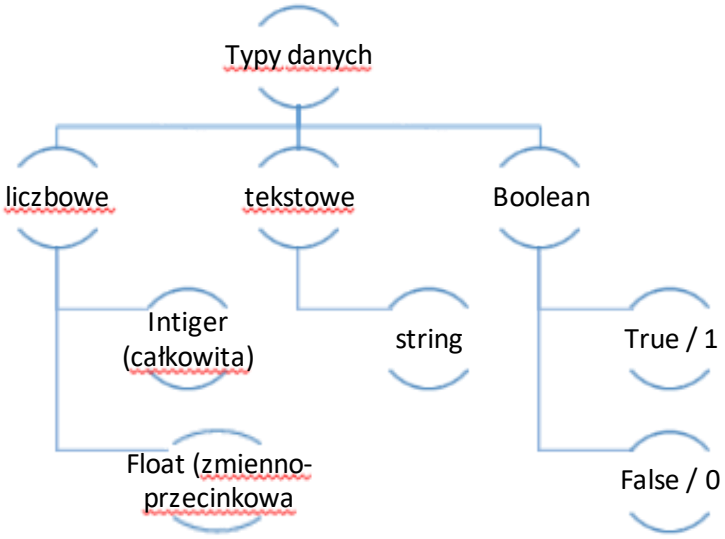


TWORZENIE ZMIENNYCH

```
# definiowanie zmiennej
zmienna = 5
zmienna = 'napis'
zmienna = 7.5
zmienna = True
```

```
# sprawdzanie typu
type(zmienna)
# konwertowanie typów
zmienna = int(zmienna)
zmienna = str(zmienna)
zmienna = float(zmienna)
zmienna = 'napis'
zmienna = float(zmienna) # tu będzie błąd
zmienna = 4.5
zmienna = int(zmienna) # tu będzie 4, bo zaokrągliło nam wartość po przecinku
```

OPERACJE NA LICZBACH (int, float)

```
wynik = 2+2
wynik = 2*3
wynik = (2+2**2*4)/9
wynik = (2+2**2*4)/0 # błąd - brak możliwości dzielenia przez zero
reszta = 13%2
reszta = 9%7
```

```
# zadziała też tak
a = 4
b = 5
wynik = a*b
```

OPERATORY LOGICZNE

```
a > b
a == b
a != b
```

```
imie == 'Jan' and nazwisko == 'Kowalski'
imie == 'jan' or nazwisko == 'Kowalski'
(imie == 'jan') & (nazwisko == 'Kowalski') # and
(imie == 'jan') | (nazwisko == 'Kowalski') # or
```

WYRAŻENIE WARUNKOWE IF

```
kwota = 100
if kwota<100: # wyrażenie jest prawdziwe
        print('kwota poniżej 100')
elif kwota>100: # poprzednie wyrazenie jest falszywe, ale to prawdziwe
        print('kwota powyzej 100')
else: # jesli zadne z powyzszych nie jest prawdziwe
        print('kwota ==100')

# można też sprawdzać, czy dany obiekt nie jest pusty
if '':
        print('True')
else:
        print('false')

# to samo z zerem lub false
# słowo not zmienia true na false

if not '':
        print('True')
else:
        print('false')
```

OPERACJE NA STRINGACH

```
imie = jan
nazwisko = kowalski
dane_osobowe = imie + nazwisko
dane_osobowe = imie + " " + nazwisko
napis = "Client's details"
len(napis)
napis.index("l") #1
napis[1]
piece_of_napis = napis[1:4]
len(piece_of_napis)
napis[1:]
napis[:-1]
napis[-4:-1] #'ail'
napis[::2] # "Cin' eal"
napis.count("l") # 2
napis = napis.upper()
napis = napis.lower()
napis = napis.title()
napis.startswith("C") # True
napis.endswith("a") #False
```

```
'a' in 'napis'
napis.islower()
napis.isnumeric()
'napis'.replace('a', 'o')

print(napis)
imie = "Jan"
nazwisko = "Kowalski"
wartosc = 400
napis = 'klient ' + imie + " " + nazwisko + " posiada
kapital " + wartosc + '.'
napis = f'klient {imie} {nazwisko} posiada kapital
{wartosc}."
napis = "klient {} {} posiada kapital {}.".format(imie,
nazwisko, wartosc)
# stary sposób
napis = "klient %s %s posiada kapital %d." %(imie,
nazwisko, wartosc)
        print('false')

# tworzenie szeregu znakow
a = 's'*8
```

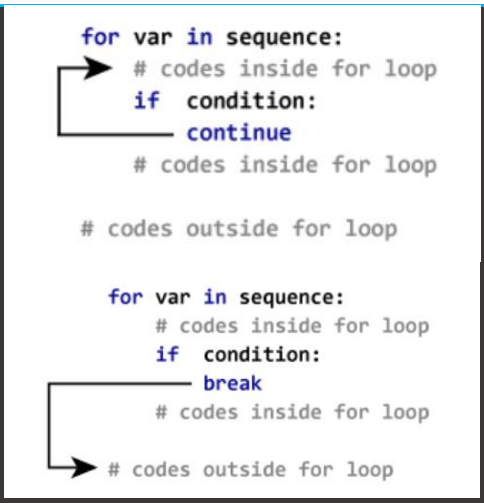
LISTY I TUPLE

```
lista = []
lista = list()
lista.append(1)
lista.append('dwa')
lista.append([1,2,3])
lista.insert(0,'dodany')
lista.extend([4,5,6])
del lista[0]
lista.remove(1)

lista[0]
lista[1]
lista[10] # tu error - nie ma takiego elementu

tupla = (1,2,3)
tupla[0]
tupla[0] = 10 # error
tupla.append(3) # error
del tupla[0] # error
```

```
lista[1:3]
lista[-1]
tupla[-1]
tupla[0:2]

#przydatne operacje
lista = [1,1,1,1,2,3,4,4,4]
lista_bez_powtorzen = set(lista)
lista = [6,7,4,3,7]
lista.sort()
','.join(lista)

# tworzenie listy tych samych elementów
a = [1]*8

max([1,2,3])
min([1,2,3])
sum([1,2,3])
```

PĘTLA FOR

```
for x in lista:
        print(x)

lista_2 = []
for x in lista:
        lista_2.append(x*2)

# tworzenie zakresu poprzez range
for x in range(10): od 2 do 9
        print(x)

for x in range(2,10,2): # od 2 do 9 co dwa elementy
        print(x)


# enumerate i zip
name_list = ['adam', 'anna', 'hanna']
for num, name in enumerate(name_list):
        print(num, name_list)

city_list = ['warsaw', 'london', 'new york']
for name, city for zip(name_list, city_list):
        print(name, city)
```



FUNKCJE

```
# definiowanie funkcji
def sum_numbers():
        print(2+2)

# wywoływanie funkcji
sum_numbers

def sum_numbers(number_1, number_2):
        print(number_1 + number_2)

sum_numbers(1,2)

def sum_numbers(number_1, number_2):
        return number_1 + number_2

number = sum_numbers(1, 2)

num_1 = input()

def divide_numbers(number_1, number_2):
  if(number_2 == 0):
    return 'Cannot divide' # zakoncz funkcje nic nie zwracajac
  else:
    return number_1 / number_2

number = divide_numbers(1, 2)
number = divide_numbers(1, 0)


# zmienna suma wystepuje jedynie w funkcji
def funk1():
        suma = 2+1

suma # nie ma takiej zmiennej

# dokumentacja
def calc_perc(num1, num2):
        """
        calculate percentage # tu opisujemy funkcje
        """
        return num1*num2/100
```

```
# zwracanie wielu argumentów
def fukcja_wielu(num1, num2):
        return (num1+num2, num1*num2,
num1/num2)

# wartości domyslne:
def calculate_sum(arg1, arg2=100):
        return arg1+arg2

# możemy dodać wiele niezdefiniowanych
argumentów do funkcji
def myFun(*argv):
  for arg in argv:
    print (arg)

myFun('Ala', 'ma', 'kota')

# możemy dodać argumenty w postaci klucz :
wartość
def myFun(**kwargs):
  for key, value in kwargs.items():
    print (f"{key} : {value}")


# mozemy tez zwracac wiele argumentow z funkcji
za pomocą tupli
def fun1(a,b):
        return (a*b, a+b)

c,d = fun1(1,2)

# rekurencja – funkcja wywołuje samą siebie
def odliczanie(n):
  if n > 0:
    print( "----------" )
    odliczanie (n-1)
```

SŁOWNIK

```
kontakty = {}
kontakty["Jan"] = 938477566
kontakty["Jacek"] = 938377264
kontakty["Janusz"] = 947662781

kontakty = {
   "Jan" : 938477566,
   "Jacek" : 938377264,
   "Janusz" : 947662781
}

# pobieranie elementów
kontakty['Janusz']
kontakty.get('Janusz')
# indeksowanie tylko po kluczu a nie po indeksie
kontakty[0]
# kluczem w słowniku nie może być lista bo jest mutowalna
```

```
# iterowanie po słowniku - nie bedzie po kolei:
for imie, numer in kontakty.iteritems():
        print(imie, numer)

for imie in kontakty:
   print(imie, kontakty[imie])

for imie in kontakty.keys():
        print(imie)

for numer in kontakty.values():
        print(numer)

# usuwanie
del kontakty["Jan"]

# przeszukiwanie
'Julita' in kontakty # sprawdza czy klucz jest w
słowniku

# mieszanie typów danych:
s1 = {'a':2, (1,2):'wartosc'}
s2 = {123: [1,2,3]}
lista_slownikow = [s1, s2]
```

LIST COMPREHENSION

```
a = [x for x in range(10)]
a = [x*2 for x in range(10)]

lista1 = [1,2,3]
lista2 = [4,5,6]

a = [x+y for x,y in zip(lista1,lista2)]
```

```
#wszystkie podzielne przez 2 :
a = [x for x in range(10) if x%2 ==0]
[x for x in range(10) if (x % 3 == 0 and x % 2 == 0) ]

#wszystkie podzielne przez 2 lub jesli nie to zero:
a = [x if x%2 ==0 else 0 for x in range(10)]
```

LAMBDA

```
# pozwala stworzyć anonimową funkcję z wyrażenia

funkcja_lambda = lambda x,y: x+y

# jest rownowazne
def funkcja_lambda(x,y):
   return x+y
```

```
# lambda moze byc uzywana z wbudowanymi
funkcjami typu filter() czy map()
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
final_list = list(filter(lambda x: (x%2 != 0) , li))

li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
final_list = list(map(lambda x: x*2 , li))
```

OBSŁUGA WYJĄTKÓW

```
def division():
  try:
    a = 2
    b = 0
    print(a/b)
  except Exception as e:
    print("blad ", e)

try:
 print(x)
except NameError:
 print("Variable x is not defined")
except:
 print("Something else went wrong")

try:
 print("Hello")
except ZeroDivisionError as e:
 print("Something went wrong")
else: # jesli nie ma błędu
 print("Nothing went wrong")
```

```
try:
 print(x)
except:
 print("Something went wrong")
finally: # wykona sie niezależnie od błędu
 print("The 'try except' is finished")


# mozemy tez sami decydowac kiedy chcemy
wywołac błąd
x = -1
if x < 0:
 raise Exception("Sorry, no numbers below zero")


x = "hello"
if not type(x) is int:
 raise TypeError("Only integers are allowed")

# lista wszystkich wyjątkow wbudowanych
https://docs.python.org/3/library/exceptions.html
```

DODATKOWE BIBLIOTEKI

```
import string
string.ascii_letters
string.digits

import statistics
dir(statistics)
help(statistics.mean)

# dwa sposoby importowania modułow
import matplotlib.pyplot as plt
from statistics import mean

mean([1,4,4])

import random

random.choice([1,2,3]) # wybiera jedną wartość
random.choices([1,2,3], k=6) # losowanie ze zwracaniem
random.randint(1,4) # do 4 włącznie
random.randrange(1,5)
```

```
lista = [1,2,3]
random.shuffle(lista)

# ustawianie ziarna losowania jako stałe pozwala
uzyskac zawsze taki sam wynik losowania
for i in range(5):
   random.seed(25)
   print(random.choice(lista))

import collections
a = collections.Counter([1,1,1,1,3,3,4,4,4,4,5,5,5])
a.most_common(1) # zwraca najpopularniejszy
element w zbiorze

import itertools
iloczyn_kartezjanski =
list(itertools.product([1,2,3],['Asia','Kasia','Basia']))
```