

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

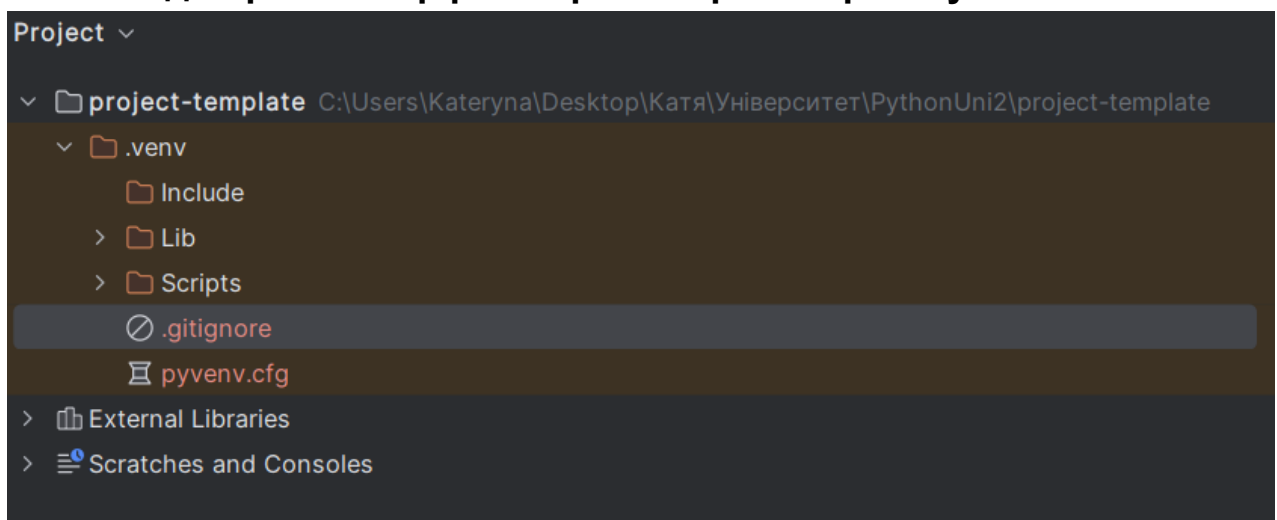
Катерина Братюк, 3 група

<https://github.com/katernabratiuk/Python-for-Big-Data-and-Data-Science-p.2>

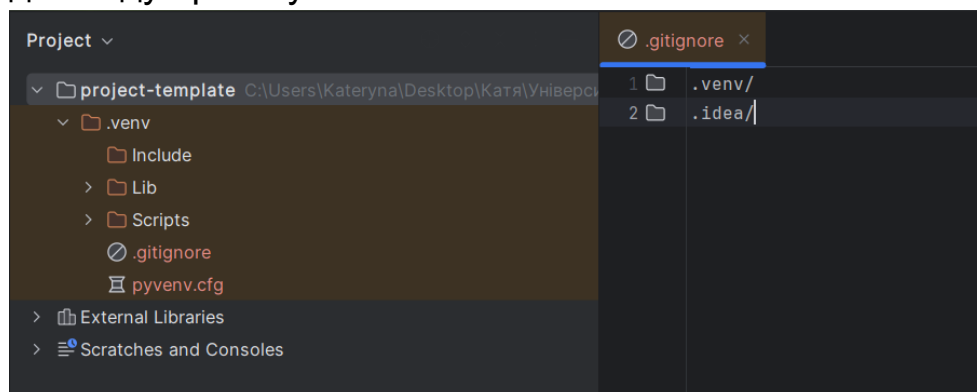
1. Створення нового проєкту.

- 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 рірenv самостійно на оцінку 80-90**), main.py створимо пізніше. Назву папки віртуального середовища запам'ятайте, ми використаємо її пізніше.

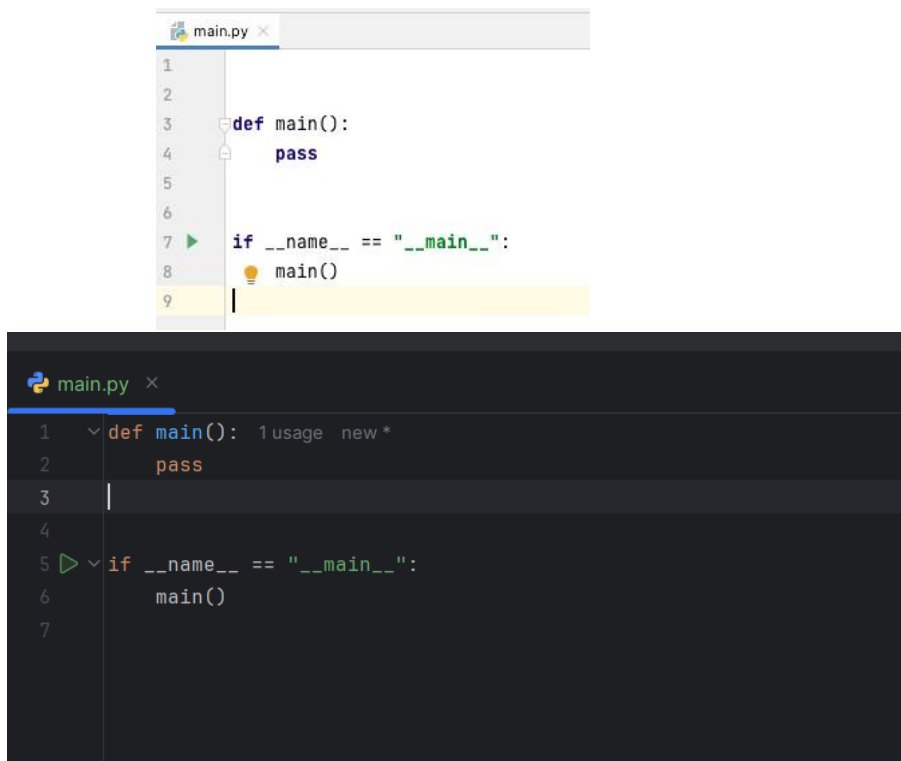
***для роботи з рірenv при створенні проєкту**



- b. Підготуйте файл .gitignore, щоб папки типу venv або .idea і.т.п. не потрапили до репозиторію, який призначений суто для коду проєкту.



- с. Створіть файл `main.py` у директорії проєкту, який матиме наступний вигляд:



```
1
2
3 def main():
4     pass
5
6
7 if __name__ == "__main__":
8     main()
9
```

```
1  def main(): 1 usage  new *
2      pass
3
4
5  if __name__ == "__main__":
6      main()
7
```

- d. Створіть також новий репозиторій на GitHub (теж підглянути, як це робиться, можете у ПЗ 1).
- е. 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.

- а. 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).

- б. У середині 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 ріреnv, 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 менеджера 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.

Опис проєкту



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



```
Kateryna@katebratiuk MINGW64 ~/Desktop/Катя/Університет/PythonUni2/project-template/app/io (main)
$ pip install numpy
Requirement already satisfied: numpy in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (2.0.1)

[notice] A new release of pip is available: 24.3.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip

Kateryna@katebratiuk MINGW64 ~/Desktop/Катя/Університет/PythonUni2/project-template/app/io (main)
$ python.exe -m pip install --upgrade pip
```

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 рурі бібліотеку pandas, в історії версій (релізів) знайдіть **передостанню** версію та введіть у терміналі команду, щоб встановити його з відповідною версією:

Для Windows:

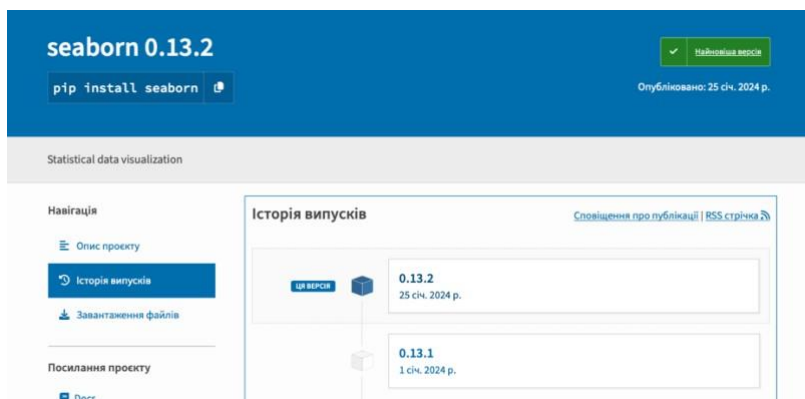
```
python -m pip install "SomeProject==1.4" або
```

```
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-andvirtual-environments/>

```
kateryna@katebratiuk MINGW64 ~/Desktop/Катя/Університет/PythonUni2/project-template/app/io (main)
$ pip install "pandas==2.2.2"
Requirement already satisfied: pandas==2.2.2 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from pandas==2.2.2) (2.0.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from pandas==2.2.2) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from pandas==2.2.2) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from pandas==2.2.2) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from python-dateutil>=2.8.2->pandas==2.2.2) (1.16.0)
```

5. Тепер пропоную вам встановити самостійно додатково пакети matplotlib та pylint, black.

```
$ pip install "matplotlib==3.10.1"
Collecting matplotlib==3.10.1
  Downloading matplotlib-3.10.1-cp312-cp312-win_amd64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib==3.10.1)
  Downloading contourpy-1.3.1-cp312-cp312-win_amd64.whl.metadata (5.4 kB)
Collecting cycler>=0.10 (from matplotlib==3.10.1)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib==3.10.1)
  Downloading fonttools-4.56.0-cp312-cp312-win_amd64.whl.metadata (103 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib==3.10.1)
  Downloading kiwisolver-1.4.8-cp312-cp312-win_amd64.whl.metadata (6.3 kB)
Requirement already satisfied: numpy>=1.23 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from matplotlib==3.10.1) (2.0.1)
Requirement already satisfied: packaging>=20.0 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from matplotlib==3.10.1) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from matplotlib==3.10.1) (10.4.0)
Collecting pyparsing>=2.3.1 (from matplotlib==3.10.1)
  Downloading pyparsing-3.2.3-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from matplotlib==3.10.1) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from python-dateutil>=2.7->matplotlib==3.10.1) (1.16.0)
Downloading matplotlib-3.10.1-cp312-cp312-win_amd64.whl (8.1 MB)
  8.1/8.1 MB 31.2 MB/s eta 0:00:00
Downloading contourpy-1.3.1-cp312-cp312-win_amd64.whl (220 kB)
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.56.0-cp312-cp312-win_amd64.whl (2.2 MB)
  2.2/2.2 MB 60.2 MB/s eta 0:00:00
Downloading kiwisolver-1.4.8-cp312-cp312-win_amd64.whl (71 kB)
Downloading pyparsing-3.2.3-py3-none-any.whl (111 kB)
Installing collected packages: pyparsing, kiwisolver, fonttools, cycler, contourpy, matplotlib
```

```
Kateryna@katebratiuk MINGW64 ~/Desktop/Катя/Університет/PythonUni2/project-template/app/io (main)
$ pip install "pylint==3.3.6"
Collecting pylint==3.3.6
  Downloading pylint-3.3.6-py3-none-any.whl.metadata (12 kB)
Collecting astroid<=3.4.0.dev0,>=3.3.8 (from pylint==3.3.6)
  Downloading astroid-3.3.9-py3-none-any.whl.metadata (4.5 kB)
Requirement already satisfied: colorama>=0.4.5 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from pylint==3.3.6) (0.4.6)
Collecting dill>=0.3.6 (from pylint==3.3.6)
  Downloading dill-0.3.9-py3-none-any.whl.metadata (10 kB)
Collecting isort!=5.13.*>=4.2.5 (from pylint==3.3.6)
  Downloading isort-6.0.1-py3-none-any.whl.metadata (11 kB)
Collecting mccabe<0.8,>=0.6 (from pylint==3.3.6)
  Downloading mccabe-0.7.0-py2.py3-none-any.whl.metadata (5.0 kB)
Collecting platformdirs>=2.2 (from pylint==3.3.6)
  Downloading platformdirs-4.3.7-py3-none-any.whl.metadata (11 kB)
Collecting tomlkit>=0.10.1 (from pylint==3.3.6)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Downloading pylint-3.3.6-py3-none-any.whl (522 kB)
Downloading astroid-3.3.9-py3-none-any.whl (275 kB)
Downloading dill-0.3.9-py3-none-any.whl (119 kB)
Downloading isort-6.0.1-py3-none-any.whl (94 kB)
Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
Downloading platformdirs-4.3.7-py3-none-any.whl (18 kB)
Downloading tomlkit-0.13.2-py3-none-any.whl (37 kB)
Installing collected packages: tomlkit, platformdirs, mccabe, isort, dill, astroid, pylint
Successfully installed astroid-3.3.9 dill-0.3.9 isort-6.0.1 mccabe-0.7.0 platformdirs-4.3.7 pylint-3.3.6 tomlkit-0.13.2
```

```
Kateryna@katebratiuk MINGW64 ~/Desktop/Катя/Університет/PythonUni2/project-template/app/io (main)
$ pip install "black==25.1.0"
Collecting black==25.1.0
  Downloading black-25.1.0-cp312-cp312-win_amd64.whl.metadata (81 kB)
Collecting click>=8.0.0 (from black==25.1.0)
  Using cached click-8.1.8-py3-none-any.whl.metadata (2.3 kB)
Collecting mypy_extensions>=0.4.3 (from black==25.1.0)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: packaging>=22.0 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from black==25.1.0) (24.1)
Collecting pathspec>=0.9.0 (from black==25.1.0)
  Downloading pathspec-0.12.1-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: platformdirs>=2 in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from black==25.1.0) (4.3.7)
Requirement already satisfied: colorama in c:\users\kateryna\appdata\local\programs\python\python312\lib\site-packages (from click>=8.0.0->black==25.1.0) (0.4.6)
Downloading black-25.1.0-cp312-cp312-win_amd64.whl (1.4 MB)
  1.4/1.4 MB 12.5 MB/s eta 0:00:00
Using cached click-8.1.8-py3-none-any.whl (98 kB)
Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)
Installing collected packages: pathspec, mypy_extensions, click, black
Successfully installed black-25.1.0 click-8.1.8 mypy_extensions-1.0.0 pathspec-0.12.1
```

- Після цього утворимо список з усіма пакетами та їхніми версіями для зручнішої роботи у команді. Зазвичай це робиться через файл requirements.txt або рірfile при роботі з рірenv. Отже, якщо ви робите цю роботу з рірenv, вам необхідно додати до репозиторію рірfile та рірfile.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 проекту з назвою data. Обов'язково додайте її до .gitignore.
8. 8. Зробіть commit з відповідним повідомленням.
Дивитися репозиторій

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.

а. Що зробили?

Освіжила пам'ять щодо правильного створення репозиторію, грамотного написання doctring'ів, грамотного написання тестів.

б. Що нового дізнались для себе?

Як правильно створити Python package, важливість присутності __init__.py файлів.

с. Що було корисним? Що б Ви використали в майбутньому?

Думаю, що все з цієї практичної буде використовуватися в майбутньому, бо це база.

- d. Що можна було б покращити нам для студентів в цій роботі?

Дати якісь конкретніші визначення щодо функцій, які потрібно створити, тому що не дуже зрозуміло, чи це повинні бути більш загальні функції, чи студенту дозволено покреативити.

7. Надсилання звіту.

- a. Готовий звіт прикріпити у Мудл згідно дедлайнів.

8. Наостанок.

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

Дякую, що доклали зусиль, у Вас вийшло!

