Practical

Classes

1. Create the class **Circle** with attributes **radius** and **color**. Inside your class, create the method **getDesc(self)** which should print the text “A **color** circle with radius **radius**.”, using the values of the corresponding attributes**.**
Create class object(s) and test your class.

2. Create a class which has 1 attribute **my_str** of type String and 2 methods: **get_String(self)** and **print_String(self)**. The method **get_String(self)** returns the value of the attribute **my_str.** The method **print_String(self)** prints the value of the attribute **my_str,** making all the letters uppercase**.**

3. Create the class **Employee,** which has the following attributes: **name**, **last_name** and a private attribute **monthly_salary**. Inside your class, create the method **getFullName(self),** which will rerturn “**name last_name**”, using the values of the corresponding attributes. Create the method **annualSalary(self)**, which will calculate the annual salary of the employee, using the values of the corresponding attributes and will return “High” in case the salary is >100 and “Low”, otherwise.
Create class object(s) and test your class.

4. Create the class **Car** with the following attributes: **model**, **color** and **max_speed**. Inside the class, create the method **compareCar(self, car2)** which gets an object of type **Car** as an arument and returns the text “car1 is better than car2” if the **maxSpeed** attribute of your car is larger than the **maxSpeed** attribute of car2 and returns the text “car2 is better than car1”` otherwise.
Create class object(s) and test your class.

5. Create the class **Police_car** which has the following attributes: **owner**, **price** and a private attribute **pass_code.** Create a class attribute **tax_value** with a value 0.2. Create the method **tax(self),** which returns the tax you are supposed to pay for the car using the following formula: **tax_value * price**. Create the method **greeting(self)** which prints the text “Welcome to your car, **owner**”, using the value of the attribute **owner,** only if the value of the attribute **pass_code** is “admin”.

(OPTIONAL) Add set and get methods for the private attribute **pass_code**.

Create class object(s) and test your class.

Assertions

1. Create a function **Alarm(day),** which gets 1 argument day (the format is as follows: “Monday”, “Tuesday”, etc.). Inside the function, write an assert statement, which checks whether the value of the attribute **day** is not “Sunday”, in case the condition is not satisfied, it should give an error message “I won’t wake you up today!”.

2. Create the function **sum(x, y),** which gets 2 attributes **x** and **y** and returns their sum. Inside the function write an assert statement which checks whether the type of the arguments **x** and **y** is int, in case the condition is not satisfied, it gives an error message “Arguments of type int required”.

Exceptions

1. Create the function **div(x, y),** which gets 2 attributes **x** and **y** and returns x/y. Inside the function write a try … except block, which checks if **y** is not 0, in case the condition is not satisfied, it throws a general exception Exception.

2. Repeat the previous exercise but fiure out what should be a specific exception in this case and replace the general exception with the specific one.

3. Create a list with the following values: **['a', 0, 2].** Write a program which will go over the list using a loop and print the reciprocal of each value from the list (1/x). If there are cases when you cannot calculate 1/x for the value, you should cover those by a corresponding exception.

The output of the program should be of the following format:

The entry is: **the current entry of the list**
The reciprocal of **the current entry of the list** is **the value of the reciprocal**

OR

The entry is: **the current entry of the list**

Oops! **The exception that occured**

4. Write a program which gets an input from the user (using the input() function) and stores the value in the variable **username**. If the value of the variable **username** is “Rambo”, raise an exception which will print the text “Rambo is an invalid username”, otherwise, print the following text “Welcome, **username**”, using the value of the variable **username.**

Homework

1. Հասկացեք, թե որ դեպքում է առաջանում **ModuleNotFoundError** exception ու գրեք այդպիսի exception առաջացնող օրինակ:

2. Ստեղծեք **div(x, y)** ֆունկցիան, որն ընդունում է 2 attribute **x** ու **y** ու վերադարձնում է x/y: Ֆունկցիայի ներսում գրեք assert statement, որը ստուգում է թե արդյոք **y-ը** 0 չէ ու, պայմանի չբավարարման դեպքում, տալիս է error message “Can’t divide”.

3. Ստեղծեք **Person** class-ը.
   Attributes: **name**, **last_name**, **age**, **gender, student** (սա boolean attribute է՝ այսինքն ընդունում է True/False արժեքներ), ինչպես նաև private attribute **password**
   Methods:
   Greeting(self, second_person) - ստանում է Person տիպի object որպես input, տպում է “Welcome dear X.” ` որտեղ X-ը **second_person-ի** name-ն է::
   Goodbye(self) - տպում է “Bye everyone!”
   Favourite_num(self, num1) - ստանում է integer տեսակի **num1**-ը որպես input և վերադարձնում է “My favourite number is **num1**” `օգտագործելով **num1** attribute-ի արժեքը.
   Read_file(self, filename) - ստանում է String տիպի **filename** փոփոխականը ու փորձում է կարդալ այդ անունով ֆայլը՝ **filename**-ի վերջում ավելացնելով “.txt” (“**filename**.txt”). Կարդալու համար օգտագործեք open() ֆունկցիան:

   Ձեր ստեղծած class-ին ավելացրեք exception-ներ (առնվազն 1 ընդհանուր ու 1 կոնկրետ exception, որտեղ համարում եք, որ կա դրա կարիքը):
   Ավելացրեք համապատասխան set ու get method-ներ password private attribute-ի համար:

   Optional: Ավելացրեք decorator, որը կստուգի թե ինչքան ժամանակ է խլում Greeting method-ն աշխատացնելը: