**Задание 1**

Создайте класс **Phone** с параметрами:

name

model

price

Обязательно создайте конструкторы.

Далее, создайте в этом классе функцию **getData()** которая возвращает все параметры (name, model, price) данного класса.

class Phone:  
  
 def \_\_int\_\_(self, name, model, price):  
 self.name = name  
 self.model = model  
 self.price = price  
  
 def getData(self):  
 return self.name, self.model, self.price

**Задание 2**

Создайте 5 объектов из класса **Phone** которую мы объявили выше. Задайте им разные значения (name, model, price). Затем, выведите на консоль данные о каждом телефоне при помощи метода **getData()**.

class Phone:  
  
 def \_\_init\_\_(self, name, model, price):  
 self.name = name  
 self.model = model  
 self.price = price  
  
 def getData(self):  
 return self.name, self.model, self.price  
  
  
p1 = Phone("Samsung", "a5", 1000)  
p2 = Phone("Apple", "12PRO", 2000)  
p3 = Phone("Xiaomi", "NOT", 3000)  
p4 = Phone("Samsung", "S12", 4000)  
p5 = Phone("Apple", "PRO", 5000)  
  
print(p1.getData())  
print(p2.getData())  
print(p3.getData())  
print(p4.getData())  
print(p5.getData())

**Задание 3**

Создайте список. Заполните этот список объектами, которые вы создали в предыдущем задании. Используя обычный цикл, выведите данные каждого объекта в списке при помощи функции **getData()**.

class Phone:  
  
 def \_\_init\_\_(self, name, model, price):  
 self.name = name  
 self.model = model  
 self.price = price  
  
 def getData(self):  
 return self.name, self.model, self.price  
  
  
p1 = Phone("Samsung", "a5", 1000)  
p2 = Phone("Apple", "12PRO", 2000)  
p3 = Phone("Xiaomi", "NOT", 3000)  
p4 = Phone("Samsung", "S12", 4000)  
p5 = Phone("Apple", "PRO", 5000)  
  
Phone\_list = [p1, p2, p3, p4, p5]  
for i in Phone\_list:  
 print(i.getData())

**Задание 4**

Создайте класс **Student** с параметрами:

id

name

surname

gpa

Обязательно создайте конструктор.

Создайте функцию**getStudentData()**;

Данная функция возвращает все данные студента.

Вы должны создать 5 объектов разных студентов с разными параметрами.

Создайте список, заполните список 5 объектами класса **Student**, которые мы создали до этого, и используя цикл, выведите данные каждого студента.

class Student:  
  
 def \_\_init\_\_(self, id, name, surname, gpa):  
 self.id = id  
 self.name = name  
 self.surname = surname  
 self.gpa = gpa  
  
 def getStudentData(self):  
 return self.id, self.name, self.surname, self.gpa  
  
  
s1 = Student("123", "Abai", "Rublev", 20)  
s2 = Student("456", "Aset", "Rublev", 21)  
s3 = Student("789", "Erden", "Rublev", 22)  
s4 = Student("234", "Baglan", "Rublev", 20)  
s5 = Student("546", "Sanzhar", "Rublev", 24)  
  
Student\_list = [s1, s2, s3, s4, s5]  
for i in Student\_list:  
 print(i.getStudentData())

### Задание 5

Используйте предыдущий класс **Student**.

Создайте специальную функцию:

**topStudent(students)**

Данная функция принимает в аргументы список из студентов. Функция должна возвращать из списка самого лучшего студента, у которого высокий **gpa**.

Создайте 10 объектов разных студентов с разными параметрами. Примените этот метод.

class Student:  
  
 def \_\_init\_\_(self, id, name, surname, gpa):  
 self.id = id  
 self.name = name  
 self.surname = surname  
 self.gpa = gpa  
  
 def getStudentData(self):  
 return self.id, self.name, self.surname, self.gpa  
  
def gettopStudent(Student\_list):  
 max = Student\_list[0].gpa  
 topStudent = Student\_list[0]  
 for i in Student\_list:  
 if i.gpa>max:  
 max=i.gpa  
 topStudent = i  
 return topStudent  
  
s1=Student("123", "Abai", "Rublev", 50)  
s2=Student("456", "Aset", "Rublev", 60)  
s3=Student("789", "Erden", "Rublev", 53)  
s4=Student("234", "Baglan", "Rublev", 49)  
s5=Student("546", "Sanzhar", "Rublev", 57)  
s6=Student("111", "Aleksadr", "Rublev", 56)  
s7=Student("222", "Evgeni", "Rublev", 53)  
s8=Student("333", "Anastasia", "Rublev", 57)  
s9=Student("444", "Marina", "Rublev", 80)  
s10=Student("555", "Daria", "Rublev",90)  
  
Student\_list=[s1,s2,s3,s4,s5,s6, s7, s8, s9, s10]  
  
topStudent = gettopStudent(Student\_list)  
print(topStudent.getStudentData())

### Задание 6

Создайте мини программу, с меню панелью отображенным ниже.

PRESS [1] TO ADD STUDENT

PRESS [2] TO LIST STUDENTS

PRESS [0] TO EXIT

**При нажатии [1]:**

**Insert name?**

        Вводите имя с консоли

**Insert surname?**

        Вводите фамилию с консоли

**Insert GPA?**

        Вводите GPA с консоли

После введения данных, программа должна добавить в список нашего студента.

**При нажатии [2]:**

Программа должна вывести список всех студентов.

1) Ilyas Zhuanyshev 4.0

2) Aruzhan Yelemes 3.9

3) Eldar Khassen 3.5

4) Anel Mukhamejanova 3.4

5) Aybek Bagit 3.2

**При нажатии [0]**:

Программа должна выйти.

class STUDENTS:  
 def \_\_init\_\_(self, name, surname, GPA):  
 self.name=name  
 self.surname=surname  
 self.GPA=GPA  
  
 def getStudentData(self):  
 return self.name, self.surname, self.GPA  
  
Student\_list = []  
while 1==1:  
 PRESS=int(input())  
 if PRESS==1:  
 print("Insert name?")  
 name=str(input())  
  
 print("Insert surname?")  
 surname=str(input())  
  
 print("Insert GPA?")  
 GPA = float(input())  
  
 s=STUDENTS(name, surname, GPA)  
 Student\_list.append(s)  
  
 elif PRESS==2:  
 for i in Student\_list:  
 print(i.getStudentData())  
  
 elif PRESS==0:  
 break  
  
 else:  
 break

**Задание 7**

Создайте класс **Employee** с параметрами:

name

age

salary

+ \_\_init\_\_(self,name, age, salary)

+ def **printData()** // Данная функция возвращает все данные работника.

Создайте 2 класса которые наследуют от класса **Employee:**

        1 – **Programmer**

        programmingLanguage

+ \_\_init\_\_(self,name, age, salary, programmingLanguage)

Переопределите функцию **printData()** , относительно их параметрам.

2 – **DataAnalitics**

databaseTool

+ \_\_init\_\_(self,name, age, salary, databaseTool)

Переопределите функцию **printData()** , относительно их параметрам.

В основном классе вы должны создать несколько объектов класса Programmer, DataAnalitics и добавить их в общий список для Employee.

Нужно вывести всю информацию и найти среднее значение зарплаты всех сотрудников.

class Employee:  
  
 def \_\_init\_\_(self, name, age, salary):  
 self.name = name  
 self.age = age  
 self.salary = salary  
   
 def printData(self):  
 return self.name, self.age, self.salary  
  
class Programmer(Employee):  
 programmingLanguage=""  
 def \_\_init\_\_(self,name, age, salary, programmingLanguage):  
 self.name=name  
 self.age=age  
 self.salary=salary  
 self.programmingLanguage=programmingLanguage  
  
 def printData(self):  
 return self.name, self.age, self.salary, self.programmingLanguage  
  
class DataAnalitics(Employee):  
 databaseTool = ""  
 def \_\_init\_\_(self, name, age, salary, databaseTool):  
 self.name = name  
 self.age = age  
 self.salary = salary  
 self.databaseTool = databaseTool  
  
 def printData(self):  
 return self.name, self.age, self.salary, self.databaseTool  
  
p=Programmer("Ilyas", 32, 200000,"Python")  
p1=Programmer("Zharas", 25, 190000,"Python")  
p2=Programmer("Almas", 28, 150000,"Python")  
d=DataAnalitics("Anel", 30, 250000, "Exel")  
d1=DataAnalitics("Serik", 35, 300000, "Exel")  
d2=DataAnalitics("Erbol", 30, 240000, "Exel")  
  
Emp=[p,p1,p2,d,d1,d2]  
sum=0  
  
for i in Emp:  
 sum+=i.salary  
AVG=sum/len(Emp)  
print("Средняя ЗП="+str(AVG))  
  
for i in Emp:  
 print(i.printData())

### Задание 8

Создайте класс **User** с параметрами:

id

login

password

name

surname

+ \_\_init\_\_(self,id, login, password, name, surname)

+ def **getData()** // Данная функция возвращает все данные пользователя

Создайте 2 класса которые наследуют от класса **User:**

        1 – **Staff**

        (float) salary

   (list) **subjects[]**+ \_\_init\_\_(self,id, login, password, name, surname, salary)

Переопределите функцию **getData()**, относительно их параметрам

Создайте функцию addSubject(self,subject), которая будет добавлять один урок в список subjects

2 – **Student**

(float) gpa

(list) **courses[]**

+ \_\_init\_\_(self,id, login, password, name, surname, gpa)

Переопределите функцию **getData()**, относительно их параметрам

Создайте функцию addCourse(self,course), которая будет добавлять один урок в список courses

Вы должны создать как минимум по 5 объектов класса **Student**, **Staff** и **Users**, и добавить их в общий список.  
Чтобы добавить курсы пользователям, просто вызовите функции **addSubject() & addCourse()**для каждого объекта!

**Подсказка: addSubject() & addCourse()**находятся внутри классов Staff & Student соответственно. Эти функции должны быть написаны вами. Они в аргументы принимают (string) subject/course и добавляют его в список с помощью list.append()

class User:  
 def \_\_init\_\_(self, id, login, password, name, surname):  
 self.id=id  
 self.login=login  
 self.password=password  
 self.name=name  
 self.surname=surname  
  
 def getData(self):  
 return self.id, self.login, self.password, self.name, self.surname  
  
class Staff(User):  
 def \_\_init\_\_(self, id, login, password, name, surname, salary):  
 self.id = id  
 self.login = login  
 self.password = password  
 self.name = name  
 self.surname = surname  
 self.salary=float(salary)  
 self.subjects=[]  
  
 def addSubjects(self,subject):  
 self.subjects.append(subject)  
  
 def getData(self):  
 return self.id, self.login, self.password, self.name, self.surname, self.salary, self.subjects  
  
class Student(User):  
 def \_\_init\_\_(self, id, login, password, name, surname, gpa):  
 self.id = id  
 self.login = login  
 self.password = password  
 self.name = name  
 self.surname = surname  
 self.gpa=float(gpa)  
 self.courses=[]  
  
 def addCourse(self,course):  
 self.courses.append(course)  
  
 def getData(self):  
 return self.id, self.login, self.password, self.name, self.surname, self.gpa, self.courses  
  
s1=Staff(123, "Asel123", "Asel123","Asel", "Barbolova", 200000)  
s2=Staff(456, "Zharas456", "Zharas456","Zharas", "Zharas", 250000)  
s3=Staff(789, "Almas789", "Almas789","Almas", "Almas", 190000)  
s4=Staff(234, "Marat234", "Marat234","Marat", "Marat", 150000)  
s5=Staff(567, "Anel567", "Anel567","Anel", "Anel", 290000)  
s=[s1,s2,s3,s4,s5]  
  
st1=Student(123, "Asel123", "Asel123","Asel", "Barbolova", 5)  
st2=Student(456, "Zharas456", "Zharas456","Zharas", "Zharas", 3)  
st3=Student(789, "Almas789", "Almas789","Almas", "Almas", 4)  
st4=Student(234, "Marat234", "Marat234","Marat", "Marat", 2)  
st5=Student(567, "Anel567", "Anel567","Anel", "Anel", 5)  
st=[st1,st2,st3,st4,st5]  
  
u1=User(123, "Asel123", "Asel123","Asel", "Barbolova")  
u2=User(456, "Zharas456", "Zharas456","Zharas", "Zharas")  
u3=User(789, "Almas789", "Almas789","Almas", "Almas")  
u4=User(234, "Marat234", "Marat234","Marat", "Marat")  
u5=User(567, "Anel567", "Anel567","Anel", "Anel")  
u=[u1,u2,u3,u4,u5]  
  
for i in s:  
 print("Добавьте список предметов для "+str(i.id) +" "+ i.name +" "+ i.surname+". Для окончания ввода введите '0'")  
 while 1:  
 subject\_ = str(input())  
 if subject\_=="0":  
 break  
 i.addSubjects(subject\_)  
 print(i.getData())  
 print()  
  
for i in st:  
 print("Добавьте курс для "+str(i.id) +" "+i.name +" "+i.surname+". Для окончания ввода введите '0")  
 while 1:  
 courses\_=str(input())  
 if courses\_=="0":  
 break  
 i.addCourse(courses\_)  
 print(i.getData())  
 print()  
  
for i in u:  
 print(i.getData())  
 print()

### Задание 9

Создайте класс **Car** с параметрами:

name;

model;

maxSpeed;

year;

volume;

Обязательно создайте конструктор.

Далее, создайте в этом классе функцию **ride()**, которая выводит на экран все параметры данного класса и имитирует езду в виде текста (Car is riding).

Создайте класс **Toyota**, которая наследует от класса **Car**.

У класса **Toyota** будет дополнительный параметр:

manufacturer; // Japan, Europe, USA, United Arabian Emirates

Переопределите функцию **ride()** для класса **Toyota**.

Создайте класс **Mercedes**, которая наследует от класса **Car**.

У класса **Mercedes** будет дополнительный параметр:

 classType; // E Class, S Class, M Class …

Переопределите функцию **ride()** для класса **Mercedes**.

Создать по 3 объекта класса **Car**, **Toyota** и **Mercedes**. Добавьте всех их в один общий список, и вызовите общий метод **ride()** для всех объектов в этом списке.

class Car:  
 def \_\_init\_\_(self, name, model, maxSpeed, year, volume):  
 self.name = name  
 self.model = model  
 self.maxSpeed = int(maxSpeed)  
 self.year = int(year)  
 self.volume = float(volume)  
  
 def ride(self):  
 return self.name, self.model, self.maxSpeed, self.year, self.volume, "Car is riding"  
  
  
class Honda(Car):  
 def \_\_init\_\_(self, name, model, maxSpeed, year, volume, manufacturer):  
 super().\_\_init\_\_(name, model, maxSpeed, year, volume)  
 self.manufacturer = manufacturer  
  
 def ride(self):  
 return super().ride(), self.manufacturer  
  
  
class Toyota(Car):  
 def \_\_init\_\_(self, name, model, maxSpeed, year, volume, manufacturer):  
 self.name = name  
 self.model = model  
 self.maxSpeed = int(maxSpeed)  
 self.year = int(year)  
 self.volume = float(volume)  
 self.manufacturer = manufacturer  
  
 def ride(self):  
 return self.name, self.model, self.maxSpeed, self.year, self.volume, self.manufacturer  
  
  
class Mercedes(Car):  
 def \_\_init\_\_(self, name, model, maxSpeed, year, volume, classType):  
 self.name = name  
 self.model = model  
 self.maxSpeed = int(maxSpeed)  
 self.year = int(year)  
 self.volume = float(volume)  
 self.classType = classType  
  
 def ride(self):  
 return self.name, self.model, self.maxSpeed, self.year, self.volume, self.classType  
  
  
c1 = Car("Toyota", "Легковая", 340, 2021, 2.4)  
c2 = Car("Mercedes", "Легковая", 460, 2021, 4.0)  
c3 = Car("Hyundai", "Легковая", 360, 2022, 2.0)  
  
t1 = Toyota("Toyota", "Легковая", 340, 2021, 2.0, "Japan")  
t2 = Toyota("Toyota", "Легковая", 340, 2021, 2.4, "Europe")  
t3 = Toyota("Toyota", "Легковая", 340, 2021, 2.2, "United Arabian Emirates")  
  
m1 = Mercedes("Mercedes", "Легковая", 400, 2022, 3.5, "E Class")  
m2 = Mercedes("Mercedes", "Легковая", 400, 2022, 4.0, "S Class")  
m3 = Mercedes("Mercedes", "Легковая", 400, 2022, 5.0, "M Class")  
  
h1 = Honda("Honda", "Легковая", 250, 2021, 3.5, "Japan")  
  
s = [c1, c2, c3, t1, t2, t3, m1, m2, m3, h1]  
  
for i in s:  
 print(i.ride())