Python - powtórzenie

Lista

Tworzenie pustej listy: **lista = [ ]**

```
lista = []
print(lista)
```

```
[]
```

Listy składane (*list comprehension*) - pozwala na szybkie tworzenie listy o określonych elementach

```
#Tworzenie listy z liczbami od 1 do 10
liczby = [i for i in range(1, 11)]
print(liczby)

#Tworzenie listy z potęgami liczby 2
potegi = [2**i for i in range(0,11)]
print(potegi)

#Tworzenie listy z literami alfabetu
alfabet = [chr(i) for i in range(65, 91)]
print(alfabet)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S'
```

.append(wartość) - dodaje element do listy

```
lista.append(6)
lista.append(2)
lista.append(3)
print(lista)
```

```
[6, 2, 3]
```

.count(wartość) - zwraca liczbe elementów o danej wartości

```
print(lista.count(6))
```

```
1
```

.reverse() - odwraca liste

```
lista.reverse()
print(lista)
```

```
[6, 2, 3]
```

.sort() - sortuje liste

```
lista.sort()
print(lista)
```

```
[2, 3, 6]
```

.insert(indeks, wartość) - dodaje element na konkretny indeks w liście

```
lista.insert(2, 3)
print(lista)
```

```
[2, 3, 3, 6]
```

.pop(indeks) - usuwa element z konkretnego indeksu

```
lista.pop(1)
print(lista)
```

```
[2, 3, 6]
```

.clear() - czyści liste

```
lista.clear()
print(lista)
```

```
[]
```

Słownik

Tworzenie słownika: **slownik = {klucz1 : wartosc1, klucz2 : wartosc2, klucz3 : wartosc3}**

```
slownik = {1 : "lampa", 2 : "stol", 3 : "krzeslo"}
print(slownik)
```

```
{1: 'lampa', 2: 'stol', 3: 'krzeslo'}
```

Możemy również tworzyć słownik z listy

```
lista = ["kot", "malpa", "pies"]
slownik = dict.fromkeys(lista, 0)
print(slownik)
```

```
{'kot': 0, 'malpa': 0, 'pies': 0}
```

Słownik usuwa duplikaty

```
lista = ["kot", "malpa", "pies", "kot", "goryl", "pies"]
slownik = dict.fromkeys(lista, 0)
print(slownik)
```

```
{'kot': 0, 'malpa': 0, 'pies': 0, 'goryl': 0}
```

Liczenie poszczególnych elementów z listy

```
for element in lista:
  slownik[element]+=1
print(slownik)
```

```
{'kot': 2, 'malpa': 1, 'pies': 2, 'goryl': 1}
```

Zbiór

Zbiór nie przechowuje duplikatów

```
zbior = set(["Mariusz","Jakub", "Mariusz", "Tomek"])
print(zbior)
```

```
{'Tomek', 'Mariusz', 'Jakub'}
```

Krotka

Krotka charakteryzuje sie tym, że jest niezmienna. Nie możemy do niej dodawać nowych elementów ani usuwać z niej istniejących.

Tworzenie krotki: **krotka = ()**

```
krotka = ("Jan", 6, True, 181.2)
print(krotka)
```

```
('Jan', 6, True, 181.2)
```

String

[start:stop:step]

```
napis = "przykladowy napis"
print(napis[3:6])
print(napis[5:])
print(napis[:10])
print(napis[2:10:2])
```

```
ykl
ladowy napis
przykladow
zkao
```

Odwracanie napisu

```
napis = "sok"
odwrocony = napis[::-1]
print(odwrocony)
```

```
kos
```

Zmiana wielkości znaków w napisie

```
napis = "Znaki Różnej Wielkości"
print(napis.upper())
print(napis.lower())
```

```
ZNAKI RÓŻNEJ WIELKOŚCI
znaki różnej wielkości
```

.count() - zlicza ilość wystąpienia danej litery w ciągu znaków

```
napis = "matura jest prosta"
print(napis.count("a"))
```

```
3
```

.index() - zwraca pierwszy index podanej przez nas litery w ciągu

```
napis = "matura jest prosta"
print(napis.index("a"))
```

```
1
```

Funkcje matematyczne

math.sqrt()

Zwraca pierwiastek z danej liczby po dodaniu modułu **math**

```
import math
print(math.sqrt(3))
```

```
1.7320508075688772
```

math.ceil()

Zaokrąglenie w góre do liczby całkowitej

```
math.ceil(3.7)
```

```
4
```

math.floor()

Zaokrąglenie w dół do liczby całkowitej

```
math.floor(3.5)
```

```
3
```

Funkcje wbudowane

ord() - funkcja pozwala sprawdzić kod ASCII danego znaku

```
print(ord('A'))
print(ord('a'))
print(ord(' '))
print(ord('/'))
```

```
65
97
32
47
```

chr() - służy do zmiany liczby kodu ASCII na znak

```
print(chr(97))
print(chr(103))
```

```
a
g
,
```

int() - możemy nie tylko zmieniać stringa w liczbę całkowitą, ale przy tym zmienić dowolny system liczbowy na dziesiętny. Składnia funkcji wygląda następująco: *int(string_w_danym_systemie_liczbowym, system_liczbowy)*

```
print(int("1010",2))
print(int("3210",4))
print(int(str(25),8))
```

```
10
228
21
```

bin(); oct(); hex() - funkcje pozwalające zamienić liczby na kolejno: system dwójkowy, ósemkowy oraz szesnastkowy

```
print(bin(10))
print(oct(10))
print(hex(10))
```

```
0b1010
0o12
0xa
```

Warto zwrócić uwagę, że dwa pierwsze znaki odpowiadają za prefix, tj. **0b** - binarny; **0o** - oktalny; **0x** - heksadecymalny. Możemy się tego łatwo pozbyć np. przypisując wartości do poszczególnych zmiennych bez dwóch pierwszych char'ów

```
binarny = bin(10)
binarny = binarny[2:]

print(binarny)
```

```
1010
```

map() - za pomocą tej funkcji możemy zastosować daną funkcję osobno na każdym elemencie np. listy

```
lista = ['1223', '1526', '457468', '233462', '23522']
lista = list(map(int, lista))

print(lista)
```

```
[1223, 1526, 457468, 233462, 23522]
```

Random

.random() - losowa liczba float od 0.0 do 1.0

```
import random
print(random.random())
```

.randint(a, b) - losowa liczba int z przedziału **a** - **b**

```
print(random.randint(5, 10))
print(random.randint(100, 200))
```

.uniform(a, b) - losowa liczba **float** z przedziału od **a** do **b**

```
print(random.uniform(1, 5))
```

```
2.3173968569915564
```

Dzielenie na voidy

Dobrą praktyką, która nie tylko pomoże egzaminatorowi ale i nam w szukaniu błędów w kodzie, jest dzielenie zadań maturalnych na puste funkcje tzw. voidy.

```
def zadanie1():

  plik = open("dane.txt", "r")
  zapis = open("zadanie4.txt", "a")

  for linia in plik:
    liczba = int(linia.strip())
    if liczba % 2 == 0:
      parzysta+=1
  print("4.1", parzysta, file=zapis)


def zadanie2():
  #kod do drugiego zadania

zadanie1()
zadanie2()
```