# Exercise: Classes and Instances

Problems for exercise and homework for the [Python OOP Course @SoftUni](https://softuni.bg/courses/python-oop). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1937>

## Point

Create a class called **Point**. Upon initialization it should receive **x** and **y** (numbers). Create **3 instance methods**:

* **set\_x(new\_x)** - changes the x value of the point
* **set\_y(new\_y)** - changes the y value of the point
* **distance(x, y)** - returns the distance between the point and the provided coordinates

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| p = Point(2, 4)  p.set\_x(3)  p.set\_y(5)  print(p.distance(10, 2)) | 7.615773105863909 |

## Circle

Create a class called **Circle**. Upon initialization it should receive a **radius** (number). Create a class attribute called **pi** which should equal **3.14**. Create **3 instance methods**:

* **set\_radius(new\_radius)** - changes the radius
* **get\_area()** - returns the area of the circle
* **get\_circumference()** - returns the circumference of the circle

The area should be rounded to the 2nd decimal.

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| circle = Circle(10)  circle.set\_radius(12)  print(circle.get\_area())  print(circle.get\_circumference()) | 452.16  75.36 |

## Account

Create a class called **Account**. Upon initialization it should receive **id** (number), **name** (string), **balance** (number; **optional**; **0** by default). The class should also have **3 instance methods**:

* **credit(amount)** - add the amount to the balance and return the new balance
* **debit(amount)** - if the amount is **less** than the balance, **reduce** the balance by the amount and **return** the new balance. Otherwise return **"Amount exceeded balance"**
* **info()** - returns **"User {name} with account {id} has {balance} balance"**

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| account = Account(1234, "George", 1000)  print(account.credit(500))  print(account.debit(1500))  print(account.info()) | 1500  0  User George with account 1234 has 0 balance |
| account = Account(5411256, "Peter")  print(account.debit(500))  print(account.credit(1000))  print(account.debit(500))  print(account.info()) | Amount exceeded balance  1000  500  User Peter with account 5411256 has 500 balance |

## Employee

Create class **Employee**. Upon initialization it should receive **id** (number), **first\_name** (string), **last\_name** (string), **salary** (number). Create **3 more instance methods**:

* **get\_full\_name()** - returns **"{first\_name} {last\_name}"**
* **get\_annual\_salary()** - returns the salary for **12 months**
* **raise\_salary(amount)** - **increase the salary** by the given amount and **return the new salary**

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| employee = Employee(744423129, "John", "Smith", 1000)  print(employee.get\_full\_name())  print(employee.raise\_salary(500))  print(employee.get\_annual\_salary()) | John Smith  1500  18000 |

## Time

Create a class called **Time**. Upon initialization it should receive **hours**, **minutes** and **seconds** (numbers). The class should also have **class attributes** **max\_hours** equal to **24**, **max\_minutes** equal to **60** and **max\_seconds** equal to **60**. You should also create **3 instance methods**:

* **set\_time(hours, minutes, seconds)** - update the time
* **get\_time()** - returns **"{hh}:{mm}:{ss}"**
* **next\_second()** - update the time with one second (use the **class attributes** for validation) and return the new time (using the **get\_time()** method)

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| time = Time(9, 30, 60)  print(time.next\_second()) | 09:31:00 |
| time = Time(10, 59, 59)  print(time.next\_second()) | 11:00:00 |
| time = Time(24, 59, 59)  print(time.next\_second()) | 01:00:00 |