|  |  |  |  |
| --- | --- | --- | --- |
| Obraz zawierający symbol, design  Opis wygenerowany automatycznie | Politechnika Bydgoska im. J. J. Śniadeckich  **Wydział Telekomunikacji,**  **Informatyki i Elektrotechniki** | | Obraz zawierający symbol, logo, Grafika, Czcionka  Opis wygenerowany automatycznie |
| **Przedmiot** | Skryptowe języki programowania | | |
| **Prowadzący** | mgr inż. Martyna Tarczewska | | |
| **Temat** | *Struktury danych języka Python* | | |
| **Student** | Marcin Ogórkiewicz | | |
| **Nr ćw.** | 3 | **Data wykonania** | 25.10.2023 |
| **Ocena** |  | **Data oddania spr.** | 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 dwukrotne.  
 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 krotką, 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 programy.  
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 były zrozumiałe, z większością nie miałem problemu. Jedynie zadania 12 nie umiałem wykonać.