
	Politechnika Bydgoska im. J. J. Śniadeckich  <b>Wydział Telekomunikacji, Informatyki i Elektrotechniki</b>		
<b>Przedmiot</b>	Skryptowe języki programowania		
<b>Prowadzący</b>	mgr inż. Martyna Tarczewska		
<b>Temat</b>	Struktury danych języka Python		
<b>Student</b>	Marcin Ogórkiewicz		
<b>Nr ćw.</b>	3	<b>Data wykonania</b>	25.10.2023
<b>Ocena</b>		<b>Data oddania spr.</b>	25.10.2023

### Zadanie 1

```
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310
```

```
# importy
import typing
```

```
# stałe i zmienne globalne
my_list = ["one", "two", "three"]
# funkcje
```

```
def zadanie1a(a_list: list) -> None:
    print("Zadanie 1 a)\n")
    print(a_list, "\n")
```

```
def zadanie1b(b_list: list) -> list:
    for i in range(5):
        b_list.append(i)
    print("\n")
    return b_list
```

```
def zadanie1c(c_list: list) -> None:
    print("Zadanie 1 c)\n")
    for i in range(2):
        print(c_list[i])
    print("\n")
    for i in range(len(c_list)-1, len(c_list)-3, -1):
        print(c_list[i])
    print("\n")
```

```
def zadanie1d(d_list: list) -> None:
    print("Zadanie 1 d)\n")
    print(len(d_list), "\n")
```

```
def zadanie1e(e_list: list) -> None:
```

```

print("Zadanie 1 e)\n")
for i in range(0, len(e_list), 2):
    print(e_list[i])
print("\n")

def zadanie1f(f_list: list) -> None:
    f_list.append(10)

def zadanie1g(g_list: list) -> None:
    g_list.append("napis")

def zadanie1h(h_list: list) -> None:
    print("Zadanie 1 h)\nLista przed posortowaniem\n", h_list)
    sorted(h_list)
    print("Lista po posortowaniu\n", h_list, "\n")

def zadanie1i(i_list: list) -> list:
    del i_list[-1]
    return i_list

def zadanie1j(j_list: list) -> None:
    print("Zadanie 1 j)\nLista przed posortowaniem\n", j_list)
    sorted(j_list, reverse=True)
    print("Lista po posortowaniu\n", j_list, "\n")

def zadanie1k(k_list: list) -> list:
    k_list.insert(2, "2")
    return k_list

def zadanie1l(l_list: list) -> None:
    print("Zadanie 1 l)\n")
    counter_13 = 0
    for i in range(len(l_list)):
        if l_list[i] == 13:
            counter_13 += 1
    print("Na liście występuje ", counter_13, " elementów o wartości 13\n")

def main() -> None:
    zadanie1a(my_list)
    zadanie1b(my_list)
    zadanie1c(my_list)
    zadanie1d(my_list)
    zadanie1e(my_list)
    zadanie1f(my_list)
    zadanie1g(my_list)
    #zadanie1h(my_list)
    zadanie1i(my_list)
    #zadanie1j(my_list)
    zadanie1k(my_list)
    zadanie1l(my_list)

"""a.Nie, ponieważ nie da się sortować list zawierających typy numeryczne i
znakowe jednocześnie.
b.W pythonie listy są numerowane od indeksu 0."""

```

```
main()
```

## Zadanie 2

```
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing

# stałe i zmienne globalne

# funkcje

def zadanie2() -> None:
    print("Zadanie 2\n")
    user_list = []
    for i in range(10):
        print("Podaj liczbę nr", i)
        user_number = int(input())
        user_list.append(user_number)
    print("Twoja lista\n", user_list)
    sorted_user_list = sorted(user_list)
    print("Największy element: ", sorted_user_list[9],
          "\nNajmniejszy element listy: ", sorted_user_list[0], "\n")
    non_negative_user_list = []
    for i in range(len(user_list)):
        if user_list[i] >= 0:
            non_negative_user_list.append(user_list[i])
    avg = sum(non_negative_user_list)/len(non_negative_user_list)
    print("Średnia liczb nieujemnych na liście: ", avg)

def main() -> None:
    zadanie2()

main()
```

## Zadanie 3

```
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing

# stałe i zmienne globalne

# funkcje

def zadanie3() -> None:
    list_1 = [1, 2, 3, 4, 5]
    list_2 = [1, 3, 5, 7, 9]
    list_1_unique = []
    for i in range(len(list_1)):
        for j in range(len(list_2)):
            if list_1[i] == list_2[j]:
                break
            elif j == 4:
                list_1_unique.append(list_1[i])
```

```

    print("Zadanie3\nLista 1: ", list_1, "\nLista 2: ", list_2, "\nUnikalne
elementy listy 1: ", list_1_unique)

def main() -> None:
    zadanie3()

main()

```

#### Zadanie 4

```

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing

# stałe i zmienne globalne

# funkcje

def zadanie4() -> None:
    list_4 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    list_4_odd = []
    print("Zadanie4\nLista: ", list_4, "\n")
    for i in range(len(list_4)):
        if list_4[i] % 2 != 0:
            list_4_odd.append(list_4[i])
    sorted_list_4_odd = sorted(list_4_odd)
    print("Lista elementów nieparzystych: ", list_4_odd, "\nNajmniejszy
element nieparzysty: ", sorted_list_4_odd[0])

def main() -> None:
    zadanie4()

main()

```

#### Zadanie 5

```

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing

# stałe i zmienne globalne

# funkcje

def zadanie5() -> None:
    list_A = ["A", "B", "C"]
    list_B = ["D", "E", "F"]
    list_A += list_B
    print("Zadanie5\na) ", list_A)
    list_A = list_A[0:3:1]
    list_A = list_B+list_A
    print("\nb) ", list_A)

```

```
def main() -> None:
    zadanie5()
```

```
main()
```

## Zadanie 6

```
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing

# stałe i zmienne globalne

# funkcje

def zadanie6() -> str:
    num_list = []
    while True:
        num = int(input("Podaj liczbę (0 kończy wprowadzanie): "))
        if num == 0:
            break
        elif num_list.count(num) == 0:
            num_list.append(num)
        else:
            for i in range(num_list.count(num)):
                num_list.remove(num)
    unique_elements = set(num_list)
    print(f"Unikatowe elementy listy: {unique_elements}")

def main() -> None:
    zadanie6()

main()
```

## Zadanie 7

```
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing

# stałe i zmienne globalne

# funkcje

def main() -> None:
    tuple = ("apple", "banana", "cherry")
    tuple_b = ("orange",)
    tuple += tuple_b # dodawanie krotek
    multi_tuple = tuple * 2 # mnożenie krotek
    print(len(tuple)) # długość krotki - liczba elementów
    for x in tuple: # wypisanie wszystkich elementów krotki
        print(x)

main()

"""a. Krotki tworzymy okrągłymi nawiasami, a po utworzeniu nie można
```

modyfikować ich zawartości.

b. Krotka zostanie powielona, zostaną do niej dopisane jej własne elementy i zwiększy swoją długość dwukrotnie.

c. Tak samo jak dodawanie; ilość powieleń krotki w samej sobie zależy od tego, przez jaką liczbę ją przemnożymy.

d. Przecinek daje pythonowi informację, że tuple\_b jest krotka, bez przecinka interpretuje jego zawartość jako string.

e. Tak."""

## Zadanie 8

```
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing

# stałe i zmienne globalne

# funkcje

def zadanie8() -> None:
    the_set = {"orange", "lychee", "watermelon", "avocado", "tomato"} #
definiowanie zbioru
    print("Zadanie8\nZbiór początkowy")
    the_set.remove("watermelon")
    the_set.discard("orange")
    print("\na) po usuwaniu metodami remove i discard: ", the_set)
    x = the_set.pop() # wyjęcie ze zbioru jakiegoś elementu
    print("\nb) metoda pop: ", x, "\nlista:\n", the_set)
    """Metody z zadania a) usuwają tylko istniejące elementy,
    zwracają błąd przy próbie usunięcia nieistniejącego elementu oraz psują
    wynik końcowy programu.
    Metoda z zadania b), pop(), usuwa zawsze inny element zbioru, jeśli nie
    podamy żadnego indeksu"""

def main() -> None:
    zadanie8()

main()
```

## Zadanie 9

```
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing

# stałe i zmienne globalne

# funkcje

def zadanie9() -> None:
    this_set = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
    the_set = {7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17}
    print("Zadanie 9\na) ", the_set.isdisjoint(this_set),
type(the_set.isdisjoint(this_set)),
        "\nb) ", the_set.issubset(this_set),
type(the_set.issubset(this_set)),
```

```

        "\nc) ", the_set.issuperset(this_set),
type(the_set.issuperset(this_set)),
        "\nd) ", the_set.union(this_set), type(the_set.union(this_set)),
        "\ne) ", the_set.difference(this_set),
type(the_set.difference(this_set)),
        "\nf) ", the_set.intersection(this_set),
type(the_set.intersection(this_set)))

def main() -> None:
    zadanie9()

main()

```

## Zadanie 10

```

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310

# importy
import typing

# stałe i zmienne globalne

# funkcje

def zadanie10() -> None:
    rndm_dict = {
        "fruit": "strawberry",
        "vegetable": "carrot",
        "candy": "fudge",
        "beverage": "beer"
    }
    print("Zadanie 10\nSłownik początkowy:\n", rndm_dict)
    rndm_dict["kitchenware"] = "spoon"
    print("\na):\n", rndm_dict)
    rndm_dict["fruit"] = "mango"
    print("\nb):\n", rndm_dict)
    pop = rndm_dict.pop("candy")
    popitem = rndm_dict.popitem()
    print("\nc) metoda pop: ", pop, ", metoda popitem", popitem)
    rndm_dict_2 = {
        1: "strawberry",
        "vegetable": "carrot",
        3: "fudge",
        "beverage": "beer"
    }
    rndm_dict_3 = {
        "fruit": "strawberry",
        "vegetable": 2,
        "candy": "fudge",
        "beverage": 4
    }
    print("\nd) Możliwe jest użycie w jednym słowniku kluczy o dwóch
różnych typach."
        "Możliwe jest również użycie w jednym słowniku wartości o różnych
typach.")

def main() -> None:
    zadanie10()

```

```
main()
```

### Zadanie 11

```
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310
```

```
# importy
import typing

# stałe i zmienne globalne

# funkcje

def zadanie11() -> None:
    grade_sheet = {
        "123456": {
            "name": "Jan",
            "family_name": "Kowalski",
            "grades": [4.5, 3.0, 5.0, 4.5, 3.5]
        },
        "234567": {
            "name": "Anna",
            "family_name": "Nowak",
            "grades": [4.0, 4.5, 3.5, 4.0, 5.0]
        },
        "345678": {
            "name": "Piotr",
            "family_name": "Wójcik",
            "grades": [3.0, 3.5, 4.0, 4.5, 3.5]
        }
    }

    for student_index, student in grade_sheet.items():
        name = student["name"]
        family_name = student["family_name"]
        grades = student["grades"]
        grade_avg = sum(grades) / len(grades)
        print("Studenci:\n")
        print(f"Numer indeksu: {student_index}")
        print(f"Imię: {name}")
        print(f"Nazwisko: {family_name}")
        print(f"Średnia ocen: {grade_avg:.2f}")
        print()

def main() -> None:
    zadanie11()

main()
```

### Wnioski

Zadanie byłoby zrozumiałe, z większością nie miałem problemu. Jedynie zadania 12 nie umiałem wykonać.