитати 1 канал растру
у вигляді масиву numpy
Raster.GetRasterBand(1).WriteArray(Band2,0,0) - запис масиву Band2 у
перший канал растру на позицію з координатами правого верхнього куту
0, 0
Raster = None - завершення роботи з растром

```

Лабораторна робота має дві частини. У першій (простішій) частині (пп. 1– 2) необхідно працювати з даними Sentinel-2, які необхідно завантажити за допомогою Copernicus Scihub за ідентифікаторами (список ідентифікаторів

дивись нижче) або скористатися архівом, що надається викладачем. У другій (складнішій) частині (п. 4) потрібно також самостійно завантажити дані Landsat з порталу USGS. Для цього підійдуть будь які 2 сусідніх знімки Landsat-8 з відсотком хмарності до 30%, (рекомендовані ідентифікатори наведено внизу у п. 3).

1. Завантажити необхідні дані Sentinel-2 (або з Copernicus SciHub, або з Google-диска). За допомогою бібліотеки sentinelhub написати скрипт для завантаження супутникових знімків Sentinel-2, які покривають Київ (ROI — POLYGON((29.073321247506765 49.845775018245774,31.986007792928522 49.845775018245774,31.986007792928522 51.278667808079206,29.073321247506765 51.278667808079206,29.073321247506765 49.845775018245774)))

) Список

ідентифікаторів Sentinel-2:

- S2A_MSIL2A_20190821T085601_N0213_R007_T36UUB_20190821T115206
- S2A_MSIL2A_20190821T085601_N0213_R007_T36UUA_20190821T115206

2. Засобами командного рядка операційної системи (демонстраційні приклади розглянуті для ОС Windows), а також за допомогою бінаріїв бібліотеки GDAL розробити автоматичний сценарій, який здійснюватиме обробку даних ДЗЗ, відповідно до поставлених завдань. Для виконання команд gdal у сценарії Python можна використовувати бібліотеки os та command.

Над завантаженими даними Sentinel-2 відпрацювати наступні дії.

- Розпакування наборів архівів з продуктами ДЗЗ в новостворені папки, назви яких відповідатимуть ідентифікаторам зображень.
- Конкатенація каналів видимого, ближнього та середнього інфрачервоного спектральних діапазонів (канали 2, 3, 4 та 8) зображення в єдиний GEOTIFF файл.
- Перепроєктування супутникового зображення у проєкцію 4326.
- Конкатенація всіх отриманих растрів у один 4-канальний файл TIFF.
- Обрізання результуючого зображення по заданому векторному контуру.

3. Протестувати різні реалізовані в gdal методи паншарпенінгу. Для цього підготуйте склеєні RGB канали Landsat-8 та окремо панхроматичний канал. Переведіть панхроматичний канал у 30 м, а RGB канали у 60 метрів. Після цього паншарпніть 60 метровий RGB за допомогою 30 м панхроматичного каналу. Після цього ви отримуєте два 30 метрових RGB зображення - перший оригінальний, другий паншарпнутий. Підрахуйте метрики точності паншарпенінгу (це будь які вам відомі метрики які можна

використовувати для валідації регресії, наприклад r^2 square). За даними метриками виберіть найкращий метод паншарпенінгу.

Список ідентифікаторів Landsat-8:

1. LC08_L1TP_182025_20190830_20190903_01_T1
2. LC08_L1TP_182026_20190830_20190903_01_T1

4.2. Приклад

Детальні інструкції по використанню бінаріїв GDAL можна отримати безпосередньо на офіційному інтернет ресурсі <http://www.gdal.org/>

Основні команди, які Вам доведеться використовувати, наведено нижче.

Розархівування каналів зображення. Ви вільні вибирати зручний архіватор. Якщо ви користуєтесь архіватором “7z”, тоді розпакувати всі архіви, що містяться в заданій папці, можна шляхом виклику з неї наступного сценарію (bat файл):

```
for /f "tokens=1 delims=." %%d in ('dir /b /d *tar.gz') do ( md
%%d
@echo Directory %%~nd Created
7z x %%d.tar.gz 7z x -o%%~nd %%d.tar del /S %%d.tar
@echo Archive %%~nd %%d.tar.gz uncompressed break)
```

де в якості параметра %%d виступатимуть імена всіх файлів з формату tar.gz.

Конкатенація каналів зображення. В архіві tar.gz. містяться канали супутникового зображення в окремих файлах GEOTIFF. Для їх конкатенації потрібно викликати наступну команду:

```
gdal_merge.py -o <output.tif> -separate <input_1.tif> <input_2.tif> ...
<input_N.tif>
```

де <output.tif> – результат конкатенації, <input.tif> – канали зображення, взяті з відповідного архіву.

Більш детальну інформацію можна знайти за адресою https://gdal.org/programs/gdal_merge.html.

Перепроєктування зображення. Для зміни географічної проекції зображення потрібно скористатися командою gdalwarp, наприклад:

```
gdalwarp -tr xSize ySize -t_srs "[projection parameters]" <input.tif>
<output.tif>
```

де xSize,ySize – розміри пікселя результуючого зображення; [projection parameters] – параметри проекції, в яку буде здійснюватися перепроєктування, наприклад “+proj=utm +zone=36” – параметри проекції

UTM Zone 36N. Для подальшого обрізання зображення по векторному контуру, потрібно буде звести растр з векторним контуром до єдиної проекції. Детальніше — <https://gdal.org/programs/gdalwarp.html>

Конкатенація зображень з різними Path Row. Для конкатенації зображень, що не співпадають територіально (повністю або частково), слід використовувати наступну команду gdalwarp:

```
gdalwarp -of GTIFF -ot UInt16 -srcnodata <value> -dstnodata <value>
<input_1.tif> <input_2.tif> ... <input_N.tif> <output.tif>
```

де значення параметрів srcnodata та dstnodata відповідають значенням пікселів вхідних та результуючого зображень з відсутніми даними відповідно. Значення параметру -of відповідає формату вихідного файлу, а значення параметру -ot – його типу.

Відмінність в конкатенації зображень командами gdal_merge та gdalwarp полягає в наступному:

- за допомогою gdal_merge можна об'єднати тільки зображення однакового розміру (Size). При цьому результат конкатенації буде мати той же розмір, а кількість каналів результуючого зображення стане рівна сумі кількостей каналів вхідних зображень;
- за допомогою gdalwarp можна об'єднати зображення що територіально не перекриваються і мають різні розміри, але кількість каналів в них обов'язково повинна бути однаковою.

Обрізання растрових зображень по векторному контуру. Оскільки супутникові зображення займають чимало дискового простору, часто виникає потреба оброблення тільки певної ділянки відзнятої території (кількох полів, району, області, тощо). Обрізання растрових зображень по векторному контуру часто здійснюється для підготування постерів та проектів. Приклад команди для здійснення обрізання растру по векторному контуру наведений нижче:

```
gdalwarp -dstnodata <value> -q -cutline <ShapeFile.shp> -crop_to_cutline -
of GTiff <input.tif> <output.tif>
```

де в файлі ShapeFile.shp міститься необхідний векторний контур. Значення <value> параметру -dstnodata відповідає значенню пікселів з відсутніми даними вхідного растру. <input.tif> та <output.tif> відповідно вхідне та обрізане растрові зображення.

Паншарпенінг. При наявному панхроматичному каналі, а такий канал є у супутника Landsat-8, можна збільшити розрізнення деяких каналів мультиспектру до розрізнення панхроматичного каналу за допомогою процедури паншарпенінгу.

```
gdal_pansharpen pan_dataset spectral_dataset out_dataset -r
```

де pan_dataset — це шлях до панхроматичного каналу, spectral_dataset - шлях до мультиспектральних каналів, out_dataset — шлях до вихідного файлу з результатами, r — метод паншарпенінгу. Детальніше https://gdal.org/programs/gdal_pansharpen.html.

Примітка. Якщо з растрового зображення необхідно вирізати деяку прямокутну ділянку, то не обов'язково будувати її векторний контур, можна просто скористатися наступною командою:

```
gdalwarp -tr xSize ySize -te XLowerLeft YLowerLeft XUpperRight YUpperRight <input.tif> <output.tif>
```

де XLowerLeft YLowerLeft XUpperRight YUpperRight — координати (x, y) крайнього лівого нижнього та правого верхнього пікселів прямокутного контуру, по якому буде здійснюватись обрізка.

4.3. Порядок виконання роботи

Отримати від викладача архіви з супутниковими зображеннями, та файл з векторним контуром для подальшого обрізання. Засобами командного рядку операційної системи (без обмежень у виборі ОС) та з використанням бінаріїв бібліотеки GDAL створити програмний сценарій для автоматичної обробки супутникових зображень відповідно до поставлених завдань.

Розроблений сценарій та результати обробки супутникових зображень продемонструвати викладачеві.

Можливі варіанти виконання лабораторної роботи.

| | |
|-------------|--|
| Варіант № 1 | пп. 1, 2 з використанням наданого архіву даних |
| Варіант № 2 | пп. 1–2 з самостійним отриманням даних Sentinel-2 |
| Варіант № 3 | пп. 1–3 з самостійним отриманням даних Sentinel-2 та Landsat-8 |

4.4. Контрольні запитання

1. Що таке супутникові зображення, які їх основні характеристики?
2. Які основні можливості оброблення растрових супутникових зображень надає бібліотека GDAL?
3. Чому відповідає кількість каналів растрового супутникового зображення?

4. Який з спектральних каналів супутникового зображення відповідає за тепловий відбиток відзнятого об'єкту?
5. Які супутникові зображення вважаються –багато, гіперспектральними?
6. Що таке просторова роздільна здатність супутникового зображення?
7. Які супутникові зображення вважаються з низьким, середнім та високим просторовим розрізненням?

4.5. Додаткові джерела інформації

1. GDAL – Geospatial Data Abstraction Library. Електронний ресурс. Режим доступу станом на 23.02.2015 – <http://www.gdal.org/>
2. Install GDAL on Windows // Cartometric Blog. Електронний ресурс. Режим доступу станом на 03.02.2015 – <http://cartometric.com/blog/2011/10/17/install-gdal-on-windows/>
3. Installing GDAL (and OGR) for Python on Windows // A technical blog concerning Python programming and, in particular, GIS applications. Електронний ресурс. Режим доступу станом на 23.02.2015 – <https://pythongisandstuff.wordpress.com/2011/07/07/installing-gdal-and-ogr-for-python-on-windows/>
4. Лупян Е.А. Базовые продукты обработки данных дистанционного зондирования Земли / Е.А. Лупян, В.П. Саворский // Современные проблемы дистанционного зондирования Земли из космоса. – 2012. Т.9. №2. С. 87-96. http://d33.infospace.ru/d33_conf/sb2012t2/87-96.pdf
5. Интерпретация комбинаций каналов данных Landsat TM/ETM+ // GisLab. Географические ИС и ДЗЗ. Електронний ресурс. Режим доступу станом на 23.02.2015 – <http://gis-lab.info/qa/landsat-bandcomb.html>.