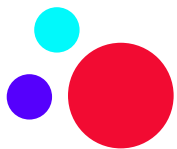


# Python Typy Danych

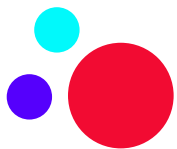


# Python Typy Danych

## Agenda

info **Share**  
ACADEMY

- 1 **Wiersz polecenia**
- 2 Python – czym jest, jego historia i zastosowanie
- 3 Python – typy danych
- 4 Python – operatory arytmetyczne

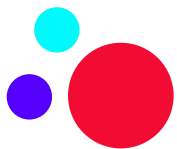


# Python Typy Danych

## Agenda

info **Share**  
ACADEMY

- 1 Wiersz polecenia
- 2 **Python – czym jest, jego historia i zastosowanie**
- 3 Python – typy danych
- 4 Python – operatory arytmetyczne



# Python Typy Danych

## Agenda

1

Wiersz polecenia

2

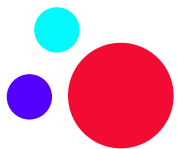
Python – czym jest, jego historia i zastosowanie

3

**Python – typy danych**

4

Python – operatory arytmetyczne

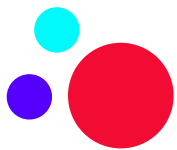


# Python Typy Danych

## Agenda

info **Share**  
ACADEMY

- 1 Wiersz polecenia
- 2 Python – czym jest, jego historia i zastosowanie
- 3 Python – typy danych
- 4 **Python – operatory arytmetyczne**

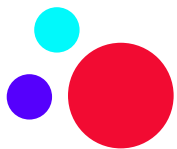


# Python Typy Danych

## Wprowadzenie do wiersza polecenia

```
Wiersz polecenia
Microsoft Windows [Version 10.0.18362.1256]
(c) 2019 Microsoft Corporation. Wszelkie prawa zastrzeżone.
C:\Users\Lenovo>
```

```
mac -- zsh -- 126x31
Last login: Tue Oct 10 19:27:10 on ttys000
mac@iMac ~ % open $1P01A
mac@iMac ~ %
[0]tworzone: 11.10.2020 o 06:01:22
Last login: Wed Oct 11 08:01:00 on console
mac@iMac ~ %
[0]tworzone: 22.07.2020 o 20:04:07
Last login: Mon Feb 12 18:04:24 on console
mac@iMac ~ %
```



# Python Typy Danych

## Uruchomienie interfejsu wiersza polecenia

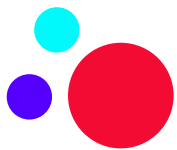
### Windows:

- Przejdź do menu Start lub ekranu i wpisz "Command Prompt", lub „cmd” w polu wyszukiwania.
- Idź do Start menu → Windows System → Command Prompt.
- Idź do Start menu → All Programs → Accessories → Command Prompt.
- Przytrzymaj specjalny klawisz Windows na klawiaturze i naciśnij klawisz "X". Wybierz "Command Prompt" (wiersz poleceń) z wyskakującego menu.
- Przytrzymaj klawisz Windows i naciśnij klawisz "R", aby uzyskać okno "Run"('Uruchom'). Wpisz "cmd" w polu i kliknij przycisk OK.

### macOS:

- Przejdź do Applications → Utilities → Terminal.






# Python Typy Danych

## Bieżący katalog

Bieżący katalog, czyli „gdzie się znajdujemy w ścieżce”

### Windows:

cd – change directory

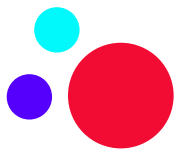
 Wiersz polecenia

```
Microsoft Windows [Version 10.0.18362.1256]  
(c) 2019 Microsoft Corporation. Wszelkie prawa zastrzeżone.  
  
C:\Users\Lenovo>cd  
C:\Users\Lenovo  
  
C:\Users\Lenovo>
```

### macOs:

pwd – print working directory





# Python Typy Danych

## Lista plików i folderów

### Windows:

- Dir – directory

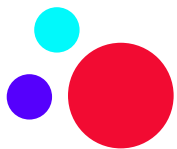
### macOS lub PowerShell:

- ls

```
C:\> Wiersz polecenia
C:\Users\Lenovo> dir
Volume in drive C has no label.
Volume Serial Number is F0B1-835C

Directory of C:\Users\Lenovo

26.10.2023  11:07    <DIR>          .
26.10.2023  11:07    <DIR>          ..
21.10.2023  18:08    <DIR>          .anaconda
02.04.2022  22:24    <DIR>          .aws
```



# Python Typy Danych

Zmiana katalogu bieżącego

## Windows i macOS:

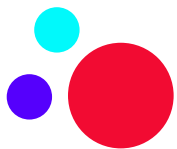
- cd Documents

```
C:\Users\Lenovo>cd Documents
```

```
C:\Users\Lenovo\Documents>cd
```

```
C:\Users\Lenovo\Documents
```

info **Share**  
ACADEMY



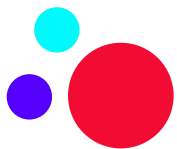
# Python Typy Danych

## Tworzenie folderu

### Windows i macOS:

- mkdir nazwa\_folderu

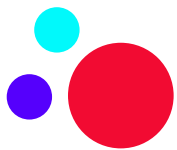
```
C:\Users\Lenovo\Documents>mkdir nowy_folder
```



# Python Typy Danych

## Wiersz poleceń – podsumowanie

Polecenie (Windows)	Polecenie (Mac OS / Linux)	Opis	Przykład
wyjście	wyjście	zamknięcie okna	<b>wyjście</b>
cd	cd	zmiana katalogu	<b>cd test</b>
cd	pwd	pokazywane bieżącego katalogu	<b>cd</b> (Windows) lub <b>pwd</b> (Mac OS / Linux)
dir	ls	lista katalogów/plików	<b>dir</b>
copy	cp	kopiowanie pliku	<b>copy c:\test\test.txt</b> <b>c:\windows\test.txt</b>
move	mv	przenoszenie pliku	<b>move c:\test\test.txt</b> <b>c:\windows\test.txt</b>
mkdir	mkdir	tworzenie nowego katalogu	<b>mkdir testdirectory</b>
rmdir (lub del)	rm	usuwanie pliku	<b>del c:\test\test.txt</b>
rmdir /S	rm -r	usuwanie katalogu	<b>rm -r testdirectory</b>
[CMD] /?	man [CMD]	uzyskiwanie pomocy na temat komendy	<b>cd /?</b> (Windows) lub <b>man cd</b> (Mac OS / Linux)



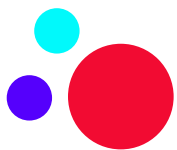
# Python Typy Danych

## Czym jest Python?

Python jest zorientowanym obiektowo językiem skryptowym o typowaniu dynamicznym.

Istnieje wiele implementacji języka Python:

- CPython – oryginalna standardowa implementacja.
- Jython / JPython – alternatywna implementacja języka celowana w integrację z językiem Java.
- IronPython – implementacja w .NET.
- ...



# Python Typy Danych

Historia

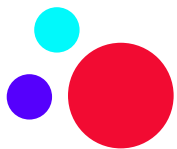
python

1997-2006

python™

2006-PRESENT

info **Share**  
ACADEMY

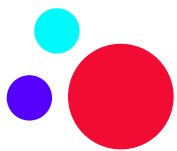


# Python Typy Danych

## Zastosowanie

1. Web Development.
2. Data Science i Analiza Danych.
3. Automatyzacja i skrypty systemowe.
4. Rozwijanie aplikacji desktopowych.
5. Game Development.
6. Inżynieria oprogramowania.



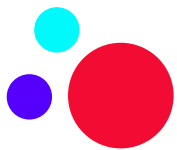


# Python Typy Danych

## Python i Data Science

Manipulacja danymi:





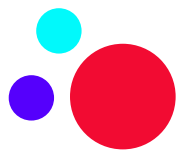
# Python Typy Danych

Python i Data Science

Wizualizacje:

matplotlib

seaborn



# Python Typy Danych

Python i Data Science

Analiza statystyczna:



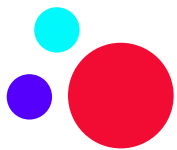
**SciPy**



statsmodels

[infoShareAcademy.com](https://infoShareAcademy.com)

info **Share**  
ACADEMY



# Python Typy Danych

Python i Data Science

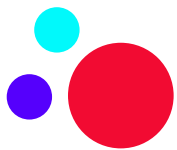
Machine Learning:



TensorFlow

[infoShareAcademy.com](https://infoShareAcademy.com)

info **Share**  
ACADEMY



# Python Typy Danych

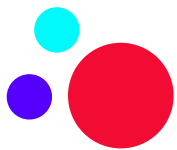
## Python i Data Science

Eksploracyjna analiza danych i przetwarzanie języka naturalnego:



[infoShareAcademy.com](https://infoShareAcademy.com)

**info** Share  
ACADEMY

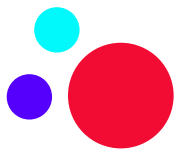


**Python Typy Danych**  
Edytory









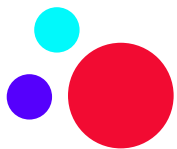
# Python Typy Danych

Organizacja kodu według PEP8

1

Wcięcia:

```
def pole_kwadratu(a, b):  
    → return a * b  
  
print(pole_kwadratu(5, 4))
```



# Python Typy Danych

Organizacja kodu według PEP8

2

Systemy notacji:

A. lower\_case\_with\_underscores

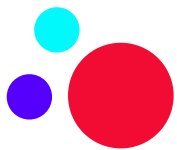
```
def pole_kwadratu(a, b):  
    return a * b
```

B. CAPS\_WITH\_UNDER

```
>>> PI = 3.14
```

C. PascalCase

```
class SportsCar:  
    def __init__(self, name):  
        self.name = name
```

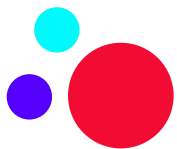


# Python Typy Danych

## Język skryptowy

W językach skryptowych instrukcje są wykonywane jedna po drugiej.

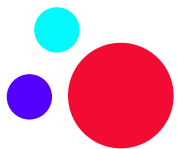
Inaczej niż w przypadku języków kompilowanych (np. C/C++, Fortran, itp.), gdzie cały kod jest tłumaczony na kod maszynowy w procesie kompilacji.



# Python Typy Danych

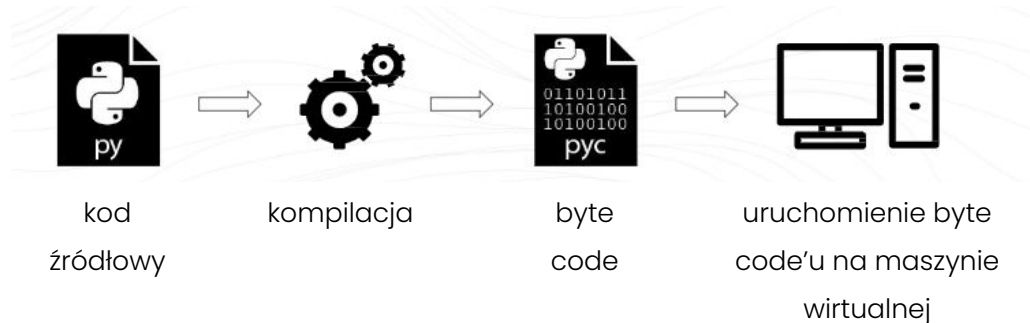
Język skryptowy

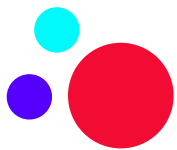
1. Interpretacja kodu źródłowego.
2. Brak kompilacji wstępnej.
3. Dynamiczne typowanie.
4. Wysoki poziom abstrakcji.
5. Szybki cykl rozwoju.



# Python Typy Danych

Jak Python wykonuje kod?



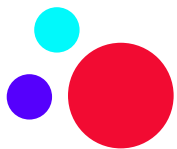


# Python Typy Danych

## Program w języku Python

Program to kod (zbiór instrukcji), zwykle zapisany w pliku tekstowym.

```
lib > script.py > ...  
1  def hello_world():  
2      print("Hello World")  
3  
4  hello_world()
```



# Python Typy Danych

## Interpreter i wykonanie kodu

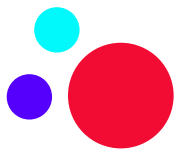
Interpreter to program, który wykonuje inne programy na podstawie instrukcji (kodu).

Interpreter odczytuje kod (np. napisany w języku Python) i wykonuje go na danym urządzeniu.

Odczytanie kodu należy rozumieć jako jego interpretację (identyfikację poszczególnych instrukcji) i kompilację (zamianę na format zrozumiały dla maszyny wirtualnej).

Wykonanie kodu oznacza uruchomienie maszyny wirtualnej i wykonanie byte code'u.

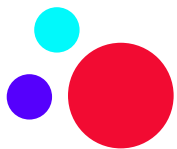




# Python Typy Danych

Maszyna wirtualna Pythona

Maszyna wirtualna Pythona (Python Virtual Machine – PVM)  
to część oprogramowania, która przechodzi kolejno przez  
instrukcje zawarte w byte code i je wykonuje.

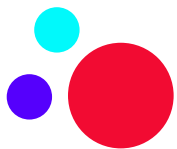


# Python Typy Danych

## Instalacja pakietów

PIP – instalator paczek / modułów / bibliotek (packages) języka Python. Jest to domyślne narzędzie instalowania modułów dla języka Python.

W przypadku używania środowiska Anaconda zwykle nie ma potrzeby używania instalatora PIP.



# Zadanie 5.1

## Instalacja pakietów (instrukcja)

1. W terminalu sprawdź wersję PIP komendą: `pip --version`
2. Zainstaluj bibliotekę NumPy za pomocą komendy: `pip install numpy`
3. Stwórz plik `requirements.txt`, a w nim zapisz pakiety o wersjach:

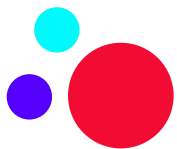
```
pandas==1.3.3
```

```
matplotlib==3.4.3
```

```
seaborn==0.11.2
```

A następnie dokonaj instalacji zawartości pliku za pomocą komendy:

```
pip install -r requirements.txt
```



# Python Typy Danych

conda

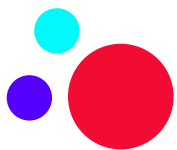
System zarządzania środowiskami wirtualnymi i instalacją pakietów dla wielu języków programowania (w tym języka Python).

Przykładowe komendy:

```
conda create --name nazwa_środowiska
```

```
conda activate nazwa_środowiska
```

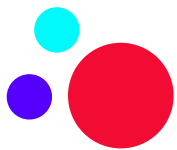
```
conda install numpy
```



# Python Typy Danych

wheel\*

Zaawansowany i niedoceniany system instalacji pakietów  
języka Python.

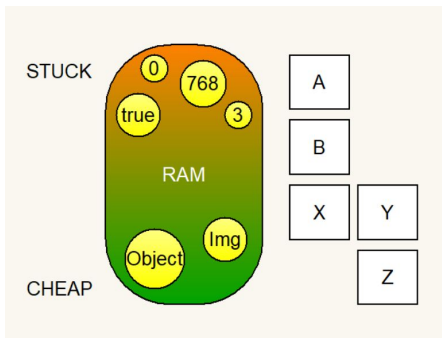


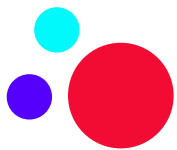
# Python Typy Danych

## Czym jest zmienna?

Zmienna – konstrukcja programistyczna posiadająca trzy atrybuty:

- nazwę symboliczną
- miejsce przechowywania
- wartość

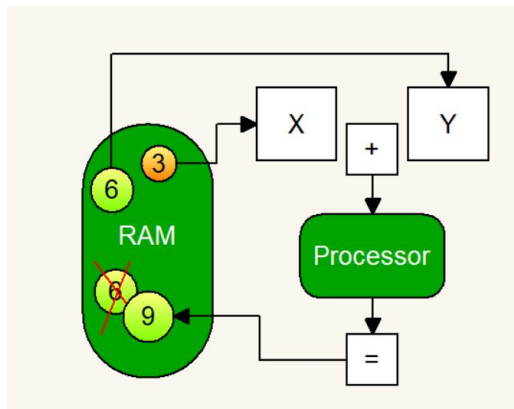




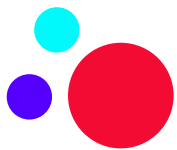
# Python Typy Danych

## Operacje na zmiennych

Wykonywanie operacji na zmiennych oznacza kopiowanie ich wartości z pamięci RAM do procesora, przeprowadzenie żądanej operacji i ostatecznie zapisanie nowej wartości dla tej zmiennej w pamięci RAM.







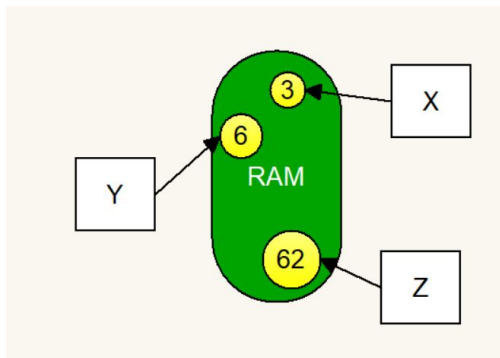
# Python Typy Danych

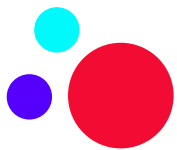
## Zmienne w języku Python

W języku Python zmienne odnoszą się w danym momencie do określonego obiektu. Przypisanie wartości zmiennej oznacza przypisanie nazwy do obiektu.

Poniższa instrukcja powoduje przypisanie nazwy 'x' do obiektu 3, czyli utworzenie referencji (odnośnika, wskaźnika) do obiektu 3.

```
>>> x = 3
```





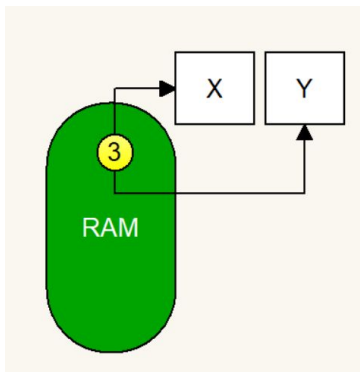
# Python Typy Danych

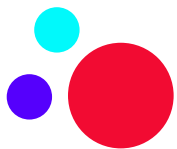
Zarządzanie pamięcią w Pythonie: referencje

W języku Python zmienne odnoszące się do tych samych obiektów (wartości) są odniesieniami (referencjami) do tego samego miejsca w pamięci.

```
>>> x = 3
```

```
>>> y = 3
```





# Python Typy Danych

## Czyszczenie pamięci: garbage collector

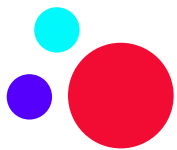
Obiekty z pamięci są usuwane po tym jak wszystkie referencje do danego obiektu zostaną zlikwidowane.

```
>>> x = 3 # utworzenie obiektu 3 i referencja do zmiennej x
>>> y = 3 # referencja do zmiennej y
>>> x = 5 # zmiana referencji zmiennej x
>>> y = 4 # tu następuje usunięcie obiektu 3 z pamięci
```

W języku Python nie ma konieczności jawnego usuwania zmiennych i zwalniania pamięci. Robi to automatycznie tzw. garbage collector.

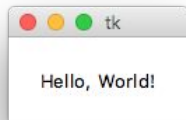
W celu wymuszenia usunięcia zmiennej (referencji do obiektu) należy użyć operatora del.

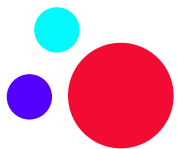
```
>>> del x
```



# Python Typy Danych

## Wprowadzenie do typów danych





# Python Typy Danych

## Funkcja print()

### 1. Wyświetlanie tekstu

```
print("Hello, World!")
```

### 2. Wyświetlanie zmiennych

```
x = 42
```

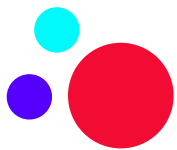
```
print(x)
```

### 3. Formatowanie tekstu

```
name = "Alice"
```

```
age = 30
```

```
print("My name is", name, "and I am", age, "years old.")
```



# Python Typy Danych

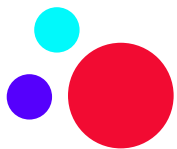
## Znaki specjalne

- Znak nowej linii

```
>>> print("Hello\nWorld!")  
Hello  
World!
```

- Znak tabulatora

```
>>> print("Hello\tWorld!")
```



# Python Typy Danych

## Komentarze

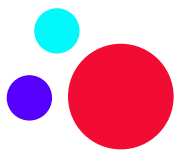
Komentarze:

- jednoliniowe `#` jednoliniowy komentarz
- wieloliniowe `'''`

Wieloliniowy komentarz

`'''`

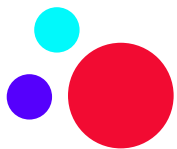




# Python Typy Danych

## Obiekty

W języku Python “wszystko jest obiektem”. Obiekt to obszar pamięci posiadający wartość. Obiekty mogą być dostarczane przez język Python i jego bibliotekę standardową lub tworzone przez programistę.



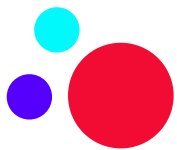
# Python Typy Danych

## Typy obiektów

W związku z koniecznością realizacji różnych zadań  
(w programowaniu zawsze wszystko sprowadza się do obliczeń)  
wygodnie jest używać różnych obiektów np.

- liczb całkowitych
- liczb "rzeczywistych"
- liczb zespolonych
- zbiorów
- list
- macierzy
- ...

Typ obiektu można sprawdzić przy pomocy funkcji `type(arg)`.



# Python Typy Danych

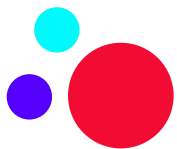
## Desygnator typu

Obiekty w Pythonie składają się z kilku elementów.  
Oprócz wartości zawierają też desygnator typu,  
dzięki czemu obiekt wie jakiego jest typu.

```
>>> x = 3
```

```
>>> type(x)
```

```
<class 'int'>
```



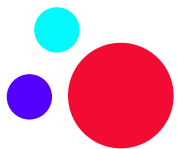
# Python Typy Danych

## Obiekty wbudowane

Obiekty wbudowane w języku Python zapewniają dostęp do podstawowych typów danych posiadających wsparcie jednostki obliczeniowej oraz do podstawowych struktur danych.

Obiekty wbudowane:

- typy liczbowe (całkowitoliczbowe, zmiennoprzecinkowe, logiczne)
- łańcuchy znaków (tekst)
- pliki
- struktury danych:
  - krotki
  - listy
  - słowniki
  - zbiory
- inne (moduły, klasy, byte code, ślady stosu)

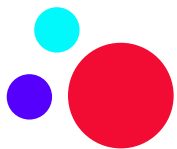


# Python Typy Danych

## Indeks

```
lista = [10, 20, 30, 40, 50]
```

Indeks    0    1    2    3    4



# Python Typy Danych

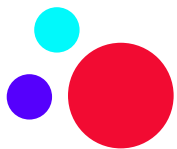
## Proporcja typów

W wyniku wykonywania operacji na różnych typach danych typ wyniku (o ile to możliwe) jest zawsze typem szerszym (o większych możliwościach).

W poniższym przykładzie do liczby całkowitej jest dodawana liczba zmiennoprzecinkowa, wynikiem jest liczba zmiennoprzecinkowa.

```
>>> x = 1 + 1.0
```

```
2.0
```



# Python Typy Danych

## Proporcja typów

Obiekt None oznacza "nic". Używa się go do reprezentowania czegoś, co nie istnieje. Poza specyficzną funkcją obiektu None używa się go tak jak innych obiektów.

```
>>> x = None
```

```
>>> x is None
```

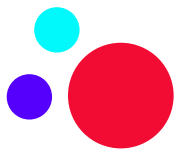
```
True
```

```
>>> y = 2
```

```
>>> y is None
```

```
False
```





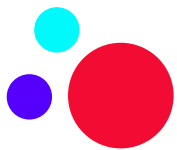
# Python Typy Danych

**Liczby całkowite: int**

Liczby całkowite są obiektami typu int.

Przykładami liczb całkowitych są: -1, 5, 10, 433453512, -435.

Liczby całkowite w Pythonie nie mają ograniczenia co do wielkości, ale duże liczby całkowite są przetwarzane inaczej (bez bezpośredniego wsparcia procesora). Na większości urządzeń zakres wartości ze wsparciem procesora to [-2147483647; +2147483647].



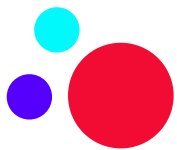
# Python Typy Danych

Definiowanie liczb całkowitych

```
>>> x = 3
```

```
>>> y = -10
```

```
>>> z = 25
```



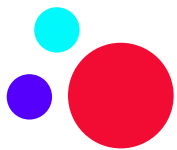
# Python Typy Danych

## Sprawdzenie typu

```
>>> x = 3
```

```
>>> type(x)
```

```
<class 'int'>
```



# Python Typy Danych

## Sprawdzenie typu

```
>>> a = 5, b = 3
```

```
>>> suma = a + b
```

```
>>> print(suma)
```

```
8
```

```
>>> a = 5, b = 3
```

```
>>> roznica = a - b
```

```
>>> print(roznica)
```

```
2
```

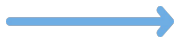
```
>>> a = 5, b = 3
```

```
>>> mnozenie = a * b
```

```
>>> print(mnozenie)
```

```
15
```

`+= / -=`

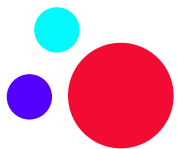


```
>>> a = 5
```

```
>>> a += 3
```

```
>>> print(a)
```

```
8
```

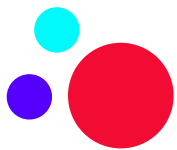


# Python Typy Danych

## Liczby zmiennoprzecinkowe: float

Liczby zmiennoprzecinkowe są obiektami typu float. Liczby zmiennoprzecinkowe są przybliżeniem liczb rzeczywistych. Przykłady liczb zmiennoprzecinkowych: 0.1, 3.2, 100000.234 itp.

Liczby zmiennoprzecinkowe mieszczą się w zakresie  $-2^{1024}$  do  $2^{1024}-1$ . Przekroczenie tego zakresu skutkuje wystąpieniem błędu.



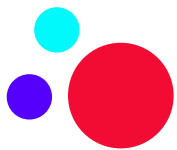
# Python Typy Danych

## Definiowanie liczb zmiennoprzecinkowych

```
>>> x = 3.0
```

```
>>> y = 2.5
```

```
<class 'float'>
```



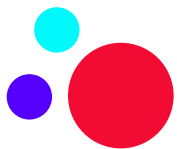
# Python Typy Danych

## Sprawdzenie typu

```
>>> x = 3.0
```

```
>>> type(x)
```

```
<class 'float'>
```

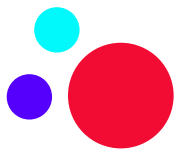


# Zadanie 5.2

Liczby całkowite i zmiennoprzecinkowe (instrukcja)

1. Oblicz sumę liczb 234, 432, 789.
2. Sprawdź czy suma z punktu 1. jest wielokrotnością liczby 5.
3. Oblicz pierwiastek kwadratowy z liczby 144.





# Python Typy Danych

## Zmienne logiczne: boolean

Zmienne logiczne są obiektami typu bool. Reprezentują wartości klasycznej logiki matematycznej: prawda (True) i fałsz (False). Wraz z operatorami logicznymi umożliwiają przeprowadzanie rachunku zdań.

```
>>> x = True
```

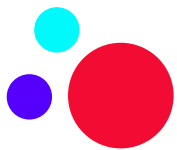
```
>>> type(x)
```

```
<class 'bool'>
```

```
>>> y = False
```

```
>>> x and y
```

```
False
```



# Python Typy Danych

## Sekwencje

Sekwencje są obiektami będącymi uporządkowanymi kolekcjami. Oznacza to, że przechowują inne elementy w sposób umożliwiający ich indeksowanie, czyli odniesienie się do danej wartości poprzez indeks. Np.:

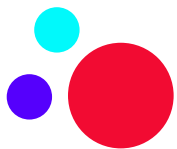
```
>>> x = "abc"
```

```
>>> x[0]
```

```
'a'
```

W języku Python sekwencje są indeksowane od wartości 0, a ostatni element  $N$  elementowej sekwencji ma indeks  $N-1$ .

Wbudowanymi sekwencjami są łańcuchy znaków, listy i krotki. Także obiekty tablic biblioteki NumPy są sekwencjami.



# Python Typy Danych

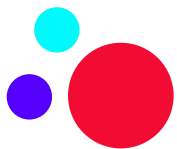
## Łańcuchy znaków: str

Łańcuchy znaków są obiektami typu str. Obiekty te służą do reprezentacji tekstu oraz dowolnych innych danych (w postaci kolekcji bajtów). Łańcuchy znaków są sekwencjami. Ponadto są zmiennymi niemodyfikowalnymi (immutable). Oznacza to, że nie można zmienić części takiej sekwencji, ale po wprowadzeniu zmiany konieczne jest zapisanie jej jako nowej zmiennej.

```
>>> x = "abc"
```

```
>>> type(x)
```

```
<class 'str'>
```

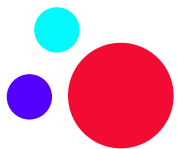


# Python Typy Danych

Definiowanie łańcucha znaków

```
>>> x = "abc"
```

```
>>> y = "Hello World!"
```



# Python Typy Danych

## Operacje na stringach

### 1. Długość łańcucha

```
tekst = "Hello, World!"
```

```
dlugosc = len(tekst)
```

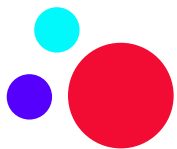
### 2. Konkatenacja

```
imie = "Ania"
```

```
nazwisko = „Nowak”
```

```
pelne_imie = imie + " " + nazwisko
```

Ania Nowak



# Python Typy Danych

## Operacje na stringach

```
tekst = "Hello World"
```

- tekst.upper()

```
>>> HELLO WORLD
```

- tekst.lower()

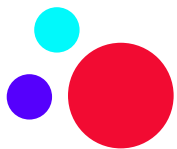
```
>>> hello world
```

- tekst.replace("Hello", "Hi")

```
>>> Hi World
```

- split() # domyślnie używa spacji jako separatora

```
>>> ['Hello', 'World']
```



# Python Typy Danych

## Wyświetlanie specjalnych znaków

Znak nowej linii

```
>>> print("Hello\nWorld!")
```

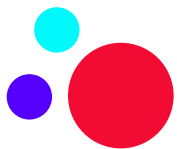
Hello

World!

Znak tabulatora

```
>>> print("Hello\tWorld!")
```

Hello World!

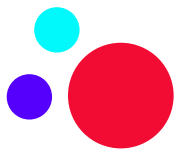


# Zadanie 5.3

## Stringi (instrukcja)

1. Stwórz kod wypisujący: "Test znaków: ' , / , " "
2. Połącz dwie zmienne imię i nazwisko, których wartości to odpowiednio Wasze imię i nazwisko.
3. Zastąp w zmiennej zawierającej imię jedną z liter na inny znak.
4. Podziel string „Cztery Pory Roku” na wyrazy.
5. Zamień ulubiony kolor na wielkie litery.





# Python Typy Danych

## Krotka: tuple

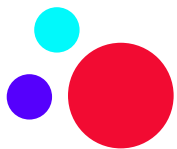
Krotka jest obiektem typu tuple. Obiekt ten jest niemodyfikowalną sekwencją elementów. Krotkę tworzy się przy użyciu nawiasów. Krotka może przechowywać elementy dowolnego typu.

```
>>> x=(1,"tekst",True,False, 0.123)
```

```
>>>
```

```
x[4]
```

```
0.123
```



# Python Typy Danych

## Lista: list

Lista jest obiektem typu list. Obiekt ten jest modyfikowalną sekwencją elementów. Listę tworzy się przy użyciu nawiasów prostokątnych. Lista może przechowywać elementy dowolnego typu.

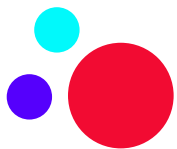
```
>>> x=[1,"tekst",True,False, 0.123]
```

```
>>> x[4] 0.123 ← Element listy o indeksie 4
```

```
>>> x[4]="zamieniony element"
```

```
>>> x=[1, 'tekst', True, False, 'zamieniony element']
```

```
>>> x[-1] 0.123 ← Ostatni  
element listy
```



# Python Typy Danych

## Operacje na listach

```
lista = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
```

1. Dodawanie elementu na koniec listy

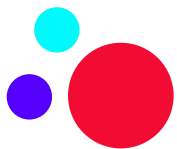
```
>>> lista.append(8)
```

```
[3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8]
```

2. Dodawanie elementu na określonej pozycji

```
>>> lista.insert(2, 10)
```

```
[3, 1, 10, 4, 1, 5, 9, 2, 6, 5, 3, 5]
```



# Python Typy Danych

## Operacje na listach

```
lista = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
```

3. Usuwanie pierwszego wystąpienia określonego elementu

```
>>> lista.remove(5)
```

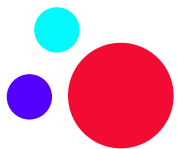
```
[3, 1, 4, 1, 9, 2, 6, 5, 3, 5]
```



4. Usuwanie elementu na określonej pozycji i zwracanie go

```
>>> lista.pop(3)
```

```
[3, 1, 4, 5, 9, 2, 6, 5, 3, 5]
```



# Python Typy Danych

## Operacje na listach

```
lista = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
```

5. Zwracanie indeksu pierwszego wystąpienia określonego elementu

```
>>> lista.index(6)
```

7

6. Zliczanie wystąpień określonego elementu w liście

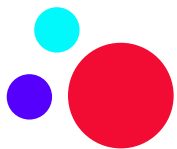
```
>>> lista.count(5)
```

3

7. Sortowanie elementów listy

```
>>> lista.sort()
```

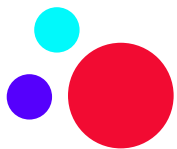
```
[1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]
```



# Zadanie 5.4

## Listy (instrukcja)

1. Zdefiniuj listę zawierającą powtarzające się elementy. Napisz program, który usuwa wszystkie powtórzenia, pozostawiając jedno wystąpienie każdego elementu, a następnie posortuj je malejąco.
2. Stwórz listę liczb całkowitych. Dodaj nowy element list na jej koniec.
3. Dla listy z punktu 1. zlicz wystąpienia wybranego elementu.
4. Dla listy z punktu 1. usuń element o indeksie 3.



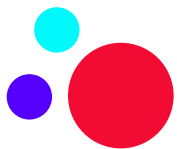
# Python Typy Danych

## Kolekcje

Kolekcje są obiektami będącymi nieuporządkowanymi kolekcjami. Oznacza to, że przechowują elementy bez określania ich kolejności. Podstawowymi elementami tego typu są zbiór i słownik.

W przypadku zbioru istotny jest sam fakt zawierania lub nie danego elementu.

Słowniki natomiast przechowują elementy wraz z kluczem, który umożliwia do nich dostęp.



# Python Typy Danych

## Zbiór: set

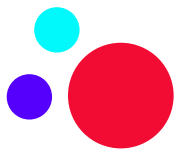
Zbiór jest obiektem typu set. Jest to nieuporządkowana kolekcją elementów. Może przechowywać elementy dowolnego typu. Istotą tego obiektu jest przechowywanie elementów oraz umożliwienie sprawdzenia czy dany element występuje w kolekcji. (Często zdarza się, że elementy są wypisywane w innej kolejności niż przy tworzeniu obiektu). Zbiory tworzy się przy użyciu nawiasów klamrowych.

```
>>> x = {1,"tekst",True,False,0.123}
```

```
>>> x
```

```
{'tekst', 1, 0.123, False}
```





# Python Typy Danych

## Operacje na listach

**zbior = {1, 2, 3}**

1. Dodawanie elementów

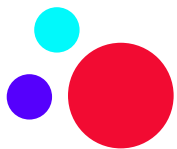
```
>>> zbior.add(4)
```

```
{1, 2, 3, 4}
```

2. Usunięcie elementu

```
>>> zbior.remove(2)
```

```
{1, 3}
```



# Python Typy Danych

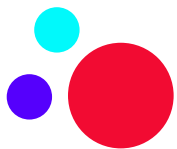
## Operacje na listach

```
zbior = {1, 2, 3}
```

3. Sprawdzenie przynależności

```
>>> 3 in zbior
```

```
True
```



# Python Typy Danych

## Operacje na listach

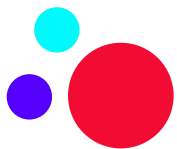
**zbior1 = {1, 2, 3}**

**zbior2 = {3, 4, 5}**

4. Operacje matematyczne – suma zbiorów

```
>>> suma_zbiorow = zbior1 | zbior2
```

```
{1, 2, 3, 4, 5}
```



# Python Typy Danych

## Operacje na listach

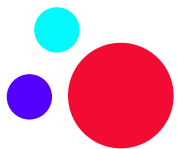
**zbior1 = {1, 2, 3}**

**zbior2 = {3, 4, 5}**

5. Operacje matematyczne – przecięcie zbiorów

```
>>> przeciecie = zbior1 & zbior2
```

```
{3}
```



# Python Typy Danych

## Operacje na listach

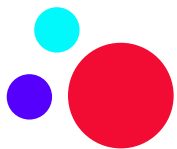
**zbior1 = {1, 2, 3}**

**zbior2 = {3, 4, 5}**

6. Operacje matematyczne – różnica zbiorów

`roznica = zbior1 - zbior2`

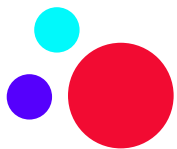
`{1, 2}`



# Zadanie 5.5

## Operacje na zbiorach (instrukcja)

1. Mając listę  $[1, 2, 4, 5, 7, 7, 7]$  wyświetl jedynie jej unikatowe wartości.
2. Mając zbiór  $\{5, 6, 10, 13, 21, 27\}$  sprawdź czy zbiór zawiera element 13.
3. Zdefiniuj zbiór zawierający kilka elementów. Napisz program, który usuwa jeden z elementów tego zbioru.
4. Zdefiniuj dwa zbiory. Napisz program, który usuwa wspólne elementy obu zbiorów, pozostawiając tylko te, które są unikalne dla każdego zbioru.



# Python Typy Danych

## Słownik: dict

Słownik jest obiektem typu dict. Słownik jest odwzorowaniem, w którym wartości są przyporządkowane kluczom.

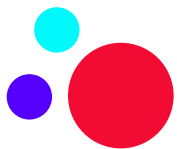
Może przechowywać elementy dowolnego typu. Istotą tego obiektu jest przechowywanie elementów, umożliwienie sprawdzenia czy dany klucz występuje w kolekcji oraz dostęp do danego elementu po kluczu.

```
>>> x = {"a":5,3:4,"key1":(1,2,3)}
```

```
>>> x
```

```
{'a': 5, 3: 4, 'key1': (1, 2, 3)}
```

```
>>> x['a']
```

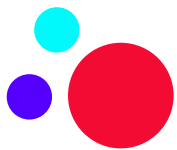


# Python Typy Danych

**Słownik: charakterystyka**

- Pary klucz-wartość
- Unikalność kluczy
- Dynamiczne i zmienne
- Nieuporządkowane
- Różne typy danych



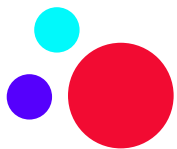


# Python Typy Danych

## Definiowanie słowników

```
sloownik = {"klucz1": "wartosc1", "klucz2": "wartosc2", "klucz3":  
"wartosc3"}
```

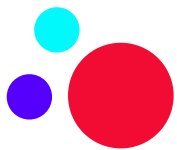
```
sloownik = dict(klucz1="wartosc1", klucz2="wartosc2",  
klucz3="wartosc3")
```



# Python Typy Danych

## Zastosowanie słowników

1. Mapowanie i Indeksowanie.
2. Przechowywanie konfiguracji.
3. Reprezentacja struktury danych w JSON.
4. Analiza i przetwarzanie danych.
5. Łączenie danych.
6. Szybki dostęp do danych.



# Python Typy Danych

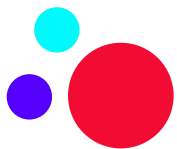
## Operatory arytmetyczne

```
slownik = {"klucz1": "wartosc1", "klucz2": "wartosc2"}
```

1

Dodawanie elementów:

- `>>> slownik["nowy_klucz"] = "nowa_wartosc"`



# Python Typy Danych

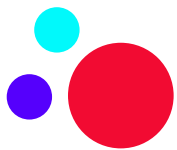
## Operatory arytmetyczne

```
slovník = {"klucz1": "wartosc1", "klucz2": "wartosc2"}
```

2

Pobieranie wartości:

- `>>> wartosc = slovník["klucz1"]`



# Python Typy Danych

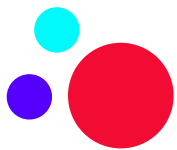
## Operatory arytmetyczne

```
slovník = {"klucz1": "wartosc1", "klucz2": "wartosc2"}
```

3

Aktualizacja wartości:

- `>>> slovník["klucz1"] = "nowa_wartosc"`



# Python Typy Danych

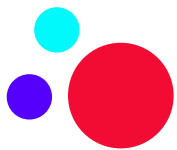
## Operatory arytmetyczne

```
slovník = {"klucz1": "wartosc1", "klucz2": "wartosc2"}
```

4

Usunięcie elementu:

- ```
>>> del slovník["klucz1"]  
{"klucz2": "wartosc2"}
```



# Python Typy Danych

## Operatory arytmetyczne

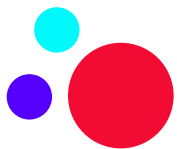
```
slovník = {"klucz1": "wartosc1", "klucz2": "wartosc2"}
```

5

Sprawdzenie elementu w słowniku:

- `>>> klucz3 in słownik`

False

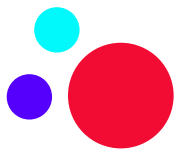


# Zadanie 5.6

## Operacje na słownikach (instrukcja)

1. Utwórz słownik ze stolicami: Hiszpanii, Włoch, Szwecji, Czech
  - A. Sprawdź stolicę Hiszpanii – w przypadku braku wyświetl „stolica nieznana”.
  - B. Dodaj stolicę Uk – Jeśli brakuje. c. Usuń stolicę Czech ze słownika
  
2. Dla słownika: dane\_osobowe = {  
    "Jan Kowalski": {"Wiek": 30, "Zawód": "Inżynier", "Miasto": „Warszawa"},  
    "Anna Nowak": {"Wiek": 25, "Zawód": "Nauczyciel", "Miasto": „Kraków"},  
    "Piotr Wiśniewski": {"Wiek": 35, "Zawód": "Lekarz", "Miasto": „Gdańsk"},
  - A. Dodaj nową osobę do słownika "Marta Nowak": {"Wiek": 22, "Zawód": "Student", „Miasto": "Poznań"}
  - B. Zmodyfikuj zawód osoby o imieniu "Anna Nowak" na „Informatyk”.
  - C. Usuń osobę o imieniu "Piotr Wiśniewski" ze słownika.





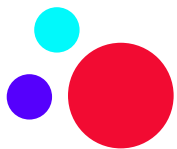
# Python Typy Danych

## Podsumowanie

Język Python oferuje wiele innych typów danych i struktur danych, np.:

- typ liczb zespolonych (complex),
- typ liczb o stałej precyzji (decimal),
- typ liczb ułamkowych (fractions),
- obiekt daty i czasu (datetime),
- krotka z nazwanymi polami (named\_tuple),
- sterta (heap\_queue),
- itd.

Ponadto istnieje wiele specjalistycznych bibliotek oferujących dodatkowe funkcjonalności. np.: zbilansowane drzewa binarne.



# Python Typy Danych

## Operator

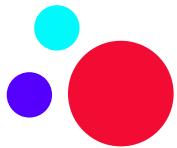
Operatory służą do wykonywania operacji na typach danych.

Operatorami są np.:

- operator dodawania +,
- odejmowania −,
- mnożenia \*,
- dzielenia /,
- itd.

```
>>> 2.0 * 0.56 # mnożenie  
1.12
```

Ponadto niektóre operatory mają zdefiniowane działania dla różnych typów danych np. operator mnożenia użyty dla tekstu i liczby całkowitej.



# Python Typy Danych

## Operatory arytmetyczne

1

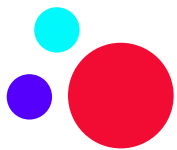
Dodawanie +

```
>>> a = 5
>>> b = 3
>>> wynik = a + b
>>> print(wynik)
8
```

```
>>> a = 5
>>> b = 3.0
>>> wynik = a + b
>>> print(wynik)
8.0
```

```
>>> a = "Hello"
>>> b = "World"
>>> wynik = a + b
>>> print(wynik)
"Hello World"

>>> lista1 = [1, 2, 3]
>>> lista2 = [4, 5, 6]
>>> wynik = lista1 + lista2
>>> print(wynik)
[1, 2, 3, 4, 5, 6]
```



# Python Typy Danych

## Operatory arytmetyczne

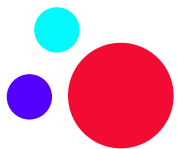
2

Odejmowanie –

```
>>> a = 8
>>> b = 3
>>> wynik = a - b
>>> print(wynik)
5
```

```
>>> a = 8
>>> b = 3.0
>>> wynik = a - b
>>> print(wynik)
5.0
```

infoShare  
ACADEMY



# Python Typy Danych

## Operatory arytmetyczne

3

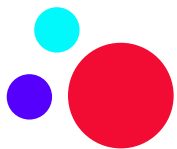
Mnożenie \*

```
>>> a = 4
>>> b = 6
>>> wynik = a * b
>>> print(wynik)
24
```

```
>>> tekst = "Python"
>>> wynik = tekst * 3
>>> print(wynik)
"PythonPythonPython"
```

```
>>> a = 4
>>> b = 6.0
>>> wynik = a * b
>>> print(wynik)
24.0
```

```
>>> lista = [1, 2, 3]
>>> wynik = lista * 2
>>> print(wynik)
[1, 2, 3, 1, 2, 3]
```



# Python Typy Danych

## Operatory arytmetyczne

4

Dzielenie /

```
>>> a = 10
```

```
>>> b = 2
```

```
>>> wynik = a / b
```

```
>>> print(wynik)
```

```
5.0
```

```
>>> a = 10.0
```

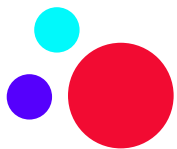
```
>>> b = 2
```

```
>>> wynik = a / b
```

```
>>> print(wynik)
```

```
5.0
```

info **Share**  
ACADEMY



# Python Typy Danych

## Operatory arytmetyczne

info **Share**  
ACADEMY

5

Dzielenie całkowite //

```
>>> a = 10
```

```
>>> b = 3
```

```
>>> wynik = a // b
```

```
>>> print(wynik)
```

```
3
```

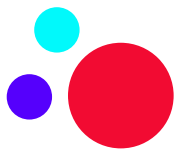
```
>>> a = 10.0
```

```
>>> b = 3
```

```
>>> wynik = a // b
```

```
>>> print(wynik)
```

```
3.0
```



# Python Typy Danych

## Operatory arytmetyczne

info **Share**  
ACADEMY

6

Reszta z dzielenia %

```
>>> a = 10
```

```
>>> b = 3
```

```
>>> reszta = a % b
```

```
>>> print(reszta)
```

```
1
```

```
>>> a = 10.0
```

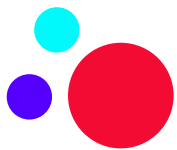
```
>>> b = 3
```

```
>>> reszta = a % b
```

```
>>> print(reszta)
```

```
1.0
```





# Python Typy Danych

## Operatory arytmetyczne

info **Share**  
ACADEMY

7

Potęgowanie \*\*

```
>>> a = 2
```

```
>>> b = 3
```

```
>>> potega = a ** b
```

```
>>> print(potega)
```

```
8
```

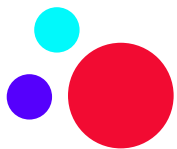
```
>>> a = 2.0
```

```
>>> b = 3
```

```
>>> potega = a ** b
```

```
>>> print(potega)
```

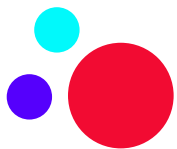
```
8.0
```



# Zadanie 5.7

## Operatory arytmetyczne (instrukcja)

- Stwórz dwie zmienne zmiennoprzecinkowe `x` i `y`. Oblicz ich sumę, różnicę, iloczyn i iloraz. Wyświetl wyniki.
- Stwórz zmienną `liczba = 21`. Oblicz wartość tej liczby podniesionej do kwadratu. Oblicz pierwiastek kwadratowy z tej liczby.
- Stwórz dwie zmienne typu string `tekst1 = Machine` i `tekst2 = Learning`. Połącz te dwa ciągi znaków w jedno wyrażenie i wyświetl wynik. Wykorzystaj operator powielania (`*`) do powtórzenia jednego z ciągów znaków.

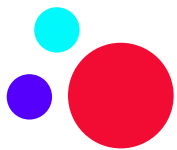


# Zadanie 5.8

## Zadanie podsumowujące (instrukcja)

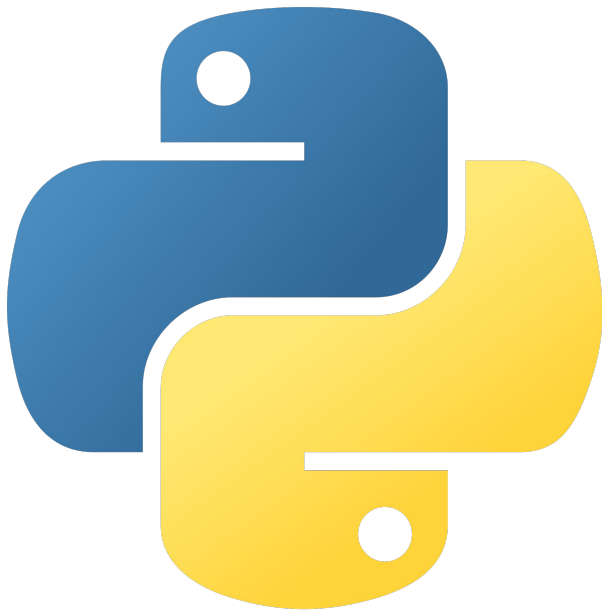
1. Parametr `opлата_za_wymiane` zapisywany jest jako liczba całkowita. Należy go przeliczyć na liczbę dziesiętną, dzieląc go przez 100. Np. jeżeli  $1,00 \text{ EUR} == 1,20 \text{ USD}$ , a `opлата_za_wymiane` wynosi 10, to rzeczywisty kurs wymiany będzie wynosić:  $1,00 \text{ EUR} == 1,32 \text{ USD}$ , ponieważ 10% z 1,20 to 0.12 i jest ono doliczane do opłaty. Pamiętaj, że nominat waluty jest liczbą całkowitą i nie można jej dzielić.

- Napisz równanie obliczającą maksymalną wartość jaką możesz wymienić w kantorze.
  - Dla zmiennych:
    - `budget` – pieniądze, które planujesz wymienić
    - `kurs_wymiany` – wartość jednostkowa waluty obcej
    - `opлата_za_wymiane` – % który jest traktowany jako opłata za wymianę
    - `denominacja` – nominat waluty
  - Szukana zmienna:
    - `max_wymiany` – maksymalna wartość jaką możesz wymienić
2. Stwórz listę, która zawiera parametr o nazwie `liczba`, a jej drugi i trzeci parametr są większe od poprzedniego o równowartość swojego indeksu w liście.



# Python Typy Danych

Podsumowanie



[infoShareAcademy.com](https://infoShareAcademy.com)

**info** Share  
ACADEMY