Język Python w wizualizacji i symulacji

Maciej Hojda

Uwaga: kod przeklejony bezpośrednio z pdf-a może stwarzać problemy (tabulacja i apostrofy).

1 Zadanie nr 1 – instalacja oprogramowania

Należy zainstalować:

1. bibliotekę Visual C++
2. język Python 3.12.2,
3. zintegrowane środowisko PyCharm (community edition),
4. podstawowe biblioteki.

System operacyjny: Windows 10 64bit.

1.1 Biblioteka Visual C++

1. Ze strony
   `https://learn.microsoft.com/en-US/cpp/windows/latest-supported-vc-redist?view=msvc-170`
2. Pobierz i zainstaluj `vc_redist.x64`
   `https://aka.ms/vs/17/release/vc_redist.x64.exe`

1.2 Python 3.12.2

1. Ze strony
   `https://www.python.org/downloads/release/python-3122/`
   pobierz i zainstaluj `Windows x86-64 executable installer`
   (`https://www.python.org/ftp/python/3.12.2/python-3.12.2-amd64.exe`)
2. Opcje instalacji
   - Use admin priviliges when installing py.exe
   - Add python.exe to PATH

1.3 Pycharm community edition

1. Ze strony
   `https://www.jetbrains.com/pycharm/download/?section=windows#section=windows`
   pobierz i zainstaluj PyCharm community edition

1.4 Biblioteki

1. Uruchom PyCharma (jest już w „menu start”)
2. Utwórz nowy projekt `New project`
   Parametry:
   `Pure Python`
   `Name: [wybierz]`
   `Interpreter: Project venv`
   `Python version: [wybierz zainstalowaną]`
3. W konsoli wpisz (pomiń `>>`):
   ```
   >> import pip
   >> pip.main(['install','numpy'])
   >> pip.main(['install','matplotlib'])
   >> pip.main(['install','networkx'])
   ```
   Zignoruj ostrzeżenia (ale nie błędy).
   Konsola typowo znajduje się na lewym pasku narzędziowym.

Te trzy zainstalowane biblioteki dodatkowe pozwalają na, kolejno

- (`numpy`) pracę z macierzami
  `https://numpy.org/doc/stable/user/index.html#user`
- (`matplotlib`) wyświetlanie wykresów
  `https://matplotlib.org/stable/api/index.html`
- (`networkx`) wyświetlanie grafów
  `https://networkx.org/documentation/stable/reference/introduction.html`

2 Zadanie nr 2 – wyświetlanie wykresów

W projekcie z

1. Utwórz plik o nazwie `main.py` w menu `File > New...`
   Plik umieść w podkatalogu `.venv`
2. W utworzonym pliku umieść kod jak następuje

   ```
   import matplotlib.pyplot as plt

   _, axes = plt.subplots()
   axes.plot([0, 1, 2, 3, 4, 5, 6], [0, 2, 4, 6, 4, 2, 0]);
   plt.show();
   ```

3. Uruchom utworzony program `Run > Run 'main'`.
4. Zaobserwuj wykres w postaci trójkąta.
5. Przeanalizuj kod śledząc komentarze (`#` symbolizuje komentarz).

   ```
   # zaimportuj bibliotekę 'pyplot' z pakietu 'matplotlib'
   # nazwij ją 'plt'
   # dalej można z niej korzystać w kodzie pod nazwą 'plt'
   import matplotlib.pyplot as plt

   # utwórz wykres
   ```

```
# funkcja 'plt.subplots' zwraca dwa argumenty
# pierwszy ignorujemy, drugi zapisujemy do zmiennej 'axes'
_, axes = plt.subplots()

# tworzymy wykres z punktów (0, 0), (1, 2), (2, 4), itd.
axes.plot([0, 1, 2, 3, 4, 5, 6], [0, 2, 4, 6, 4, 2, 0]);

# wyświetlamy wykres
plt.show();
```

3 Zadanie nr 3 – wyświetlanie grafów

1. Utwórz (`File > New...`) i uruchom nowy program (`Run > Run...`).

```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_edge('A', 'B')
G.add_edge('B', 'D')
G.add_edge('A', 'C')
G.add_edge('C', 'D')

pos = nx.spring_layout(G)
nx.draw_networkx_nodes(G, pos, node_size = 500)
nx.draw_networkx_labels(G, pos)
nx.draw_networkx_edges(G, pos)
plt.show()
```

2. Przeanalizuj kod śledząc komentarze.

```
# zaimportuj bibliotekę 'networkx' do rysowania grafów
# nazwij ją 'nx'
# dalej można z niej korzystać w kodzie pod nazwą 'nx'
import networkx as nx

# zaimportuj bibliotekę 'pyplot' z pakietu 'matplotlib'
# nazwij ją 'plt'
# dalej można z niej korzystać w kodzie pod nazwą 'plt'
import matplotlib.pyplot as plt

# utwórz obiekt reprezentujący graf 'nx.Graph()'
# przypisz go do zmiennej 'G'
G = nx.Graph()

# wykorzystaj funkcję 'add_edge' obiektu/grafu 'G'
# funkcja dodaje do grafu krawędź między dwoma wierzchołkami
# nazwy wierzchołków podane są w argumentach funkcji
G.add_edge('A', 'B')
G.add_edge('B', 'D')
G.add_edge('A', 'C')
G.add_edge('C', 'D')

# wybierz typ układu wierzchołków 'spring_layout'
```

```
    # przypisz go do zmiennej 'pos'
    pos = nx.spring_layout(G)

    # wyświetl wierzchołki
    # wierzchołki są w pozycjach zadanych przez 'pos'
    # wierzchołki mają rozmiar '500'
    nx.draw_networkx_nodes(G, pos, node_size = 500)

    # wyświetl etykiety wierzchołków
    # etykiety są w pozycjach zadanych przez 'pos'
    nx.draw_networkx_labels(G, pos)

    # wyświetl krawędzie grafu
    # wierzchołki są w pozycjach zadanych przez 'pos'
    nx.draw_networkx_edges(G, pos)

    # wyświetl graf
    plt.show()
```

4 Zadanie nr 4 – graf ważony

Uruchom (i przeanalizuj) następujący program

```
import networkx as nx
import matplotlib.pyplot as plt

# do operacji pierwiastkowania
import numpy as np

G = nx.Graph()

# nazwy wierzchołków
VV = [1, 2, 3, 4, 5]

# lista krawędzi
WW = [(1, 2), (2, 3), (3, 4), (4, 5), (1, 3), (3, 5)]

# słownik, pozycje wierzchołków
Vx = {1:-5, 2:1, 3:2, 4:3, 5:4}
Vy = {1:0, 2:1, 3:0, 4:-1, 5:0}

g = nx.Graph();

# pusty słownik
gpos = {};

# wypełnienie słownika wierzchołkami
# pętla for przechodzi przez wszystkie elementy 'VV'
for v in VV:
  g.add_node(v);
  gpos[v] = [Vx[v], Vy[v]]

# zagnieżdżone pętle for
for v1 in VV:
```

```
  for v2 in VV:
    # sprawdzenie czy krawędź istnieje w 'WW'
    if (v1, v2) in WW:
      # jeśli istnieje, to ustaw etykietę na odległość euklidesową
      # funkcja 'str' zwraca ciąg znaków
      # funkcja 'np.sqrt' zwraca pierwiastek
      # symbol '**' oznacza potęgowanie
      label = str(np.sqrt((Vx[v1] - Vx[v2])**2 + (Vy[v1] - Vy[v2])**2))
      # dodaj wagi do krawędzi
      g.add_weighted_edges_from([(v1, v2, label)])

# wyświetl żółte wierzchołki z etykietami w ustalonych wcześniej pozycjach
nx.draw(g, gpos, with_labels=True, node_color='yellow')

# pobierz i wyświetl etykiety
labels = nx.get_edge_attributes(g, 'weight')
nx.draw_networkx_edge_labels(g, gpos, edge_labels=labels)

# wyświetl graf
plt.show()
```

5 Zadanie nr 5 – wykresy złożone

Uruchom i przeanalizuj program.

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 20, 10)

_, ax = plt.subplots(figsize=(10, 5))
ax.plot(x, x, color='black', label='y=x')
ax.plot(x, np.sin(x), label='y=sin(x)')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_title("Wykres")
ax.legend()
plt.show()
```

Zmodyfikuj program

- wygładź funkcję $\sin(x)$,
- dodaj funkcje $\cos(x), x^3$,
- zapytaj się użytkownika o przedział x (skorzystaj np. z funkcji `input`),
- zapytaj się użytkownika o kolor wykresu,
- zapytaj się użytkownika, wykresy których funkcji chce wyświetlić.