

# Laboratorium 1 – Wprowadzenie do Python

Języki skryptowe

## Cele dydaktyczne

1. Uruchamianie interpretera Python
2. Uruchamianie prostych aplikacji
3. Instalacja dodatkowych modułów Python
4. Zarządzanie zależnościami

## Wprowadzenie

Python jest językiem skryptowym, którego programy wykonywane są przez interpreter. W uproszczeniu, podstawowym sposobem uruchamiania aplikacji napisanych w tym języku jest przekazanie ścieżki do skryptu jako argument interpreterowi, czyli programowi odpowiedzialnemu za wykonywanie poleceń "linia po linii". W dowolnym momencie można dokonać edycji skryptu i uruchomić program ponownie bez konieczności przebudowania (ponownej kompilacji, linkowania, itd.). Z tego powodu, języki skryptowe są dobrymi narzędziami do takich zagadnień jak prototypowanie, automatyzacja zadań, przetwarzanie danych, etc.

W salach laboratoryjnych w bud. D-2 dostępna jest maszyna wirtualna VirtualBox, na której zainstalowany jest Linux<sup>1</sup>, w którym znajdują się narzędzia niezbędne do wykonania zadań. W salach w bud. B-4 narzędzia deweloperskie powinny być zainstalowane w ramach OS. Niezależnie, poniższe zadania są mogą zostać wykonane w ramach dowolnego systemu operacyjnego dla komputerów klasy PC.

---

<sup>1</sup> użytkownik/hasło: student/student, dostęp do konta root'a przez sudo.

## Zadania

### 1. Uruchamianie interpretera w trybie REPL

Tryb REPL (Read-Eval-Print Loop) pozwala na uruchomienie interpretera w trybie powłoki. Po uruchomieniu, użytkownik wprowadza wyrażenia do ewaluacji (po symbolu zachęty `>>>`), które następnie są ewaluowane przez interpreter, a wynik ewaluacji jest wyświetlany.

- Aby poznać wersję interpretera Python dostępną w aktualnej ścieżce, uruchom konsolę/wiersz poleceń, a następnie wpisz: `python --version`.
- Uruchom Interpreter Pythona w trybie REPL i skonstruuj program, wypisujący napis "Hello world!".

```
python
```

```
>>> print("Hello world!")
```

- Wykorzystaj zmienną, aby zmodyfikować komunikat.

```
>>> name = "Python"
```

```
>>> print("Hello, {0}!".format(name))
```

```
>>> print(f"Hello, {name}!")
```

- Wykonaj kilka działań arytmetycznych, poprzez skonstruowanie wyrażeń z wykorzystaniem operatorów arytmetycznych, np:

- dodawanie ( `+` ), odejmowanie ( `-` ), mnożenie ( `*` ),
- dzielenie ( `/` ), dzielenie bez reszty ( `//` ), reszta z dzielenia ( `%` ),
- potęgowanie ( `**` ).

- Następnie, skonstruuj kilka wyrażenie arytmetycznych wykorzystujących symbol `_` jako jeden z operandów, np.

```
>>> _ * 2.0
```

```
>>> _ + 1
```

Zastanów się, jaką wartość reprezentuje ten symbol.

- Zakończ sesję interaktywną wywołaniem wyrażeniem `exit()`.

### 2. Uruchamianie notesów Jupyter.


Jupyter Notebook – graficzne narzędzie uruchamiane w przeglądarce internetowej, które pozwala na mieszanie tekstów ze skryptami w języku Python, pozwalające na tworzenie dokumentów z obliczalnymi elementami (np. wizualizacjami).

- a. Zainstaluj Jupyter Notebook wpisując w terminalu/wierszu poleceń.

```
pip install notebook
```

- b. Uruchom Jupyter notebook.

```
jupyter notebook
```

- c. Kliknij przycisk listy rozwijanej  z prawej strony i wybierz opcję "Python 3 (ipykernel)".  
d. Wprowadź poniższy kod do pola tekstowego obok napisu In [ ] :.


```
name = "World"  
  
print(f"Hello, {name}")
```

Wciśnij przycisk "Run", w celu uruchomienia kodu.

- e. W kolejnym polu tekstowym, wpisz

```
# Moja aplikacja w języku Python  
Oto mój pierwszy notes Jupyter.
```

Korzystając z listy rozwijanej umieszczonej pod menu, zmień typ bloku z Code na Markdown.  
Wciśnij przycisk "Run".

- f. Wciśnij przycisk zapisu  . Przeanalizuj zawartość pliku \*.ipynb w edytorze tekstowym.

### 3. Uruchamianie skryptów w konsoli.

- a. Uruchom wybrane IDE (np. VSCode).  
b. Utwórz plik `app.py` z następującą treścią:

```
name = input("Enter your name: ")  
  
print(f'Hello, {name}!')
```

- c. Uruchom plik z poziomu terminala/wiersza poleceń przy pomocy komendy:
- ```
python app.py
```

#### 4. Prosty program w Python.

Napisz program w języku Python, który obliczy pole trójkąta o wysokości i długości podstawy zadanych przez użytkownika. Aby dokonać konwersji pomiędzy napisem `s` a liczbą, należy wykorzystać funkcję `int()`.

#### 5. Korzystanie z menedżera zależności

Pozwa bogatą biblioteką standardową, Python posiada bogate repozytorium modułów i narzędzi: [The Python Package Index \(PyPI\)](https://pypi.org/).

Moduły można instalować z terminala/wiersza poleceń przy użyciu polecenia `pip`, tak jak robiono to w poprzednich zadaniach. Po zainstalowaniu modułu, w skrypcie należy go zaimportować przy użyciu instrukcji `import`. Dla przykładu, dla modułu `pyfiglet`, pozwalającego na generowanie ASCII-art z tekstu:

```
pip install pyfiglet

...

>>> import pyfiglet

>>> pyfiglet.print_figlet("Języki skryptowe")
```

W przypadku pracy nad większymi projektami, aby ułatwić zarządzanie modułami oraz ich wersjami, warto wykorzystać do tego odpowiednie narzędzie. Na przykład:

- Venv + pip <https://docs.python.org/3/library/venv.html>
- Poetry <https://python-poetry.org>
- Conda <https://conda.io/en/latest/>
- Pipenv <https://pipenv.pypa.io/en/latest/>

Zapoznaj się z dokumentacją wybranego narzędzia. Następnie, z jego wykorzystaniem utwórz program, który będzie zależny od modułu `pyfiglet`. Niech program przyjmuje od użytkownika

tekst na wejściu standardowym, a na wyjściu zwraca ASCII art z tym tekstem.