Практична робота №7. Проєкт, модулі, імпорт бібліотек, pip. Робота з файлами у Python.

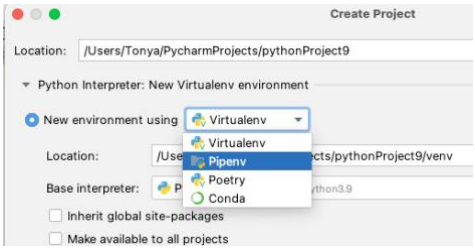
Нікіта Тєлєгін, 3 група.

https://github.com/Nikita-ukma/project_template.git

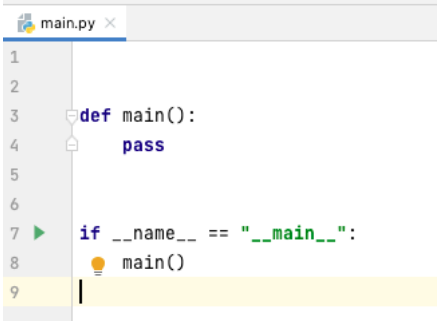
0. Підготувати звіт, де в репозиторії та скріншотах відображається кожен етап, який пізніше Ви зможете прикріпити у мудл. Цей файл використовувати в якості шаблону для звіту.

1. **Створення нового проєкту.**

   a. Створити новий проєкт локально у PyCharm або VSCode (можна частково використовувати інструкції з ПЗ 1). При створенні проєкта, назвіть його «project_template» та оберіть створення віртуального середовища venv (***робота з pipenv самостійно на оцінку 80-90**), main.py створимо пізніше. Назву папки віртуального середовища запам'ятайте, ми використаємо її пізніше.
   ***для роботи з pipenv при створенні проєкту**



   b. Підготуйте файл .gitignore, щоб папки типу venv або .idea і.т.п. не потрапили до репозиторію, який призначений суто для коду проєкту.
   c. Створіть файл main.py у директорії проєкту, який матиме наступний вигляд:



d. Створіть також новий репозиторій на GitHub (теж підглянути, як це робиться, можете у ПЗ 1).
e. Об’єднайте локальний та віддалений репозиторії. Залийте зміни на віддалений репозиторій (тут теж можете згадати ПЗ 1). Посилання на нього додайте на початок звіту.

2. **Структура проєкту.**
   a. Створити в директорії проєкту нову папку (Python Package – директорія, яка має одразу пустий файл __init__.py) і назвати її «app».
   Це є місце, де структуровано зберігаються модулі проєкту з кодом, який безпосередньо бере участь у запуску та виконанні задач застосунку. Тобто це код, який запускається користувачем (у його ролі може бути як людина, що на кнопку на фронтенді натиснула, так і інша система, яка, наприклад, використовує результати поточної).
   b. Усередині цієї директорії app створити Python Package «io» (скорочено input-output).
   c. У цій директорії io створити два файли: input.py та output.py.
   d. Створити ще один Python Package і назвати його «tests». *Це є директорія, що містить unit тести, та буде дзеркальною для app (тобто, наприклад, файл test_input.py у tests відповідатиме файлу input.py у app, і те саме для піддиректорії io у app та test_io у tests і т.д.).*
   e. Залити зміни на віддалений репозиторій з відповідним повідомленням у коміті.

3. **Робота з модулями.**
   1. **Якщо ви працюєте з pipenv, перейдіть до кроку 3.** Переконайтесь, що ваше віртуальне середовище активовано. Якщо ні, переходьте до кроку 2. Щоби перевірити, що середовище активовано, використайте відповідну команду, яка покаже, який інтерпретатор використовується в даний момент у проєкті.

Для Windows:

```
where python
```

Для Unix/MacOS:

```
which python
```

Маєте побачити повний шлях до віртуального середовища у проєкті. Наприклад:

```
(venv) Tonya@MacBook-Air-Antonina pythonProject8 % which python
/Users/Tonya/PycharmProjects/pythonProject8/venv/bin/python
```

2. Активуйте його самостійно за допомогою наступних команд у терміналі у директорії проєкту можна переключитись за допомогою команди
   `cd path/to/proj_dir`)

   Для Windows:

```
nazva_venv\Scripts\activate
```

   Для Unix/MacOS:

```
source nazva_venv/bin/activate
```

   де nazva_venv – це назва папки з віртуальним середовищем при створенні у вашому проєкті, скоріше за все, вона має назву venv.

3. Підготуйте pip.
   Для Windows:

```
py -m pip install --upgrade pip
py -m pip --version
```

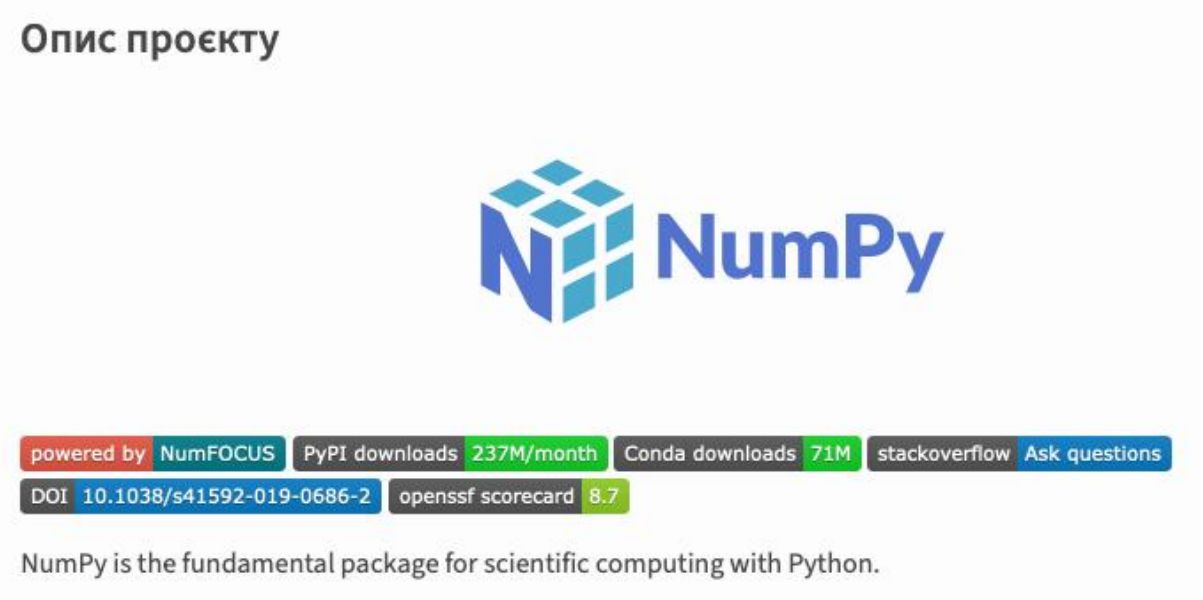
   Для Unix/MacOS:

```
python3 -m pip install --upgrade pip
python3 -m pip --version
```

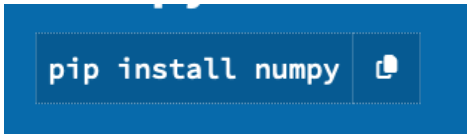
Після цього маєте побачити свіжу версію менеджеру пакетів pip.

4. Встановлюємо пакети через pip.
   Якщо ви працюєте з pipenv, після прочитання цієї статті https://realpython.com/pipenv-guide/ виконайте аналогічні для pipenv інструкції нижче (мається на увазі не виконання 1-в-1, а знаходження інструкцій, як зробити ту ж саму логіку, але через pipenv).

   4.a. Встановлення останньої версії пакету.
   Для цього рекомендую вам перейти на сайт https://pypi.org та в пошуку знайти пакет numpy.



Скопіюйте цю команду з верхньої частини сторінки та запустіть її у терміналі.



Після цього ви маєте бачити повідомлення про успішну інсталяцію пакету numpy та його dependencies (залежностей - пакетів).

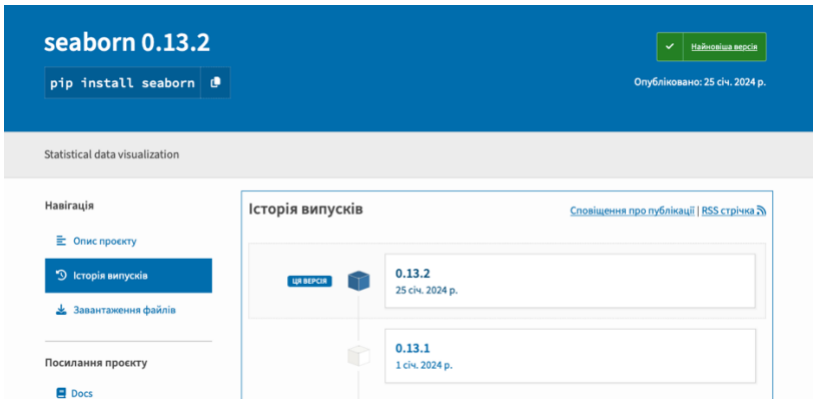
4.b. Встановлення конкретної версії пакету (рекомендований спосіб для подальшого використання).
Тепер знайдіть у pypi бібліотеку pandas, в історії версій (релізів) знайдіть **передостанню** версію та введіть у терміналі команду, щоб встановити його з відповідною версією:
Для Windows:

```
python -m pip install "SomeProject==1.4"
або
py -m pip install "SomeProject==1.4"
```

Для Unix/MacOS:

```
python3 -m pip install "SomeProject==1.4"
```

де SomeProject – назва бібліотеки для інсталювання, == це визначення для того, яка конкретна версія потрібна і 1.4 – це цифри, що відповідають номерам версії для встановлення. Наприклад,




```
python -m pip install "seaborn==0.13.1"
```

Більше про встановлення бібліотек можете прочитати тут: https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/

5. Тепер пропоную вам встановити самостійно додатково пакети matplotlib та pylint, black.
6. Після цього утворимо список з усіма пакетами та їхніми версіями для зручнішої роботи у команді. Зазвичай це робиться через файл requirements.txt або pipfile при роботі з

pipenv. Отже, якщо ви робите цю роботу з pipenv, вам необхідно додати до репозиторію pipfile та pipfile.lock, а при використанні venv – requirements.txt.

Для venv:
Для Windows:

```
python -m pip freeze
або
py -m pip freeze
```

Для Unix/MacOS:

```
python3 -m pip freeze
```

Тепер інші розробники, маючи цей файл можуть автоматично інсталювати всі ті самі пакети та версії за допомогою команди `python -m pip install -r requirements.txt`

Більше про цей файл та випадки використання можна прочитати тут: https://pip.pypa.io/en/latest/user_guide/#requirements-files

7. Зробіть commit з відповідним повідомленням.

4. Робота з файлами.

1. У файлі input.py створіть пусті 3 функції: 1) для вводу тексту з консолі, 2) для зчитування з файлу за допомогою вбудованих можливостей python, 3) для зчитування з файлу за допомогою бібліотеки pandas.
2. У файлі output.py створіть пусті 3 функції: 1) для виводу тексту у консоль, 2) для запису до файлу за допомогою вбудованих можливостей python.
3. Зробіть ще один commit з відповідним повідомленням на цьому кроці.
4. Створіть docstrings для всіх цих функцій.
5. У main.py у функції main() доповніть її тіло викликами виществорених функцій так, щоб текстові результати, що

повертаються функціями 4.1.1) , 4.1.2) та 4.1.3) були виведені у консоль, а також записані до файлу через вбудовані можливості python.

6. Реалізуйте ці функції.
7. За потреби, ви можете створити окрему папку для даних (файлів) у кореневій папці проєкту з назвою data. Обов'язково додайте її до .gitignore.
8. Зробіть commit з відповідним повідомленням.

5. ***(На оцінку 90+). Написання тестів.**
   Використовуючи пакети unittest або pytest на ваш вибір, напишіть по три тести до функцій 2 та 3 (зчитування з файлів) з файлу input.py. Після написання тестів для кожної окремої функції дуже рекомендую робити commit.

   Ресурси, які можуть вам бути корисні:
   https://docs.python.org/3/library/unittest.html
   https://docs.pytest.org/en/7.4.x/getting-started.html

   https://realpython.com/python-testing/
   https://www.dataquest.io/blog/unit-tests-python/

6. **Висновки.**

a. Що зробили?
Попрактикувалися у роботі з pipenv, вивчили особливості структури проєкту, закріпили навички роботи з докстрінгами та у створенні функцій. Навчилися на практиці вводити і виводити дані із файлів, тобто ще міцніше засвоїли роботу з бібліотеками нампай та пандочок.

b. Що нового дізнались для себе?
Цікаво було попрацювати з pipenv, дуже нагадало те, як це влаштовано в расті, максимально схожа механіка з однаковою логікою застосування.

c. Що було корисним? Що б Ви використали в майбутньому?
Ну тестики мастхев і різні структури проєктів, тепер я не буду їх боятися! Дякую Вам!

d. Що можна було б покращити нам для студентів в цій роботі?
Я не зрозумів виділене формулювання:(
Створити ще один Python Package і назвати його « tests». Це є директорія, що містить unit тести, та буде дзеркальною для app (тобто , файл test_input.py у tests відповідатиме файлу input.py у app, і те саме для піддиректорії **io у app та test_io у tests** і т.д. д.).

8. **Наостанок.**

Похваліть себе, Ви дуже багато зусиль доклали! Побалуйте себе відпочинком або якимось смаколиком.

Дякую, що доклали зусиль, у Вас вийшло!