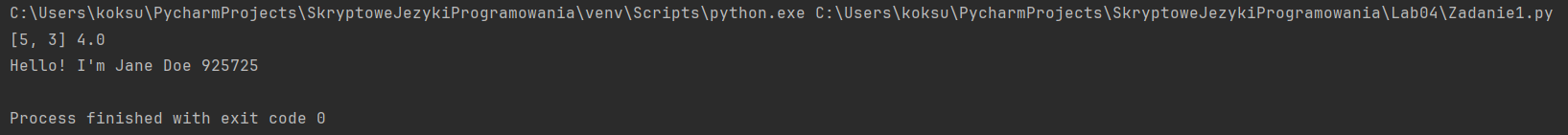
|  |  |  |  |
| --- | --- | --- | --- |
| Obraz zawierający symbol, design  Opis wygenerowany automatycznie | Politechnika Bydgoska im. J. J. Śniadeckich  **Wydział Telekomunikacji,**  **Informatyki i Elektrotechniki** | | Obraz zawierający symbol, logo, Grafika, Czcionka  Opis wygenerowany automatycznie |
| **Przedmiot** | Skryptowe języki programowania | | |
| **Prowadzący** | mgr inż. Martyna Tarczewska | | |
| **Temat** | *Obiektowość Pythona* | | |
| **Student** | Marcin Ogórkiewicz | | |
| **Nr ćw.** | 4 | **Data wykonania** | 29.10.2023 |
| **Ocena** |  | **Data oddania spr.** | 29.10.2023 |

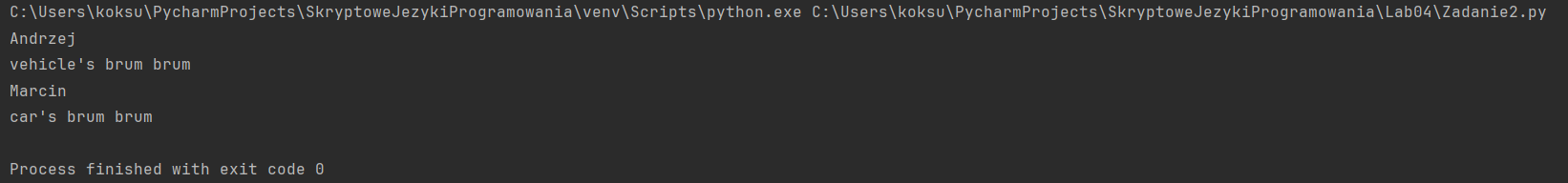
Zadanie 1

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
from random import randint  
from typing import List  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
class Student:  
 def \_\_init\_\_(self):  
 self.name = ""  
 self.last\_name = ""  
 self.index = ""  
 self.marks = []  
 self.avg = float  
  
 def give\_name(self, name: str, last\_name: str) -> None:  
 self.name = name  
 self.last\_name = last\_name  
  
 def give\_index(self) -> None:  
 self.index = str(randint(100000, 1000000))  
  
 def give\_mark(self, mark: int) -> None:  
 self.marks.append(mark)  
  
 def get\_marks(self) -> List[int]:  
 return self.marks  
  
 def get\_avg(self) -> float:  
 amount = len(self.marks)  
 sum\_of\_grades = 0  
 for i in range(amount):  
 sum\_of\_grades += self.marks[i]  
 self.avg = sum\_of\_grades/amount  
 return self.avg  
  
 def say\_hello(self) -> None:  
 print("Hello! I'm " + self.name + " " + self.last\_name + " " + self.index)  
  
  
def main() -> None:  
 s = Student()  
 s.give\_index()  
 s.give\_name("Jane", "Doe")  
 s.give\_mark(5) # wywołanie sposób 1  
 Student.give\_mark(s, 3) # wywołanie sposób 2  
 print(s.get\_marks(), s.get\_avg())  
 s.say\_hello()  
  
  
main()



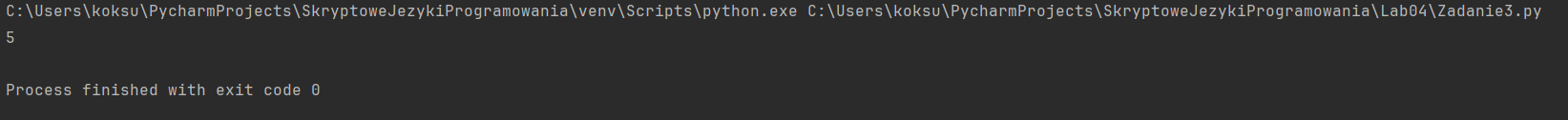
Zadanie 2

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
class Vehicle:  
  
 def \_\_init\_\_(self, owner: str, table: str):  
 self.owner = owner  
 self.table = table  
  
 def get\_owner(self) -> str:  
 return self.owner  
  
 def get\_sound(self):  
 print("vehicle's brum brum")  
  
  
class Car(Vehicle):  
  
 def get\_sound(self) -> None:  
 print("car's brum brum")  
  
  
def main() -> None:  
 vehi = Vehicle("Andrzej", "PY001KY")  
 print(vehi.get\_owner())  
 vehi.get\_sound()  
 cr = Car("Marcin", "CB001KY")  
 print(cr.get\_owner())  
 cr.get\_sound()  
  
  
main()  
  
"""Dla instancji klasy Car() wykonuje się metoda get\_sound zawarta w klasie Car(),  
podobnie dla Vehicle() wykonuje się metoda get\_sound zawarta w klasie Vehicle(). Metody get\_owner nie dało się  
wykonać dla klasy Vehicle(), ponieważ ta metoda należy do klasy Car().   
Można to naprawić przenosząc metodę \_\_init\_\_ do klasy nadrzędnej Vehicle(), wraz z metodą get\_owner."""



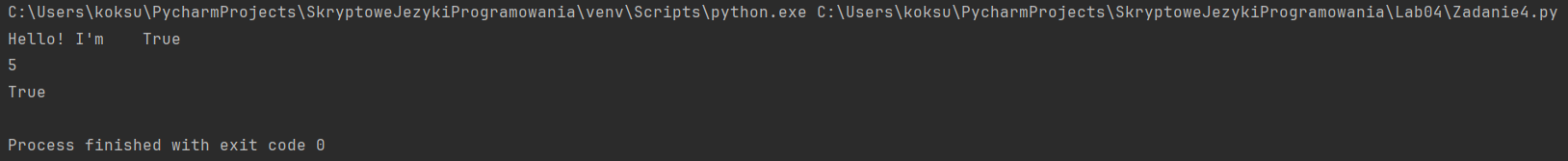
Zadanie 3

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
from random import randint  
from typing import List  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
class Student:  
 quantity = 0  
  
 def \_\_init\_\_(self):  
 self.name = ""  
 self.last\_name = ""  
 self.index = ""  
 self.marks = []  
 self.avg = float  
 Student.quantity += 1  
  
 def give\_name(self, name: str, last\_name: str) -> None:  
 self.name = name  
 self.last\_name = last\_name  
  
 def give\_index(self) -> None:  
 self.index = str(randint(100000, 1000000))  
  
 def give\_mark(self, mark: int) -> None:  
 self.marks.append(mark)  
  
 def get\_marks(self) -> List[int]:  
 return self.marks  
  
 def get\_avg(self) -> float:  
 amount = len(self.marks)  
 sum\_of\_grades = 0  
 for i in range(amount):  
 sum\_of\_grades += self.marks[i]  
 self.avg = sum\_of\_grades/amount  
 return self.avg  
  
 def say\_hello(self) -> None:  
 print("Hello! I'm " + self.name + " " + self.last\_name + " " + self.index)  
  
  
def main() -> None:  
 a = Student()  
 b = Student()  
 c = Student()  
 d = Student()  
 e = Student()  
 print(Student.quantity)  
  
  
main()  
"""Obiekty nie mają dostępu do zmiennej, ten ma jedynie klasa Student()"""



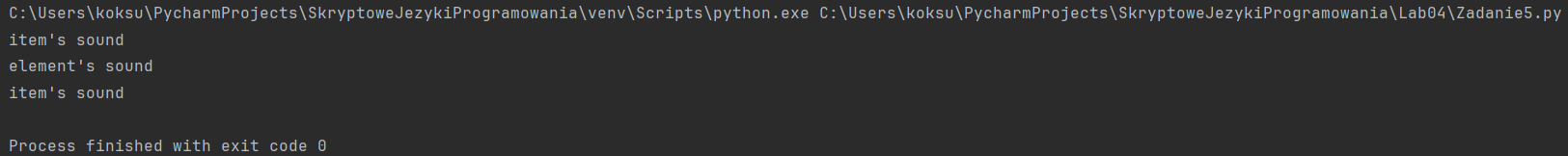
Zadanie 4

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
from random import randint  
from typing import List  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
class Student:  
 quantity = 0  
  
 def \_\_init\_\_(self):  
 self.name = ""  
 self.last\_name = ""  
 self.index = ""  
 self.marks = []  
 self.avg = float  
 Student.quantity += 1  
  
 def give\_name(self, name: str, last\_name: str) -> None:  
 self.name = name  
 self.last\_name = last\_name  
  
 def give\_index(self) -> None:  
 self.index = str(randint(100000, 1000000))  
  
 def empty\_name\_check(self) -> bool:  
 if self.name or self.last\_name == "":  
 return True  
 else:  
 return False  
  
 def give\_mark(self, mark: int) -> None:  
 self.marks.append(mark)  
  
 def get\_marks(self) -> List[int]:  
 return self.marks  
  
 def get\_avg(self) -> float:  
 amount = len(self.marks)  
 sum\_of\_grades = 0  
 for i in range(amount):  
 sum\_of\_grades += self.marks[i]  
 self.avg = sum\_of\_grades/amount  
 return self.avg  
  
 def say\_hello(self) -> None:  
 print("Hello! I'm " + self.name + " " + self.last\_name + " " + self.index + " " + str(self.empty\_name\_check()))  
  
  
def main() -> None:  
 a = Student()  
 b = Student()  
 c = Student()  
 d = Student()  
 e = Student()  
 a.say\_hello()  
 print(Student.quantity)  
 print(a.empty\_name\_check())  
  
  
main()



Zadanie 5

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
class Item:  
  
 def get\_sound(self) -> None:  
 print("item's sound")  
  
  
class Element:  
 def get\_sound(self) -> None:  
 print("element's sound")  
  
  
class Thing(Item, Element):  
 def say\_hello(self) -> None:  
 print("hello!")  
  
  
def main() -> None:  
 it = Item()  
 el = Element()  
 th = Thing()  
 it.get\_sound()  
 el.get\_sound()  
 th.get\_sound()  
  
  
main()  
"""Różnica polega na tym, że klasa Thing() będzie dziedziczyła atrybuty od klasy wymienionej jako pierwsza. """



Zadanie 6

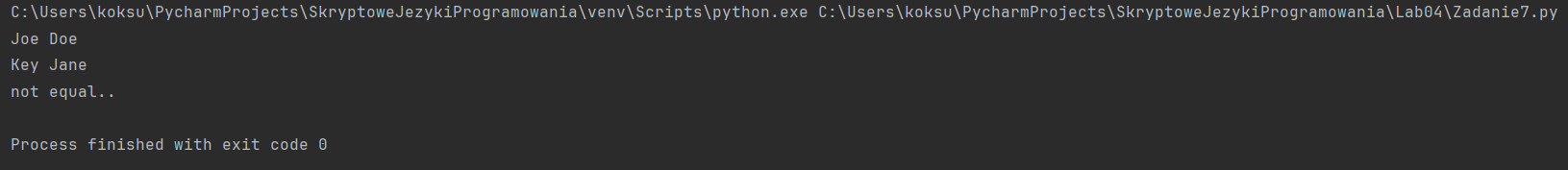
#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
class Vehicle:  
  
 def \_\_init\_\_(self, owner: str, table: str):  
 self.owner = owner  
 self.table = table  
  
 def \_\_repr\_\_(self) -> str:  
 return self.owner + " " + self.table  
  
 def \_\_str\_\_(self) -> str:  
 return self.owner + " " + self.table  
  
 def \_\_eq\_\_(self, o: object) -> bool: # funkcja do sprawdzania równości (==)  
 return self.owner == o.owner  
  
 def \_\_ne\_\_(self, o: object) -> bool: # funkcja zastępująca !=  
 return self.owner != o.owner  
  
 def \_\_lt\_\_(self, o: object) -> bool: # funkcja zastępująca <  
 return self.owner < o.owner  
  
 def \_\_gt\_\_(self, o: object) -> bool: # funkcja zastępująca >  
 return self.owner > o.owner  
  
 def \_\_le\_\_(self, o: object) -> bool:  
 return self.owner <= o.owner  
  
 def \_\_ge\_\_(self, o: object) -> bool:  
 return self.owner >= o.owner  
  
 def get\_owner(self) -> str:  
 return self.owner  
  
 def get\_sound(self):  
 print("vehicle's brum brum")  
  
  
class Car(Vehicle):  
  
 def get\_sound(self) -> None:  
 print("car's brum brum")  
  
  
def main() -> None:  
 vehi = Vehicle("Andrzej", "PY001KY")  
 cr = Car("Marcin", "CB001KY")  
 print(vehi.\_\_eq\_\_(cr))  
 print(vehi.\_\_ne\_\_(cr))  
 print(vehi.\_\_lt\_\_(cr))  
 print(vehi.\_\_gt\_\_(cr))  
 print(vehi.\_\_le\_\_(cr))  
 print(vehi.\_\_ge\_\_(cr))  
  
  
  
main()

Obraz zawierający zrzut ekranu, tekst

Opis wygenerowany automatycznie

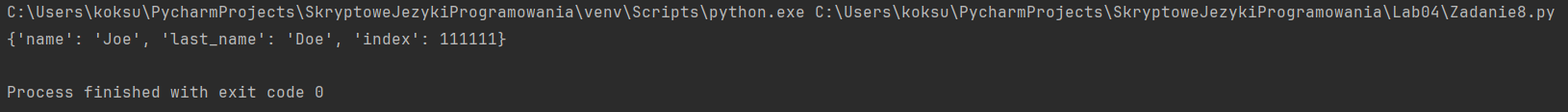
Zadanie 7

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
from random import randint  
from typing import List  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
class Student:  
 def \_\_init\_\_(self, name: str, last\_name: str, index: int):  
 self.name = name  
 self.last\_name = last\_name  
 self.index = index  
  
 def \_\_repr\_\_(self) -> str:  
 return self.name + " " + self.last\_name  
  
 def \_\_str\_\_(self) -> str:  
 return self.last\_name + " " + self.name  
  
 def \_\_eq\_\_(self, o: object) -> bool: #funkcja do sprawdzania równości (==)  
 return self.index == o.index  
  
 def \_\_ne\_\_(self, o: object) -> bool: #funkcja zastępująca !=  
 return self.index != o.index  
  
 def \_\_lt\_\_(self, o: object) -> bool: #funkcja zastępująca <  
 return self.index < o.index  
  
 def \_\_gt\_\_(self, o: object) -> bool: #funkcja zastępująca >  
 return self.index > o.index  
  
  
s1 = Student('Joe', 'Doe', 111111)  
print(repr(s1))  
s2 = Student('Jane', 'Key', 222222)  
print(str(s2))  
if s1 == s2:  
 print("objects equal!")  
else:  
 print("not equal..")  
  
"""Bez przeciążenia funkcja nie działa poprawnie."""



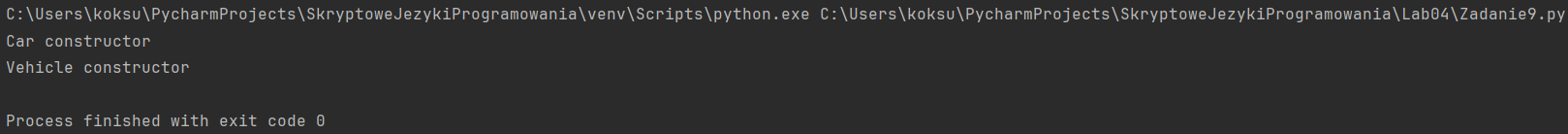
Zadanie 8

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
from random import randint  
from typing import List  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
class Student:  
 def \_\_init\_\_(self, name: str, last\_name: str, index: int):  
 self.name = name  
 self.last\_name = last\_name  
 self.index = index  
  
 def \_\_repr\_\_(self) -> str:  
 return self.name + " " + self.last\_name  
  
 def \_\_str\_\_(self) -> str:  
 return self.last\_name + " " + self.name  
  
 def \_\_eq\_\_(self, o: object) -> bool: #funkcja do sprawdzania równości (==)  
 return self.index == o.index  
  
 def \_\_ne\_\_(self, o: object) -> bool: #funkcja zastępująca !=  
 return self.index != o.index  
  
 def \_\_lt\_\_(self, o: object) -> bool: #funkcja zastępująca <  
 return self.index < o.index  
  
 def \_\_gt\_\_(self, o: object) -> bool: #funkcja zastępująca >  
 return self.index > o.index  
  
  
s1 = Student('Joe', 'Doe', 111111)  
s2 = Student('Jane', 'Key', 222222)  
print(s1.\_\_dict\_\_)  
  
"""Funkcja \_\_dict\_\_ przekształca obiekt klasa na słownik."""



Zadanie 9

#!C:\Users\koksu\AppData\Local\Programs\Python\Python310  
  
# importy  
import typing  
  
# stałe i zmienne globalne  
  
# funkcje  
  
  
class Vehicle:  
  
 def \_\_init\_\_(self, owner: str, table: str):  
 print("Vehicle constructor")  
 self.owner = owner  
 self.table = table  
  
 def get\_owner(self) -> str:  
 return self.owner  
  
 def get\_sound(self):  
 print("vehicle's brum brum")  
  
  
class Car(Vehicle):  
  
 def \_\_init\_\_(self, owner: str, table: str):  
 print("Car constructor")  
 self.table = table  
 super().\_\_init\_\_(owner, table)  
  
 def get\_sound(self) -> None:  
 print("car's brum brum")  
  
  
def main() -> None:  
 cr = Car("Marcin", "CB001KY")  
  
  
main()  
"""W klasie dziedziczącej, najpierw zostaje wywołany konstruktor własny, a potem konstrukor z klasy wyższej."""



Wnioski

Zadania były dla mnie zrozumiałe; pozwoliły mi zaznajomić się z działaniami na klasach w języku Python.