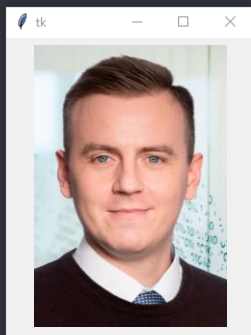


```
1
2
3  Praktyczne_zastosowanie_Pythona = {
4
5      w_naukach : ['biologicznych', 'i medycznych'],
6      dla :      ['początkujących'],
7      poziom :   ['drugi'],
8      część :    ['pierwsza'],
9
10
11      # Mateusz Dobrychłop, 14 listopada 2023
12  }
```

```
1  
2  
3 Prowadzący = {  
4  
5  
6  
7  
8  
9  
10  
11  
12 }  
13  
14
```



```
name      : 'Mateusz Dobrychłop',  
e-mail    : 'mateusz.dobrychlop@gmail.com',  
linkedin  : 'linkedin.com/in/mdobrychlop',
```

Prowadzący = {



```
1 harmonogram_listopad = {
```

```
2     M    T    W    T    F    S    S
```

```
3         01  02  03  04  05
```

```
4         06  07  08  09  10  11  12
```

```
5         13  14  15  16  17  18  19     Spotkanie 1 (17:00 – 19:30)
```

```
6         20  21  22  23  24  25  26     Spotkanie 2 (17:00 – 19:30)
```

```
7         27  28  29  30     Spotkanie 3 (17:00 – 19:30)
```

```
8     }
```

Poziom 1: co wiemy {

Środowisko
pracy

Tryb interaktywny, pliki *.py,
Sublime Text, linia komend

Absolutne
podstawy

Podstawowe typy danych,
definiowanie zmiennych, działania
na łańcuchach i liczbach

Sekwencje
i pętle

Listy, słowniki, pętle *while* i *for*

Źródła
danych

Odczyt i zapis plików tekstowych,
odczyt arkuszy Excela

}

Poziom 2: czego się dowiemy {

Środowisko
pracy

Środowiska wirtualne

Podstawy
Pythona c.d.

F-stringi, składanie list,
definiowanie własnych funkcji

Wizualizacja
danych

Wykresy z wykorzystaniem *matplotlib*

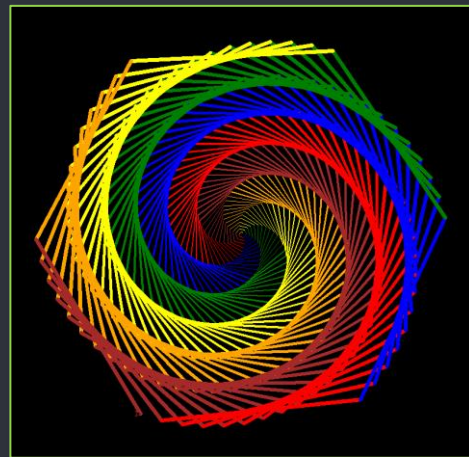
Analiza
danych

Podstawy analizy danych,
rozwinięcie *pandas*, modele językowe
w pracy z kodem

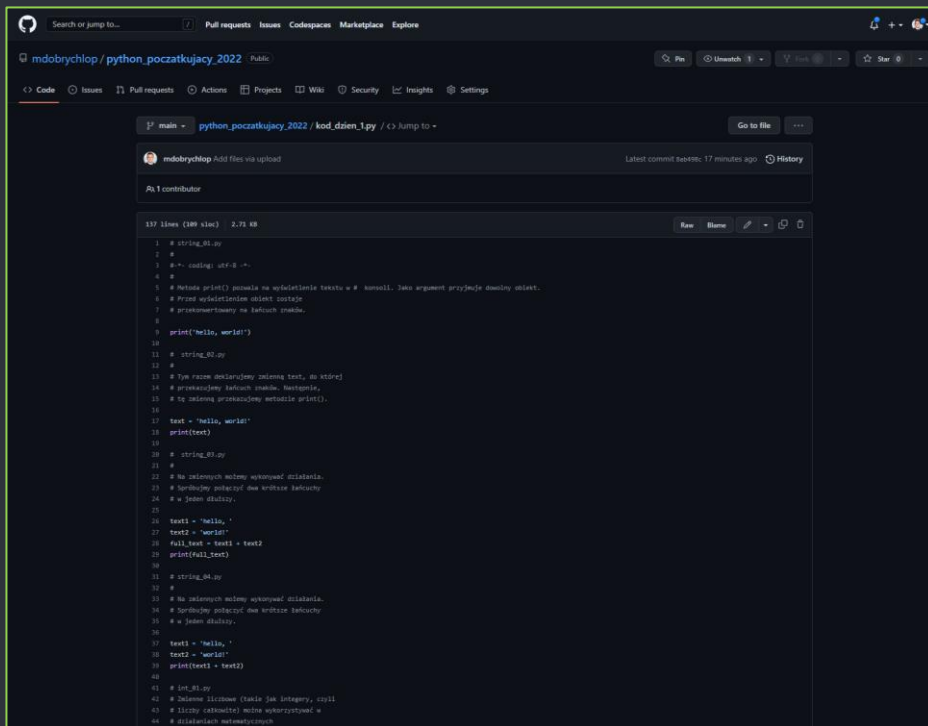
}

```
1 Spotkanie_01 = {
2
3     01 środowiska wirtualne
4
5
6
7     02 f-stringi, listy składane
8
9
10
11     03 definiowanie funkcji
12
13 }
14
```

```
1  -*- coding: utf-8 -*-
2  # W pierwszych liniach kodu znajdują się komentarze.
3
4  import turtle as t
5  import time
6
7  mycolors = ["red", "blue", "green", "yellow", "orange", "brown"]
8
9  t.pensize(5)
10
11 t.bgcolor('black')
12
13 t.speed(1000)
14
15 for x in range(360):
16     t.pencolor(mycolors[x % len(mycolors)])
17     t.pensize(x / 50)
18     t.forward(x)
19     t.left(59)
20
21 time.sleep(5)
22
23
24
25
```



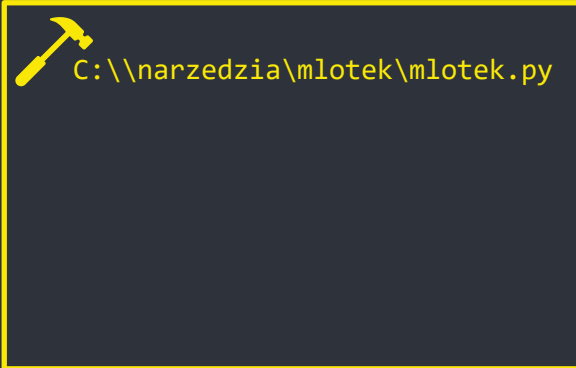
https://github.com/mdobrychlop/python_poczatkujacy_lvl2_2023/



```
1 # string_01.py
2 #
3 # -*- coding: utf-8 -*-
4 #
5 # Metoda print() służy do wyświetlenia tekstu w # konsoli. Jako argument przyjmuje dowolny obiekt.
6 # Przed wyświetleniem obiekt musi być
7 # przekonwertowany na łańcuch znaków.
8
9 print("hello, world!")
10
11 # string_02.py
12 #
13 # W tym rozdziale przedstawiamy tekst, do którego
14 # przekształcamy łańcuch znaków. Następnie,
15 # to łańcuch przekształcamy metodą print().
16
17 test = "hello, world!"
18 print(test)
19
20 # string_03.py
21 #
22 # Na zmiennych możemy wykonać działania.
23 # Np. dodajemy teksty do zmiennej test
24 # w jednej linii.
25
26 test1 = "hello, "
27 test2 = "world!"
28 full_test = test1 + test2
29 print(full_test)
30
31 # string_04.py
32 #
33 # Na zmiennych możemy wykonać działania.
34 # Np. dodajemy teksty do zmiennej test
35 # w jednej linii.
36
37 test1 = "hello, "
38 test2 = "world!"
39 print(test1 + test2)
40
41 # int_01.py
42 # Działania liczbowe (takie jak integer, czyli
43 # liczby całkowite) można wykonać w
44 # działaniach matematycznych
```

virtualenv.py

czesc_druga.py



virtualenv.py

czesc_druga.py



```
C:\\narzedzia\\mlotek\\mlotek.py  
pandas == 1.5.3?
```



virtualenv.py

czesc_druga.py



C:\\Python3\\python.exe

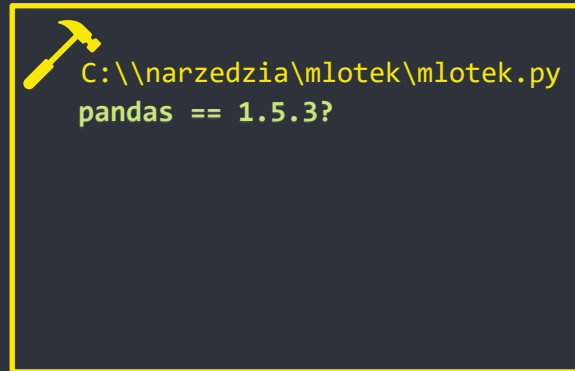
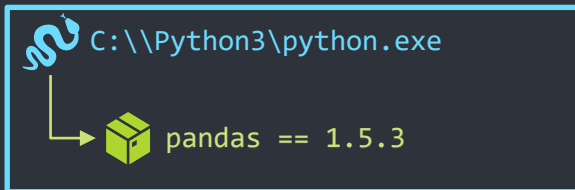


C:\\narzedzia\\mlotek\\mlotek.py
pandas == 1.5.3?




virtualenv.py

czesc_druga.py




virtualenv.py


czesc_druga.py



```
C:\\Python3\\python.exe  
pandas == 1.5.3
```

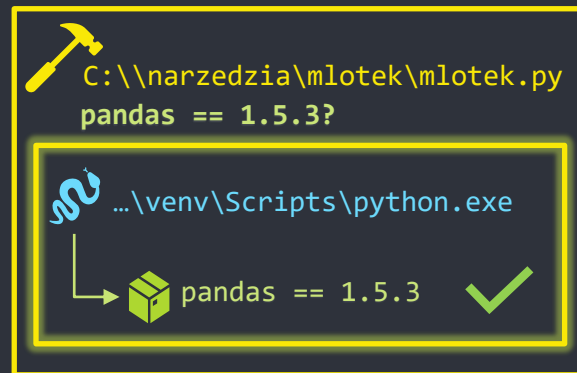
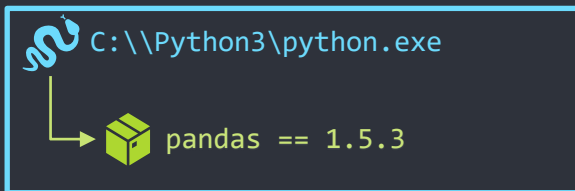


```
C:\\narzedzia\\pila\\pila.py  
pandas == 2.1.2?
```



```
C:\\narzedzia\\mlotek\\mlotek.py  
pandas == 1.5.3?
```





1. Tworzymy folder, w którym będziemy przechowywać skrypty z zajęć
2. Otwieramy linię komend (cmd, PowerShell, terminal itp.)
3. Przechodzimy do folderu z punktu 1.
4. Instalujemy *virtualenv* za pomocą komendy:

```
pip install virtualenv
```

5. Następnie tworzymy lokalne środowisko wirtualne o nazwie *venv*:

```
virtualenv venv
```

6. Następnie aktywujemy środowisko wirtualne:



```
.\venv\Scripts\activate.ps1
```



```
source venv/bin/activate
```


1. Tworzymy folder, w którym będziemy przechowywać skrypty z zajęć
2. Otwieramy linię komend (cmd, PowerShell, terminal itp.)
3. Przechodzimy do folderu z punktu 1.
4. Instalujemy *virtualenv* za pomocą komendy:

```
pip install virtualenv
```

5. Następnie tworzymy lokalne środowisko wirtualne o nazwie *venv*:

```
virtualenv venv
```

6. Następnie aktywujemy środowisko wirtualne:



```
.\venv\Scripts\activate.ps1
```



```
source venv/bin/activate
```

```
Windows PowerShell
PS C:\Users\Dobry\Documents\szkolenie2023> virtualenv venv
created virtual environment CPython3.10.6.final.0-64 in 9329ms
creator CPython3Windows(dest=C:\Users\Dobry\Documents\szkolenie2023\venv, clear
=False, no_vcs_ignore=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle,
via=copy, app_data_dir=C:\Users\Dobry\AppData\Local\pypa\virtualenv)
added seed packages: pip==23.2.1, setuptools==68.2.2, wheel==0.41.2
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShe
llActivator,PythonActivator
PS C:\Users\Dobry\Documents\szkolenie2023> .\venv\Scripts\activate.ps1
(venv) PS C:\Users\Dobry\Documents\szkolenie2023>
```

```
3  # Załóżmy, że przeprowadziliśmy serię eksperymentów
4  # i chcemy sformatować ich wyniki w czytelny sposób.
5
6  nazwa_eksperymentu = "Test wpływu pH na aktywność enzymu"
7  pH = 7.4
8  aktywnosc_enzymu = 0.6752 # Aktywność enzymu jako wartość względna
9
10 # Aktywnosc_enzymu zaokrąglona do dwóch miejsc po przecinku i przekonwertowana na procenty:
11 aktywnosc_enzymu = round(aktywnosc_enzymu * 100, 2)
12
13
14 # Wypisanie wyników w konsoli
15 opis_wynikow = "Eksperyment: " + nazwa_eksperymentu + "; " + "przy pH: " + str(pH) + ", " + \
16 |         |         |         |         | "zaobserwowano aktywność enzymu na poziomie: " + str(aktywnosc_enzymu) + "%."
17
18 print(opis_wynikow)
19
```

```
3  # Załóżmy, że przeprowadziliśmy serię eksperymentów
4  # i chcemy sformatować ich wyniki w czytelny sposób.
5
6  nazwa_eksperymentu = "Test wpływu pH na aktywność enzymu"
7  pH = 7.4
8  aktywnosc_enzymu = 0.6752  # Aktywność enzymu jako wartość względna
9
10 # Aktywnosc_enzymu zaokrąglona do dwóch miejsc po przecinku i przekonwertowana na procenty:
11 aktywnosc_enzymu = round(aktywnosc_enzymu * 100, 2)
12
13
14 # Wypisanie wyników w konsoli
15 opis_wynikow = "Eksperyment: " + nazwa_eksperymentu + "; " + "przy pH: " + str(pH) + ", " + \
16 |         |         |         | "zaobserwowano aktywność enzymu na poziomie: " + str(aktywnosc_enzymu) + "%."
17
18 print(opis_wynikow)
19
```

Eksperyment: Test wpływu pH na aktywność enzymu; przy pH: 7.4, zaobserwowano aktywność enzymu na poziomie: 67.52%.

```
1  # Załóżmy, że przeprowadziliśmy serię eksperymentów
2  # i chcemy sformatować ich wyniki w czytelny sposób.
3
4  nazwa_eksperymentu = "Test wpływu pH na aktywność enzymu"
5  pH = 7.4
6  aktywnosc_enzymu = 0.675 # Aktywność enzymu jako wartość względna
7
8  # Użycie f-stringa do sformatowania opisu
9  opis_wynikow = (f"Eksperyment: {nazwa_eksperymentu}; "
10 |               f"przy pH: {pH}, "
11 |               f"zaobserwowano aktywność enzymu na poziomie: {aktywnosc_enzymu:.2%}.")
12
13  print(opis_wynikow)
14
```

```
1  # Załóżmy, że przeprowadziliśmy serię eksperymentów
2  # i chcemy sformatować ich wyniki w czytelny sposób.
3
4  nazwa_eksperymentu = "Test wpływu pH na aktywność enzymu"
5  pH = 7.4
6  aktywnosc_enzymu = 0.675 # Aktywność enzymu jako wartość względna
7
8  # Użycie f-stringa do sformatowania opisu
9  opis_wynikow = (f"Eksperyment: {nazwa_eksperymentu}; "
10                 f"przy pH: {pH}, "
11                 f"zaobserwowano aktywność enzymu na poziomie: {aktywnosc_enzymu:.2%}.")
12
13  print(opis_wynikow)
14
```

Eksperyment: Test wpływu pH na aktywność enzymu; przy pH: 7.4, zaobserwowano aktywność enzymu na poziomie: 67.52%.

```
1  ilosc_probek = 25
2  srednia = 7.8
3  odchylenie_standardowe = 0.45
4
5  analiza_statystyczna = (f"Przeanalizowano {ilosc_probek} próbek, "
6                          f"średnia wartość pH wynosi: {srednia:.1f}, "
7                          f"przy odchyleniu standardowym: {odchylenie_standardowe:.2f}.")
8
9  print(analiza_statystyczna)
10
```

```
1  ilosc_probek = 25
2  srednia = 7.8
3  odchylenie_standardowe = 0.45
4
5  analiza_statystyczna = (f"Przeanalizowano {ilosc_probek} próbek, "
6                          f"średnia wartość pH wynosi: {srednia:.1f}, "
7                          f"przy odchyleniu standardowym: {odchylenie_standardowe:.2f}.")
8
9  print(analiza_statystyczna)
10
```

Przeanalizowano 25 próbek, średnia wartość pH wynosi: 7.8, przy odchyleniu standardowym: 0.45.

```
1  # Lista wartości pH dla różnych próbek wodnych.
2  wartosci_pH = [6.5, 7.2, 7.0, 8.3, 6.8, 4.9, 7.3, 6.1, 5.5, 7.8]
3
4  # Pusta lista, która będzie zawierała wartości kwaśne pH.
5  kwasy_pH = []
6
7  # Użycie pętli for do iteracji przez listę wartości pH i dodawanie do listy tylko kwaśnych wartości.
8  for pH in wartosci_pH:
9      |   if pH < 7:
10     |       |   kwasy_pH.append(pH)
11
12  print(f'Wartości kwaśne pH: {kwasy_pH}')
13
```



```
1  # Lista wartości pH dla różnych próbek wodnych.
2  wartosci_pH = [6.5, 7.2, 7.0, 8.3, 6.8, 4.9, 7.3, 6.1, 5.5, 7.8]
3
4  # Pusta lista, która będzie zawierała wartości kwaśne pH.
5  kwasy_pH = []
6
7  # Użycie pętli for do iteracji przez listę wartości pH i dodawanie do listy tylko kwaśnych wartości.
8  for pH in wartosci_pH:
9      if pH < 7:
10         kwasy_pH.append(pH)
11
12  print(f'Wartości kwaśne pH: {kwasy_pH}')
13
```

Wartości kwaśne pH: [6.5, 6.8, 4.9, 6.1, 5.5]

```
1  # Lista wartości pH dla różnych próbek wodnych.
2  wartosci_pH = [6.5, 7.2, 7.0, 8.3, 6.8, 4.9, 7.3, 6.1, 5.5, 7.8]
3
4  # Użycie list comprehension do wyodrębnienia wartości kwaśnych (pH < 7).
5  kwasy_pH = [pH for pH in wartosci_pH if pH < 7]
6
7  print(f'Wartości kwaśne pH: {kwasy_pH}')
8
```

```
1  # Lista wartości pH dla różnych próbek wodnych.
2  wartosci_pH = [6.5, 7.2, 7.0, 8.3, 6.8, 4.9, 7.3, 6.1, 5.5, 7.8]
3
4  # Użycie list comprehension do wyodrębnienia wartości kwaśnych (pH < 7).
5  kwasy_pH = [pH for pH in wartosci_pH if pH < 7]
6
7  print(f'Wartości kwaśne pH: {kwasy_pH}')
8
```

```
for pH in wartosci_pH:
    if pH < 7:
        kwasy_pH.append(pH)
```

```
1  # Lista wartości pH dla różnych próbek wodnych.  
2  wartosci_pH = [6.5, 7.2, 7.0, 8.3, 6.8, 4.9, 7.3, 6.1, 5.5, 7.8]  
3  
4  # Użycie list comprehension do wyodrębnienia wartości kwaśnych (pH < 7).  
5  kwasy_pH = [pH for pH in wartosci_pH if pH < 7]  
6  
7  print(f'Wartości kwaśne pH: {kwasy_pH}')
```


Wartości kwaśne pH: [6.5, 6.8, 4.9, 6.1, 5.5]

```
1 temperature_celsius = [0, 10, 20, 30, 40]
2
3 # Konwersja temperatur z C na F
4 temperature_fahrenheit = [(celsius * 9/5) + 32 for celsius in temperature_celsius]
5
6 print(f'Temperature w Fahrenheitach: {temperature_fahrenheit}')
7
```

```
1 temperature_celsius = [0, 10, 20, 30, 40]
2
3 # Konwersja temperatur z C na F
4 temperature_fahrenheit = [(celsius * 9/5) + 32 for celsius in temperature_celsius]
5
6 print(f'Temperature w Fahrenheitach: {temperature_fahrenheit}')
7
```

Temperature w Fahrenheitach: [32.0, 50.0, 68.0, 86.0, 104.0]

```
1 nazwy = ['Drosophila melanogaster', 'Homo sapiens', 'Escherichia coli']
2 klasy = ['Owad', 'Ssak', 'Bakteria']
3
4 # Tworzenie listy słowników reprezentujących organizmy
5 organizmy = [{'nazwa': nazwa, 'klasa': klasa} for nazwa, klasa in zip(nazwy, klasy)]
6
7 print(f'Lista organizmów: {organizmy}')
8
```



```
1 nazwy = ['Drosophila melanogaster', 'Homo sapiens', 'Escherichia coli']
2 klasy = ['Owad', 'Ssak', 'Bakteria']
3
4 # Tworzenie listy słowników reprezentujących organizmy
5 organizmy = [{'nazwa': nazwa, 'klasa': klasa} for nazwa, klasa in zip(nazwy, klasy)]
6
7 print(f'Lista organizmów: {organizmy}')
```

```
Lista organizmów: [{'nazwa': 'Drosophila melanogaster', 'klasa': 'Owad'}, {'nazwa': 'Homo sapiens', 'klasa': 'Ssak'}, {'nazwa': 'Escherichia coli', 'klasa': 'Bakteria'}]
```



```
1 sekwencje = ['ATCGGTAG', 'GGGCGC', 'TTATTA', 'CGATATCGAT']
2
3 # Obliczanie procentowej zawartości GC dla każdej sekwencji
4 zawartosc_GC = [(sekwencja.count('G') + sekwencja.count('C')) / len(sekwencja) * 100 for sekwencja in sekwencje]
5
6 print(f'Procentowa zawartość GC w sekwencjach: {zawartosc_GC}')
7
```

```
1  sekwencje = ['ATCGGTAG', 'GGGCGC', 'TTATTA', 'CGATATCGAT']
2
3  # Obliczanie procentowej zawartości GC dla każdej sekwencji
4  zawartosc_GC = [(sekwencja.count('G') + sekwencja.count('C')) / len(sekwencja) * 100 for sekwencja in sekwencje]
5
6  print(f'Procentowa zawartość GC w sekwencjach: {zawartosc_GC}')
7
```

Procentowa zawartość GC w sekwencjach: [50.0, 100.0, 0.0, 40.0]

1. Otwieramy linię komend (cmd, PowerShell, terminal itp.)
2. Przechodzimy do folderu z naszymi skryptami.
3. Aktywujemy środowisko wirtualne:



```
.\venv\Scripts\activate.ps1
```



```
source venv/bin/activate
```

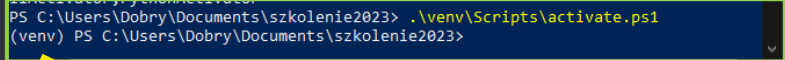
4. Instalujemy bibliotekę *matplotlib*:

```
pip install matplotlib
```

5. Przy okazji, zainstalujemy również pandas i openpyxl

```
pip install pandas
```

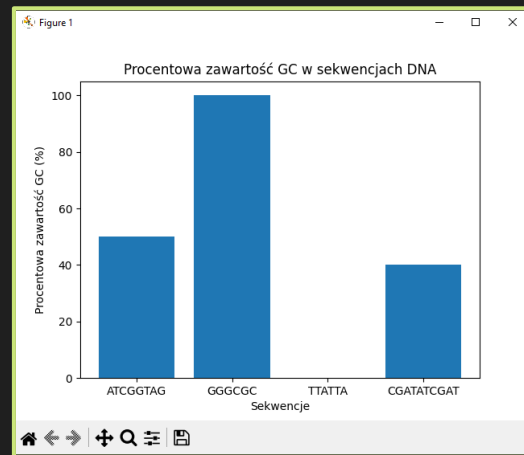
```
pip install openpyxl
```



```
PS C:\Users\Dobry\Documents\szkolenie2023> .\venv\Scripts\activate.ps1  
(venv) PS C:\Users\Dobry\Documents\szkolenie2023>
```

```
1  import matplotlib.pyplot as plt
2
3  # Dane wejściowe: sekwencje DNA
4  sekwencje = ['ATCGGTAG', 'GGGCGC', 'TTATTA', 'CGATATCGAT']
5
6  # Obliczanie procentowej zawartości GC dla każdej sekwencji
7  zawartosc_GC = [(sekwencja.count('G') + sekwencja.count('C')) / len(sekwencja) * 100 for sekwencja in sekwencje]
8
9  # Wydrukowanie wyników
10 print(f'Procentowa zawartość GC w sekwencjach: {zawartosc_GC}')
11
12 # Tworzenie wykresu słupkowego
13 plt.bar(sekwencje, zawartosc_GC)
14
15 # Dodawanie tytułu i etykiet
16 plt.title('Procentowa zawartość GC w sekwencjach DNA')
17 plt.xlabel('Sekwencje')
18 plt.ylabel('Procentowa zawartość GC (%)')
19
20 # Wyświetlenie wykresu
21 plt.show()
22
```

```
1  import matplotlib.pyplot as plt
2
3  # Dane wejściowe: sekwencje DNA
4  sekwencje = ['ATCGGTAG', 'GGGCGC', 'TTATTA', 'CGATATCGAT']
5
6  # Obliczanie procentowej zawartości GC dla każdej sekwencji
7  zawartosc_GC = [(sekwencja.count('G') + sekwencja.count('C')) / len(sekwencja) * 100 for sekwencja in sekwencje]
8
9  # Wydrukowanie wyników
10 print(f'Procentowa zawartość GC w sekwencjach: {zawartosc_GC}')
11
12 # Tworzenie wykresu słupkowego
13 plt.bar(sekwencje, zawartosc_GC)
14
15 # Dodawanie tytułu i etykiet
16 plt.title('Procentowa zawartość GC w sekwencjach DNA')
17 plt.xlabel('Sekwencje')
18 plt.ylabel('Procentowa zawartość GC (%)')
19
20 # Wyświetlenie wykresu
21 plt.show()
22
```



1. Za pomocą pandas, pobierz dane z pliku bialka_sekwencje.xlsx
2. Dla każdej rodziny sekwencji białkowych (dane syntetyczne) oblicz
 - średnią długość sekwencji
 - średnią liczbę cystein ("C") w sekwencjach
3. Dla każdej z policzonych wartości stwórz:
 - opis wyników w konsoli, z wykorzystaniem f-stringów
 - wizualizację w postaci wykresu słupkowego

```
12 data = pd.read_excel("bialka_sekwencje.xlsx")
13
14 for index, row in data.iterrows():
15     family_id = row["family"]
16     sequence = row["sequence"]
```

```
1 lista_liczb = [10, 20, 30, 40, 50]
2 suma = sum(lista_liczb)
3 liczba_liczb = len(lista_liczb)
4 print(suma / liczba_liczb)
5
```

30.0

```
1 lista_liczb1 = [10, 20, 30, 40, 50]
2 suma1 = sum(lista_liczb1)
3 liczba_liczb1 = len(lista_liczb1)
4 print(suma1 / liczba_liczb1)
5
6 lista_liczb2 = [11, 20, 33, 40, 50]
7 suma2 = sum(lista_liczb2)
8 liczba_liczb2 = len(lista_liczb2)
9 print(suma2 / liczba_liczb2)
10
11 lista_liczb3 = [12, 1110, 15, 430, 540]
12 suma3 = sum(lista_liczb3)
13 liczba_liczb3 = len(lista_liczb3)
14 print(suma3 / liczba_liczb3)
15
16 lista_liczb4 = [13, 20, 630, 88, 50]
17 suma4 = sum(lista_liczb4)
18 liczba_liczb4 = len(lista_liczb4)
19 print(suma4 / liczba_liczb4)
20
21 lista_liczb5 = [14, 50, 3111, 12, 50]
22 suma5 = sum(lista_liczb5)
23 liczba_liczb5 = len(lista_liczb5)
```

```
30.0
30.8
421.4
160.2
647.4
...
```



```
1 def srednia_liczb(lista_liczb):
2     """Funkcja obliczająca średnią arytmetyczną z listy liczb."""
3     suma = sum(lista_liczb)
4     liczba_liczb = len(lista_liczb)
5     return suma / liczba_liczb
6
7 print(srednia_liczb([10, 20, 30, 40, 50]))
8 print(srednia_liczb([11, 20, 33, 40, 50]))
9 print(srednia_liczb([12, 1110, 15, 430, 540]))
10 print(srednia_liczb([13, 20, 630, 88, 50]))
11 print(srednia_liczb([14, 50, 3111, 12, 50]))
12
```

```
30.0
30.8
421.4
160.2
647.4
...
```

```
1 def srednia_liczb_wazona(lista_liczb, wagi=None):
2     """
3     Funkcja obliczająca ważoną średnią arytmetyczną z listy liczb.
4     Jeśli wagi nie są podane, oblicza zwykłą średnią arytmetyczną.
5
6     Argumenty:
7     - lista_liczb: lista wartości liczbowych.
8     - wagi: lista wag dla każdej wartości; domyślnie None, co oznacza równą wagę dla każdej liczby.
9     """
10    if wagi is None:
11        # Jeśli wagi nie są podane, zachowuje się jak zwykła średnia
12        return sum(lista_liczb) / len(lista_liczb)
13    else:
14        # Obliczenie ważonej średniej
15        suma_wazona = sum(waga * liczba for waga, liczba in zip(wagi, lista_liczb))
16        suma_wag = sum(wagi)
17        return suma_wazona / suma_wag
18
19 # Przykłady użycia funkcji
20 wynik_zwykly = srednia_liczb_wazona([10, 20, 30])
21 wynik_wazono = srednia_liczb_wazona([10, 20, 30], wagi=[1, 2, 3])
22
23 print("Zwykła średnia:", wynik_zwykly)
24 print("Ważona średnia:", wynik_wazono)
25
```

```
1 def srednia_liczb_wazona(lista_liczb, wagi=None):
2     """
3     Funkcja obliczająca ważoną średnią arytmetyczną z listy liczb.
4     Jeśli wagi nie są podane, oblicza zwykłą średnią arytmetyczną.
5
6     Argumenty:
7     - lista_liczb: lista wartości liczbowych.
8     - wagi: lista wag dla każdej wartości; domyślnie None, co oznacza równą wagę dla każdej liczby.
9     """
10    if wagi is None:
11        # Jeśli wagi nie są podane, zachowuje się jak zwykła średnia
12        return sum(lista_liczb) / len(lista_liczb)
13    else:
14        # Obliczenie ważonej średniej
15        suma_wazona = sum(waga * liczba for waga, liczba in zip(wagi, lista_liczb))
16        suma_wag = sum(wagi)
17        return suma_wazona / suma_wag
18
19 # Przykłady użycia funkcji
20 wynik_zwykly = srednia_liczb_wazona([10, 20, 30])
21 wynik_wazono = srednia_liczb_wazona([10, 20, 30], wagi=[1, 2, 3])
22
23 print("Zwykła średnia:", wynik_zwykly)
24 print("Ważona średnia:", wynik_wazono)
25
```

Zwykła średnia: 20.0

Ważona średnia: 23.333333333333332

```
1 def srednia_liczb_wazona(*lista_liczb, wagi=None):
2     """
3     Funkcja obliczająca ważoną średnią arytmetyczną z listy liczb.
4     Jeśli wagi nie są podane, oblicza zwykłą średnią arytmetyczną.
5
6     Argumenty:
7     - lista_liczb: lista wartości liczbowych.
8     - wagi: lista wag dla każdej wartości; domyślnie None, co oznacza równą wagę dla każdej liczby.
9     """
10    if wagi is None:
11        # Jeśli wagi nie są podane, zachowuje się jak zwykła średnia
12        return sum(lista_liczb) / len(lista_liczb)
13    else:
14        # Obliczenie ważonej średniej
15        suma_wazona = sum(waga * liczba for waga, liczba in zip(wagi, lista_liczb))
16        suma_wag = sum(wagi)
17        return suma_wazona / suma_wag
18
19 # Przykłady użycia funkcji
20 wynik_zwykly = srednia_liczb_wazona(10, 20, 30)
21 wynik_wazono = srednia_liczb_wazona(10, 20, 30, 40, wagi=[1, 2, 3, 4])
22
23 print("Zwykła średnia:", wynik_zwykly)
24 print("Ważona średnia:", wynik_wazono)
25
```

```
1 def srednia_liczb_wazona(*lista_liczb, wagi=None):
2     """
3     Funkcja obliczająca ważoną średnią arytmetyczną z listy liczb.
4     Jeśli wagi nie są podane, oblicza zwykłą średnią arytmetyczną.
5
6     Argumenty:
7     - lista_liczb: lista wartości liczbowych.
8     - wagi: lista wag dla każdej wartości; domyślnie None, co oznacza równą wagę dla każdej liczby.
9     """
10    if wagi is None:
11        # Jeśli wagi nie są podane, zachowuje się jak zwykła średnia
12        return sum(lista_liczb) / len(lista_liczb)
13    else:
14        # Obliczenie ważonej średniej
15        suma_wazona = sum(waga * liczba for waga, liczba in zip(wagi, lista_liczb))
16        suma_wag = sum(wagi)
17        return suma_wazona / suma_wag
18
19 # Przykłady użycia funkcji
20 wynik_zwykly = srednia_liczb_wazona(10, 20, 30)
21 wynik_wazony = srednia_liczb_wazona(10, 20, 30, 40, wagi=[1, 2, 3, 4])
22
23 print("Zwykła średnia:", wynik_zwykly)
24 print("Ważona średnia:", wynik_wazony)
25
```

Zwykła średnia: 20.0

Ważona średnia: 30.0

1. Zmodyfikuj funkcję `srednia_liczb_wazona()` tak, żeby liczba podanych w drugim argumencie wag, musiała być równa liczbie podanych w pierwszym argumencie liczb (na przykład: wyświetl komunikat w przypadku niezgodności).

1. Zdefiniuj taką funkcję, lub kilka funkcji, które umożliwią wykonywanie operacji z ćwiczenia_01 w oparciu o następujące argumenty:

- dowolną ścieżkę do arkusza kalkulacyjnego
- dowolny symbol reszty aminokwasowej (do obliczenia średniej zawartości)
- (na podstawie materiałów bonusowych) dowolny kolor wykresu słupkowego

2. Napisz funkcję, która przyjmować będzie plik tekstowy zawierający sekwencję DNA w formacie FASTA, a następnie stworzy wykres słupkowy, w którym każdy słupek będzie innego koloru i będzie przedstawiał liczbę wystąpień konkretnej zasady azotowej.

Format FASTA - przykład:

```
> seq_01
ACTGACCCCATAGACAGATTACA
```

Kilka kolorów na jednym wykresie:

```
plt.bar(nazwy, wartosci, color=['blue', 'green', 'red', 'yellow'])
```

```
1 harmonogram_listopad = {
```

```
2  
3     M    T    W    T    F    S    S  
4
```

```
5         01    02    03    04    05  
6
```

```
7     06    07    08    09    10    11    12  
8
```

```
9     13    14    15    16    17    18    19    Do 18 listopada  
10
```

```
11    20    21    22    23    24    25    26    Zadania domowe  
12
```

```
13    27    28    29    30    mateusz.dobrychlop@gmail.com  
14
```

```
}
```


1

2

3

4

5

6

7

8

9

10

11

12

13

14

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**