REPORT01-LABS

We started our second laboratories with setting up the programming environment. We decided on using miniconda. We installed the package and got to know basic commands. We moved on to activating the environment:

```
conda activate pite
```

After installing miniconda, we moved on to installing virtualenv. We made it by using following commands:

```
sudo apt-get update
sudo apt-get install python3-pip
sudo pip3 install virtualenv
```

And finally we created and activated our virtual environment:

```
virtualenv venv-pite
```

```
source venv-pite/bin/activate
(venv-pite)$ pip install -r requirements.txt
(venv-pite)$ pip install -r requirements.txt --upgrade
```

According to an official website, Jupyter "is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text." We installed Jupyter to have a smart tool for creating documents with usage of computational output, explanatory text and multimedia resources in a single document. We went back to our conda and we installed jupyter:

```
(pite) conda install jupyter
(pite) conda install jupyterlab
(pite) conda install -c anaconda ipykernel
(pite) python -m ipykernel install --user --name=pite
```

And we opened jupyter:

```
(pite) jupyter notebook
```

Next step was to open VS Code and configure the interpreter and jupyter kernel so we could use VS Code connected to the things we had installed in th terminal before.

The next part of labs was spared to analyze the Dunder variables and methods. We learned that "Dunder variables and methods are names that are preceded and succeeded by **double underscores**, hence the name dunder. They are also called special, magic methods and can help override functionality for built-in functions for custom classes." The dunder names are used in python to execute the code inside the 'if' statement only if the program is excecuted directly by the Python interpreter.?

In VS Code we implemented a short block of code to take a look at the dunder expressions in main_name.py file.

```
if __name__ == ?__main__?:
         print(f"The value of __name__ is {__name__}")
```

And the output was: 'The value of __name__ is __main__'.

Next excersise was to run given code:

```
def step1():
    print("Executing step1...")
def step2():
    print("Executing step2...")
def step3():
    print("Executing step3...")


step1()
step2()
step3()
```

We executed it as follows:

```
(pite)$ python main_name.py
>>> import main_name
>>> main_name.step1()
```

And the output was:

Executing step1...
Executing step2...
Executing step3...

We modified the program:

```
def step1():
? ? ?print("Executing step1...")
def step2():
? ? ?print("Executing step2...")
def step3():
? ? ?print("Executing step3...")
if __name__ == "__main__":
  step1()
  step2()
  step3()
```

We ran the code by importing the modul and the program didn't compile. That's because we execute the code inside the if statement only when the program is executed directly by the Python interpreter.?
When the code in the file is imported as a module the code inside the if statement is not executed.

This approach can be useful not only beacuse it is used to increase the readability of the program but also to be sure of validation of used arguments.

The last task was to create a git repository using bitbucket. We created it using simple commands in the terminal:

```
git clone https://agile.fis.agh.edu.pl/bitbucket/scm/pite2122w/pite_common.git
cd pite_common/
mkdir 1_karolina_kotlowska
git checkout -b 1_karolina_kotlowska_lab02
touch comprehension.py
git add --all
git commit -m ?personal branch init?
git push -u origin 1_karolina_kotlowska_lab02
```

We created a pull request and added our colleague on the right as the rewiever.