Object Oriented Programming

# Before Class

1. Familiarise yourself with principles and techniques on Object Oriented Programming in Python – Lessons 54 to 59 in "Introduction to Computer Science and Programming Using Python (MIT)".

<https://youtube.com/playlist?list=PLRJdqdXieSHN0U9AdnmwD-9QcR9hmw04d>

1. Familiarise yourself with the resources available on w3schools on inheritance and creating derived classes:

<https://www.w3schools.com/python/python_inheritance.asp>

1. Familiarise yourself with inheritance rules in the Python Tutorial, Section 9.5:

<https://docs.python.org/3/tutorial/index.html>

# During Class

## String representation of object

1. For the convenience and readability of the program code, it is possible to create a text representation of an object in the form of a string. Such an object can then be used wherever string data is required, e.g. when calling print().

Run the program below. Note the \_\_str\_\_ method and the call of the print() function.

class University():  
   
 def \_\_init\_\_(self, name):  
 self.name = name   
   
 def \_\_str\_\_(self):  
 return self.name + "is the best!"  
   
my\_university = University('UEK Kraków')  
print(my\_university)

1. Create a class that represents pieces of music. Define a class constructor that allows you to set the initial values of the music piece (artist, track title, album, year) when the object is created. Complete the class with the \_\_str\_\_ method returning the song data as a string, in the format as below (4 lines).

Performer: Ed Sheeran  
Song: Hearts Don't Break Around Here  
Album: Divide  
Year: 2017

Then create three objects that represent three different pieces of music. Display these objects.

## Static methods

…

## Operator overloading

…

# After Class

1. Create a class that describes cell phones with at least 3 phone states and 2 behaviors. Define the text representation of an object. Then create 2 objects representing 2 phones. Display their features and call their bahaviors.
2. Books are published both in a traditional (paper) and electronic (e-books) form. Create a class that describes a book, no matter what its type. Next, create derived classes for the description of the paper and the e-book. A paper book should contain the number of its pages, while an electronic book should contain the name of the file in which it is saved. Create one traditional book and one electronic book. Display data describing these books.
3. A variable created outside of the \_\_init\_\_ method is a class variable. Unlike an instance variable, a class variable holds a value that is common (and therefore identical) to all objects created from that class.

Run the program below. Note how the class variable is declared and how it is modified. Specify a class variable in the program, the place where it is modified and the place where its value was used.

# class definition  
class Film():  
   
 # class variables  
 cinema = "Multikino"  
   
 def \_\_init\_\_(self, title):  
 self.title = title  
   
 def \_\_str\_\_(self):  
 return f"{self.title} ({Film.cinema})"  
   
# program  
film1 = Film("The Shawshank Redemption")  
print(film1)  
film2 = Film("Pulp Fiction")  
print(film2)  
  
# renaming the cinema (changing the value  
# of a class variable)   
Film.cinema = "Cinema City "  
print(film1)  
print(film2)

1. The student has a name, surname, ID (album number) and a field of study. All students study at the same university (UEK Kraków). Create a class describing a student. Student ID should be assigned automatically as a sequential natural number starting from 100000. For this purpose, create a class variable to store the last student’s ID number. When creating a new student (object), increase the value of this variable by one and then use it as the identifier of the created student. Then write a program that creates 3 different students and displays their personal data in the format as below. Use the \_\_str\_\_ method.

Anna MAJ (100001), Applied Informatics, UEK Kraków

1. A static method is the kind of method that is called in the context of a class, not objects that are created from that class. The following program defines a static method.

class Arrays():  
  
 @staticmethod  
 def print\_in\_col(array):  
 for c in array:  
 print(c)  
   
my\_array = [4,1,8,7,2]  
Arrays.print\_in\_col(my\_array)

Then complete the class with another static method that displays the contents of an array in a row, separating the values with a comma. Make sure that the comma is not displayed after the last value. Call a new method in the program.

1. In the Arrays class, add three static methods that:
   1. create an array with a given number of elements with the same values. Use list.append():  
        
      method\_name(number\_of\_array\_elements, value\_of\_array\_elements)
   2. create an array with a given number of elements and the random value of these elements in the range of <m, n>:  
         
      method\_name(number\_of\_array\_elements, value\_from, value\_to)
   3. determines the number of array elements whose values are in the given range <m, n>:  
        
      method\_name(array, value\_from, value\_to)

Then, write a program that creates a 10-element array with element values equal to 4 and a 20-element array of random integers in the range of <-7,8>. Display the contents of arrays and calculate how many values between <-1,1> are contained in a 20-element array.

1. Create a class with three static methods for calculating the surface area of figures: triangle, rectangle, circle. Then use these methods to calculate the area of the following figures:
   1. Circle with a radius of 3
   2. Rectangle with sided 4 and 7
   3. Triangle with base 6 and height 2
2. Operator overloading is defining the meaning of existing operators for your own data types. Often implemented in the form of the use of special methods. For example, in Python, the new functionality for the comparison operator == can be implemented by defining the \_\_eq\_\_ function in the class.

On the Internet, find examples of \_\_eq\_\_ definitions. Then complete the Point class below, describing a point on the plane with coordinates (x, y), by adding the \_\_eq\_\_ method to compare two points.

class Point():  
 def \_\_init\_\_(self,x,y):  
 self.x = x  
 self.y = y  
 def \_\_str\_\_(self):  
 return f'P({self.x},{self.y})'

Using the Point class, create a program that will calculate the distance on the plane between two defined points. Using the conditional statement check if these points are identical - use the comparison operator ==, i.e. p1 == p2. If the points are identical, display a message that the distance between them is 0. Otherwise, calculate and display the distance between the two points.