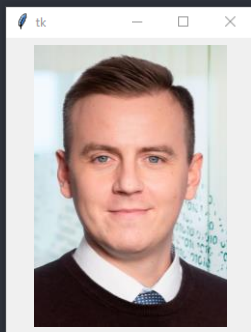


```
1
2
3  Praktyczne_zastosowanie_Pythona = {
4
5      w_naukach : ['biologicznych', 'i medycznych'],
6      dla :      ['początkujących'],
7      część :    ['pierwsza'],
8
9          # Mateusz Dobrychłop, 16 maja 2023
10
11
12  }
```

```
1  
2  
3 Prowadzący = {  
4
```



```
12  
13  
14
```

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

```
1 harmonogram_maj = {
```

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

```
3     01   02   03   04   05   06   07
```

```
4     08   09   10   11   12   13   14
```

```
5     15   16   17   18   19   20   21     Spotkanie 1 (17:00 - 19:30)
```

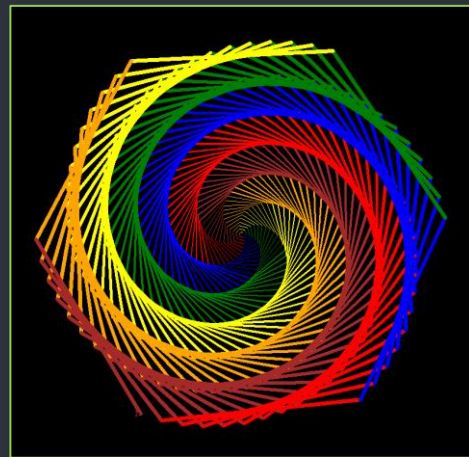
```
6     22   23   24   25   26   27   28     Spotkanie 2 (17:00 - 19:30)
```

```
7     29   30   31     Spotkanie 3 (17:00 - 19:30)
```

```
8 }  
9  
10  
11  
12  
13  
14
```

```
1 Plan_szkolenia = {
2
3     01 Absolutne podstawy
4       [ uruchamianie kodu, podstawowe
5         typy danych, warunki ]
6
7     02 Praca z sekwencjami
8       [ listy, pętle, pliki
9         tekstowe ]
10
11     03 Arkusze danych
12       [ pandas, arkusze,
13         dokumentacja ]
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\\_pocztakujacy\\_lvl1\\_2023/](https://github.com/mdobrychlop/python_pocztakujacy_lvl1_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 zostanie
7 # przekonwertowany na łatwy string.
8
9 print("hello, world!")
10
11 # string_02.py
12 #
13 # Ten razem deklarujemy zmienną test, do której
14 # przekazujemy łatwy obiekt. Następnie,
15 # tą zmienną przekazujemy metodzie print().
16
17 test = "hello, world!"
18 print(test)
19
20 # string_03.py
21 #
22 # Na zmiennych możemy wykonać działania.
23 # Np. możemy połączyć dwa krótkie teksty
24 # w jeden długi.
25
26 test1 = "hello, "
27 test2 = "world!"
28 full_text = test1 + test2
29 print(full_text)
30
31 # string_04.py
32 #
33 # Na zmiennych możemy wykonać działania.
34 # Np. możemy połączyć dwa krótkie teksty
35 # w jeden długi.
36
37 test1 = "hello, "
38 test2 = "world!"
39 print(test1 + test2)
40
41 # int_01.py
42 # Długość liczbowa (takie jak integer, czyli
43 # liczby całkowite) może występować w
44 # działaniach matematycznych
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```


**\_01 = {**

**'Absolutne podstawy' :**

**[** 'pliki \*.py',  
'tryb interaktywny',  
'deklarowanie zmiennych',  
'liczby całkowite',  
'łańcuchy znaków',  
'instrukcje warunkowe'  
**]**


**}**

```
1 Kod_w_pliku = {
```

```
2  
3 | [] : ['plik tekstowy z rozszerzeniem .py',  
4 |   'uruchamianie w konsoli lub w edytorze',  
5 |   'złożone skrypty']
```

```
6 }  
7
```

```
8 Tryb_interaktywny = {  
9
```

```
10 | [] : ['bez konieczności zapisywania pliku',  
11 |   'uruchamianie np. w konsoli lub w IDLE',  
12 |   'szybki efekt, proste instrukcje']
```

```
13 }  
14
```



```
1 Kod_w_pliku = {
```

```
2  
3 | [📄] : ['plik tekstowy z rozszerzeniem .py',  
4 |      'uruchamianie w konsoli lub w edytorze',  
5 |      'złożone skrypty']
```

```
6 }  
7
```

```
8 Tryb_interaktywny = {  
9
```

```
10 | [🖥️] : ['bez konieczności zapisywania pliku',  
11 |       'uruchamianie np. w konsoli lub w IDLE',  
12 |       'szybki efekt, proste instrukcje']
```

```
13 }  
14
```



[www.online-python.com](http://www.online-python.com)

```
1  Uruchamiamy_plik_py = {
2
3      'Krok_1': 'Tworzymy nowy plik w Sublime Text,
4                zapisujemy go z rozszerzeniem .py'
5
6      'Krok_2': 'Wklejamy do pliku kod z następnego
7                slajdu'
8
9      'Krok_3': 'Zapisujemy plik'
10
11      'Krok_4': 'Wybieramy Tools / Build lub
12                naciskamy Ctrl+B'
13  }
14
```

```
1  -*- coding: utf-8 -*-
2  #
3  # Metoda print() pozwala na wyświetlenie tekstu w
4  # konsoli. Jako argument przyjmuje dowolny obiekt.
5  # Przed wyświetleniem obiekt zostaje
6  # przekonwertowany na łańcuch znaków.
7
8  print('hello, world!')
9
10
11
12
13
14
```

```
1  -*- coding: utf-8 -*-
2  #
3  # Metoda print() pozwala na wyświetlenie tekstu w
4  # konsoli. Jako argument przyjmuje dowolny obiekt.
5  # Przed wyświetleniem obiekt zostaje
6  # przekonwertowany na łańcuch znaków.
7
8  print('hello, world!')
9
10
11
12
13
14
```

hello, world!

```
1  # Tym razem deklarujemy zmienną text, do której
2  # przekazujemy łańcuch znaków. Następnie,
3  # tę zmienną przekazujemy metodzie print().
4
5  text = 'hello, world!'
6  print(text)
7
8
9
10
11
12
13
14
```

```
1  # Tym razem deklarujemy zmienną text, do której
2  # przekazujemy łańcuch znaków. Następnie,
3  # tę zmienną przekazujemy metodzie print().
4
5  text = 'hello, world!'
6  print(text)
7
8
9
10
11
12
13
14
```

hello, world!

```
1 Przykładowe_typy_danych = {
```

```
2  
3  
4     String
```

```
5     'lancuch znakow'  
6     'po prostu tekst'  
7     'jeden, dwa 1 2'
```

```
     Integer
```

```
1     1  
2     42  
3     999999999
```

```
8  
9     Float
```

```
10    3.14  
11    120.5  
12    5.33333
```

```
     List
```

```
1     ['jajka', 'maslo', 'ryz']  
2     [1, 10, 12, 3, 5, 7, 7, 4, 10]  
3     ['kot', 25, 'pies', 1.5]
```

```
13  
14 }
```

```
1  # Na zmiennych możemy wykonywać działania.
2  # Spróbujemy połączyć dwa krótsze łańcuchy
3  # w jeden dłuższy.
4
5  text1 = 'hello, '
6  text2 = 'world!'
7  full_text = text1 + text2
8  print(full_text)
9
10
11
12
13
14
```



```
1  # Na zmiennych możemy wykonywać działania.
2  # Spróbujmy połączyć dwa krótsze łańcuchy
3  # w jeden dłuższy.
4
5  text1 = 'hello, '
6  text2 = 'world!'
7  full_text = text1 + text2
8  print(full_text)
9
10
11
12
13
14
```

hello, world!

```
1  # Na zmiennych możemy wykonywać działania.
2  # Spróbujmy połączyć dwa krótsze łańcuchy
3  # w jeden dłuższy.
4
5  text1 = 'hello, '
6  text2 = 'world!'
7  print(text1 + text2)
8
9
10
11
12
13
14
```

```
1  # Na zmiennych możemy wykonywać działania.
2  # Spróbujmy połączyć dwa krótsze łańcuchy
3  # w jeden dłuższy.
4
5  text1 = 'hello, '
6  text2 = 'world!'
7  print(text1 + text2)
8
9
10
11
12
13
14
```

hello, world!

```
1  # Zmienne liczbowe (takie jak integery, czyli
2  # liczby całkowite) można wykorzystywać w
3  # działaniach matematycznych
4
5  text1 = 'hello, '
6  text2 = 'world!'
7  num1 = 13
8  num2 = 7
9  print(num1 + num2)
10
11
12
13
14
```

```
1  # Zmienne liczbowe (takie jak integer, czyli
2  # liczby całkowite) można wykorzystywać w
3  # działaniach matematycznych
4
5  text1 = 'hello, '
6  text2 = 'world!'
7  num1 = 13
8  num2 = 7
9  print(num1 + num2)
10
11
12
13
14
```

20

```
1  # Spróbujemy dodać łańcuch znaków do
2  # liczby całkowitej.
3
4  text1 = 'hello, '
5  text2 = 'world!'
6  num1 = 13
7  num2 = 7
8  print(text1 + num2)
9
10
11
12
13
14
```

```
1  # Spróbujemy dodać łańcuch znaków do
2  # liczby całkowitej.
3
4  text1 = 'hello, '
5  text2 = 'world!'
6  num1 = 13
7  num2 = 7
8  print(text1 + num2)
9
10
11
12
13
14
```

```
Traceback (most recent call last):
  File "C:\Users\Dobry\Documents\szkolenie_python_2022\kod.py",
line 8, in <module>
    print(text1 + num2)
TypeError: can only concatenate str (not "int") to str
```

```
1  # Metoda str() konwertuje dane różnego typu na
2  # lancach znaków, a metoda int() na liczbę
3  # całkowitą. Metoda float() konwertuje liczbę
4  # całkowitą na zmiennoprzecinkową.
5
6  text1 = '3'
7  num1 = 13
8  num_to_text = str(num1)
9  text_to_num = int(text1)
10 print(text1 + num_to_text)
11 print(num1 + text_to_num)
12 print(float(num1))
13 print(text1, num1, num_to_text, text_to_num)
14
```



```
1  # Metoda str() konwertuje dane różnego typu na
2  # lancach znaków, a metoda int() na liczbę
3  # całkowitą. Metoda float() konwertuje liczbę
4  # całkowitą na zmiennoprzecinkową.
5
6  text1 = '3'
7  num1 = 13
8  num_to_text = str(num1)
9  text_to_num = int(text1)
10 print(text1 + num_to_text)
11 print(num1 + text_to_num)
12 print(float(num1))
13 print(text1, num1, num_to_text, text_to_num)
14
```

```
313
16
13.0
3 13 13 3
```

```
1  # Instrukcja warunkowa if (jeżeli warunek jest
2  # spełniony, to...)
3
4  num1 = 15
5
6  if num1 > 10:
7      print('true!')
8
9  print('done.')
10
11
12
13
14
```

```
1  # Instrukcja warunkowa if (jeżeli warunek jest
2  # spełniony, to...)
3
4  num1 = 15
5
6  if num1 > 10:
7      print('true!')
8
9  print('done.')
10
11
12
13
14
```

```
true!
done.
```

```
1  # Instrukcja warunkowa if (jeżeli warunek jest
2  # spełniony, to...) / else (w innym wypadku...)
3
4  num1 = 15
5
6  if num1 == 10:
7      print('true!')
8  else:
9      print('not true!')
10
11 print('done.')
12
13
14
```

```
1  # Instrukcja warunkowa if (jeżeli warunek jest
2  # spełniony, to...) / else (w innym wypadku...)
3
4  num1 = 15
5
6  if num1 == 10:
7      print('true!')
8  else:
9      print('not true!')
10
11 print('done.')
12
13
14
```

not true!  
done.

```
1  # Instrukcja elif (sprawdź kolejny warunek,  
2  # jeśli ten powyżej nie jest spełniony)  
3  
4  num1 = 15  
5  num2 = 20  
6  
7  if num1 > num2:  
8      print('pierwsza liczba jest wieksza')  
9  elif num1 == num2:  
10     print('liczby sa rowne')  
11 else:  
12     print('druga liczba jest wieksza')  
13  
14 print('done.')
```

```
1  # Instrukcja elif (sprawdź kolejny warunek,  
2  # jeśli ten powyżej nie jest spełniony)  
3  
4  num1 = 15  
5  num2 = 20  
6  
7  if num1 > num2:  
8      print('pierwsza liczba jest wieksza')  
9  elif num1 == num2:  
10     print('liczby sa rowne')  
11 else:  
12     print('druga liczba jest wieksza')  
13  
14 print('done.')
```

druga liczba jest wieksza  
done.

```
1  # Metoda input() pozwala na wprowadzenie tekstu
2  # po uruchomieniu programu. Jako argument
3  # przyjmuje komunikat dla użytkownika.
4  num1 = int(input('Podaj pierwsza liczbe:'))
5  num2 = int(input('Podaj druga liczbe:'))
6
7  if num1 > num2:
8      print('pierwsza liczba jest wieksza')
9  elif num1 == num2:
10     print('liczby sa rowne')
11 else:
12     print('druga liczba jest wieksza')
13
14 print('done.')
```



```
1  # Metoda input() pozwala na wprowadzenie tekstu
2  # po uruchomieniu programu. Jako argument
3  # przyjmuje komunikat dla użytkownika.
4  num1 = int(input('Podaj pierwsza liczbe:'))
5  num2 = int(input('Podaj druga liczbe:'))
6
7  if num1 > num2:
8      print('pierwsza liczba jest wieksza')
9  elif num1 == num2:
10     print('liczby sa rowne')
11 else:
12     print('druga liczba jest wieksza')
13
14 print('done.')
```

Podaj pierwsza liczbe:35  
Podaj druga liczbe:12  
pierwsza liczba jest  
wieksza  
done.

```
1  # Zadanie 1: LICZBY PARZYSTE  
2  # - Użytkownik wprowadza liczbę  
3  # - W konsoli wyświetlony zostaje komunikat, czy liczba jest  
4  #   parzysta  
5  # Zadanie 2: SUMA LICZB n-1 I n+1  
6  # - Użytkownik wprowadza liczbę  
7  # - W konsoli wyświetlony zostaje wynik dodawania liczby o 1  
8  #   mniejszej i o 1 większej niż wprowadzona (np. dla liczby 5,  
9  #   wyświetlony zostaje wynik działania 4+6)  
10 # Zadanie 3: KALKULATOR  
11 # - Użytkownik wprowadza dwie liczby oraz symbol działania  
12 #   (funkcja input())  
13 # - W konsoli wyświetlony zostaje wynik działania  
14 # Zadanie 4: SORTOWANIE  
15 # - Użytkownik wprowadza po kolei trzy liczby  
16 # - W konsoli wyświetlone zostają te liczby, posortowane  
17 #   od najmniejszej do największej  
18  
19 # mateusz.dobrychlop@gmail.com
```

```
1 harmonogram_maj = {
```

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

```
3     01   02   03   04   05   06   07
```

```
4     08   09   10   11   12   13   14
```

```
5     15   16   17   18   19   20   21     Do 20 maja
```

```
6     22   23   24   25   26   27   28     Zadania domowe
```

```
7     29   30   31     mateusz.dobrychlop@gmail.com
```

```
8 }  
9  
10  
11  
12  
13  
14
```

## list\_01.py

```
1  # Lista to typ danych, który pozwala na
2  # przechowywanie wielu elementów różnego typu.
3  # Odwołujemy się do nich za pomocą indeksów.
4  #
5  # indeksy:   0       1       2       3       4
6  animals = ['dog', 'cat', 'bird', 'hamster', 'bat']
7
8  print(animals[0])
9  print(animals[2])
10
11 animals[3] = 'mouse'
12 print(animals)
13
14
15
16
17
18
19
20
21
22
23
24
25
```

## listy\_i\_indeksy.py

## list\_02.py

```
1  # Lista to typ danych, który pozwala na
2  # przechowywanie wielu elementów różnego typu.
3  # Odwołujemy się do nich za pomocą indeksów.
4  #
5  # indeksy:   0       1       2       3       4
6  animals = ['dog', 'cat', 'bird', 'hamster', 'bat']
7
8  print(animals[0])
9  print(animals[2])
10
11 animals[3] = 'mouse'
12 print(animals)
13
14 print('[-1]', animals[-1])
15 print('[2:5]', animals[2:5])
16
17
18
19
20
21
22
23
24
25
```

## listy\_i\_indeksy.py

```
1  # Zasady wykrawania list:
2  #
3  # lista[pierwszy_element:ostatni_element+1]
4  #
5  # przykład: lista[2:5]:
6  # - element 2 będzie uwzględniony
7  # - element 5 nie będzie uwzględniony
8
9
10 # indeks:      0      1      2      3      4      5
11 animals = ['dog', 'cat', 'bird', 'hamster', 'bat', 'cow']
12 # od końca: -6     -5     -4     -3     -2     -1
13
14
15 animals[2:5] # -> 2, 3, 4      / bird, hamster, bat
16 animals[2:]  # -> 2, 3, 4, 5   / bird, hamster, bat, cow
17 animals[1:4] # -> 1, 2, 3     / cat, bird, hamster
18 animals[:4]  # -> 0, 1, 2, 3   / dog, cat, bird, hamster
19 animals[-3:] # -> 3, 4, 5     / hamster, bat, cow
20
21
22
23
24
25
```

```
1  # Za pomocą indeksów można również odwoływać się
2  # do kolejnych znaków w łańcuchu znaków.
3
4
5  #      012345
6  name = 'Janusz'
7  print(name[2])
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
```

```
1  # Za pomocą indeksów można również odwoływać się
2  # do kolejnych znaków w łańcuchu znaków.
3
4
5  #      012345
6  name = 'Janusz'
7  print(name[2])
8
9  animals = ['dog', 'cat', 'bird', 'hamster', 'bat']
10 print(animals[2][-1])
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
```



```
1  # Metoda append() pozwala na dodawanie elementów do listy.
2  # Metoda remove() pozwala na usuwanie elementów z listy.
3  # Metoda len() pozwala na sprawdzenie długości listy.
4
5
6  # indeksy:  0      1      2      3      4
7  animals = ['dog', 'cat', 'bird', 'hamster', 'bat']
8
9  animals.append('cat')
10 print(animals)
11
12 animals.remove('cat') # usuwa pierwszy znaleziony przykład
13 print(animals)
14
15 print(len(animals))
16
17
18
19
20
21
22
23
24
25
```

```
1  # Pętla to niezwykle istotny aspekt programowania.
2  # Pętla while pozwala powtarzać pewną instrukcję, dopóki
3  # zdefiniowany przez nas warunek jest spełniony.
4
5
6  # indeksy:  0      1      2      3      4
7  animals = ['dog', 'cat', 'bird', 'hamster', 'bat']
8
9  counter = 0
10 list_length = len(animals)
11
12 while counter < list_length:
13     print(animals[counter])
14     counter += 1  # to znaczy to samo co: counter = counter + 1
15
16
17
18
19
20
21
22
23
24
25
```

```
1  # Pętla to niezwykle istotny aspekt programowania.
2  # Pętla while pozwala powtarzać pewną instrukcję, dopóki
3  # zdefiniowany przez nas warunek jest spełniony.
4
5
6  # indeksy:  0      1      2      3      4
7  animals = ['dog', 'cat', 'bird', 'hamster', 'bat']
8
9  counter = 0
10 list_length = len(animals)
11
12 while counter < list_length:
13     print(animals[counter])
14     # counter += 1 # to znaczy to samo co: counter = counter + 1
15
16
17
18
19
20
21
22
23
24
25
```

```
1  # Pętla for pozwala na iterowanie po elementach sekwencji
2  # - na przykład listy lub łańcucha znaków.
3
4
5  # indeksy:   0       1       2       3       4
6  animals = ['dog', 'cat', 'bird', 'hamster', 'bat']
7
8  counter = 0
9  list_length = len(animals)
10
11 for a in animals:
12     print(a)
13     # counter += 1 # to znaczy to samo co: counter = counter + 1
14
15
16
17
18
19
20
21
22
23
24
25
```

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**