Classes  
And Objects

# Before Class

1. Familiarise yourself with object-oriented programming concept. Then watch the video:

<https://youtu.be/SS-9y0H3Si8?feature=shared>

1. Watch the video on how Python classes are defined:

<https://youtu.be/apACNr7DC_s?feature=shared>

1. Familiarise yourself with tutorials available on w3schools which deals with class creation and their components (fields and methods).

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

1. From the course textbook, read the Chapter 14 (Object-oriented programming).
2. Read how to make a string representation of an object:

<https://www.pythontutorial.net/python-oop/python-__str__/>

# During Class

## Class and object creation

1. Learn on how to use classes and objects in Python:

<https://youtu.be/f0TrMH9s-VE?feature=shared>

1. Identify at least 3 states and 3 behaviors for a book object. Then, for the listed states and behaviors, create a class with fields (attributes) and methods. Try to use verbs in method names as they describe activities. Finally, create an object, call its methods and display object’s properties.

class Book():  
 def \_\_init\_\_(self,title,author,pages):  
 self.title = title  
 self.author = author  
 self.pages = pages  
 self.current\_page = 1  
 self.is\_open = False  
 def open(self):  
 self.is\_open = True  
 def close(self):  
 self.is\_open = False  
 def change\_page(self,page):  
 self.current\_page = page  
  
book = Book(  
 "Harry Potter and the Philosopher's Stone",  
 "J. K. Rowling",223)  
  
print(f"My favourite book is {book.title}, ",end="")  
print(f"written by {book.author}. ",end="")  
print(f"This book has {book.pages} pages.")

book.open()  
book.change\_page(15)  
  
if book.is\_open:  
 print(f"Reading the book, page {book.current\_page}")  
else:  
 print("I am going to read the book later.")

1. Identify at least 3 states and 3 behaviors for a telephone object. Then, for the listed states and behaviors, create a class with fields (attributes) and methods. Try to use verbs in method names as they describe activities. Finally, create a object, call its methods and display object’s properties.

class Phone():  
 …  
 …  
 …  
 …  
 …

## 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. Pay attention to 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('KUE')  
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). Then, create two objects that represent two different pieces of music. Display these objects. Sample result:

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

## Class with multiple components

1. Write a program in which create a TV class that describes a TV set. The class should contain one boolean attribute called 'is\_on' that specifies whether the TV set is turned on. Initially, the TV is turned off. Add turn\_on() and turn\_off() methods in the class to turn the TV on and off, respectively. Also, add a show\_status() method to display whether the TV is on or off. Sample message:

TV is on

Then, try using the TV set in the program:

* 1. Create TV set
  2. Show TV status
  3. Turn TV on
  4. Show TV status
  5. Turn TV off
  6. Show TV status

1. In the TV class, add the channel\_no attribute indicating the number of the TV channel displayed by the TV set. Initially, the TV is set to channel 1. Modify the show\_status() method so that it also displays the TV channel number, but only if the TV is turned on:

TV is on, channel 1

Then, try using the TV set.

1. Add the set\_channel(new\_channel\_no) method in the TV class to set the TV channel number. Then try using the TV set.
   1. Create a TV set
   2. Show TV status
   3. Turn TV on
   4. Show TV status
   5. Change TV channel to 5
   6. Show TV status
   7. Turn TV off
   8. Show TV status
2. In the TV class, add the channels attribute containing a list of available TV channel names (array). Initially, the array should be empty (TV not programmed, no available channels). Add set\_channels(channels\_list) and show\_channels() methods in the TV class, which allows you to set channels on the TV and display the list of available channels. Sample list of available channels:

Channel list:  
1. TVP1  
2. TVP2  
3. Polsat  
4. TVN  
5. Filmbox  
6. Discovery

Then, try using the TV set:

* 1. Create a TV set
  2. Show TV status
  3. Turn TV on
  4. Show TV status
  5. Display the list of available channels
  6. Set TV channels: TVP1, TVP2, Polsat, TVN, Filmbox, Discovery
  7. Display the list of available channels
  8. Show TV status
  9. Turn TV offs

1. In the TV class, make changes to the show\_status() method so that it displays not only the selected channel number but also its name. When the selected channel number exceeds the list of available channels, the channel name is not displayed.

TV is on, channel 4 (TVN)

Then, try using the TV. Set at least 7 channels (seven TV stations), change channel numbers and display TV status every time.

# After Class

1. Create a class that describes cell phones with at least 3 phone states and 2 behaviors. Define a text representation of an object. Then, create 2 objects. Display their features and call their bahaviors.
2. In the TV class, add support for volume adjustment in the range 0 to 10. The initial value of the volume level is 0. Add two methods to increase and decrease the TV volume level by one. Note that you cannot increase or decrease the volume beyond the specified range. Display the current volume level in the show\_status() method. Then check the operation of the TV by adjusting and displaying its volume level.
3. E-book is a digital book that can be read using a computer or other electronic devices (electronic book readers). Write a program in which define a class that describes states and behaviors of an e-book. Each book has a title, author, number of pages and the current page number that is currently being read. It is possible to open a book - then we can read it. If a book is open, it is possible to go to the next or previous page.

Place the class describing e-books in a separate file/module. In the main program file, try using the e-book:

* 1. Create a book with a title, author, number of pages (check how to set the initial values of the fields at the time of creating the object using the \_\_init\_\_ method / constructor and passing the initial values as arguments to the method call)
  2. Open a book
  3. Display a book status (title, author, page numbers, current page no)
  4. Read a few pages
  5. Display a book status
  6. Close a book
  7. Read a few pages (it should not be possible to perform this operation - display the message that the book is closed).

1. The medical thermometer measures the patient's temperature in the range from 34.0 to 42.0 degrees Celsius, with an accuracy of 0.1 degrees. Write a program in which define a class that describes the states and behaviors of the thermometer. The thermometer should enable temperature measurement (do it by generating a random number from the 34.0 to 42.0 range) and display the measured value. If the temperature is at least 37 degrees Celsius, the thermometer should additionally display the 'Fever' message, e.g.

Temperature: 37.2C (fever)

When the temperature is at least 41.0, the thermometer should additionally display the message 'CRITICAL TEMPERATURE!!'. Place the class definition and the main program in separate files. Then use the program and:

* 1. Create a thermometer
  2. Turn thermometer on
  3. Measure temperature
  4. Display temperature
  5. Turn thermometer off

1. The bank account has a 26-digit number assigned when creating an account. The initial account balance is PLN 0. You can deposit any amount on the account. You can also withdraw any amount from the account, provided that it does not exceed the account balance. If you try to withdraw a larger amount, the following message will be displayed: "Insufficient funds on the account". At any time, it is possible to display information about the number and balance of the bank account in the following format:

Bank Account No: 11 1111 1111 1111 1111 1111 1111  
Balance: PLN 25,38

Create a program that handles the bank account.

* 1. Familiarises yourself with a problem.
  2. Identify an object
  3. Define the states and behaviors of the object.
  4. Transform the states and behaviors of the object into the fields and methods of the class that will serve as a pattern for creating an object.
  5. Create a sketch of the class without creating any method content.
  6. Create the content of each method.

Then, use the program and:

* 1. Create a bank account with the number 12 3456 5555 9090 1111 0000 7722
  2. Display account balance
  3. Deposit PLN 25,30
  4. Display account balance
  5. Withdraw PLN 31,70
  6. Display account balance
  7. Withdraw PLN 14
  8. Display account balance

1. Write a program containing a Statistics class that describes the properties of any set of numbers. The class should allow to:
   1. Add to the set of numbers, the next number read from the keyboard (store the numbers in the array)
   2. Display all numbers separated by a space
   3. Determine the greatest number
   4. Determine the smallest number
   5. Calculate the arithmetic mean of numbers
   6. Calculate of the median
   7. Display of calculated / determined statistical quantities (minimum, maximum, arithmetic mean, median)

Then, use the program for numbers:

12, 37, 6, 9, 17

1. The Contact class contains the 'name', 'email' and 'telephone' fields enabling the description of a single contact on a smartphone. The Contact\_List class allows you to store contacts (store objects describing contacts in an array) and perform the following operations:
   1. Adds a new contact
   2. Displays the contact list

Write a program consisting of 3 files (smartphone.py, contact.py, contact\_list.py). In the mail program (smartphone.py) create an object representing a contact list and add the following people data:

John Brown brown@onet.pl 555234000  
Anna May am@o2.pl 232000199  
George Small smallg@google.pl 222999100  
Paola Big bigpaola@poczta.pl 100200300

Then, display the contact list available on the smartphone.

1. An object representing an employee contains the following data: name, surname, age, and seniority (the number of years worked). Define a C class that allows you to create an object. Provide employee data at the time of creating the object, in the given order. Define a text representation of an object in the class that contains a string of last name, first letter of first name, and seniority. If the employee is an adult (at least 18 years old), use uppercase letters, otherwise lowercase letters. Sample result:

C("Anna","May",17,7) returns "maya7"  
C("George","Brown",21,4) returns "BROWNG4"

1. An object contains a list of coordinates of points on the plane, as a two-dimensional array. Define a C class that allows you to create an object. Provide the list of coordinates of points at the time of creating the object. In the class C, define a method m(n) that returns true when at least n points are in the first quadrant of the coordinate system (both point coordinates are greater than 0), or false otherwise. Sample result:

C([[2,3],[1,8],[-6,4],[3,-7]])  
m(2) returns True  
m(3) returns False

1. A stadium is divided into sectors, each marked with a letter. There is a certain number of fans in each sector. Define the class C, which allows you to create an object representing the stadium with a list of sectors and the number of fans in sectors. Data, as a dictionary, should be transferred to the object at the time of its creation. Define in the class a method m1(s,n) that allows you to change the number of fans n in sector s or add a new sector s with the given number of fans n. Define in the class a method m2(s) that returns the sum of fans in the sectors listed in the string s. Sample result:

C({"A":120,"D":150,"G":90,"K":110})  
m1("G",130)  
m2("GD") returns 280  
m2("KEJ") returns 110