Workbook 2: Object Orientated Programming Inheritance, Polymorphism and Overriding.

Please complete the tasks in this workbook.

For any code created a screen shot should be provided in this document and the code file saved as a .py file.

Each section or task will need its own code file.

When complete save it as a PDF file with the following naming convention and then submit:  
  
**oop2\_<student number>\_<student name>.pdf**

**For example:**

**oop2\_123456\_fred\_blogs.pdf**

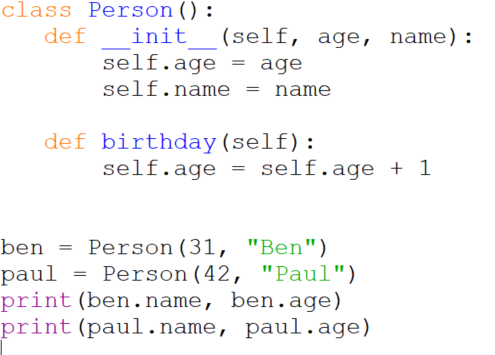
# Task 1

## Extending our person Class from the previous worksheet.

A college has many students.

Each student will have their own record.

Some new software has been installed but it is not quite complete and is missing a way of displaying a student record. You have been asked to rectify this and add some more functionality.

The Following code has been provided:  
  


This shows a class, called *Person*, it’s constructor (this will be explained in the next lesson) used to setup the object and a single method, *birthday(),* to adjust the birthday value.

Some example code is also provided on how to use the class.

You will need to copy the above code into the IDE of your choice and make sure it works before attempting the tasks below.

Task 1.A:

Produce a design on paper to enable the Person class to be a super class along with classes that create student and staff objects.

#### 1.A Answer:

import random

class Person:

    def \_\_init\_\_(self, pname, ptype, page):

        self.pname = pname

        self.ptype = ptype

        self.page = page

        self.student\_id = self.generate\_student\_id() #calling here, adding student\_id to each student

    def generate\_student\_id(self):

         """Generates a random 8-digit student ID."""

         return str(random.randint(10000000, 99999999))

    def describe\_person(self):  # Fixed typo 'describe\_pet' to 'describe\_person'

        return str(self.pname + " is a " + self.ptype + " and his ID is " + str(self.page) + " years old"  +" ID: "+self.student\_id)

def start():

    person1 = Person("Zdislav Wladislav", "Student", 17)

    person2 = Person("Henry", "Staff", 20)

    print(person1.describe\_person())

    print(person2.describe\_person())

start()

### Task 1.B

Convert this design into software and show the output below along with a copy of the software.

#### 1.B Answer:

import random

class Person:

    def \_\_init\_\_(self, pname, ptype, page):

        self.pname = pname

        self.ptype = ptype

        self.page = page

        self.student\_id = self.generate\_student\_id()

    def generate\_student\_id(self):

        """Generates a random 8-digit student ID."""

        return str(random.randint(10000000, 99999999))

    def describe\_person(self):

        return str(self.pname + " is a " + self.ptype + " and his ID is " + str(self.page) + " years old"  +" ID: "+self.student\_id)

def start():

    person1 = Person("Zdislav Wladislav", "Student", 17)

    person2 = Person("Henry", "Staff", 20)

    print(person1.describe\_person())

    print(person2.describe\_person())

start()

### 

Task 1.C

On the previous worksheet you were asked to create a method to save data to a file.  
  
Can you now create a class that will handle this process along with being able to read CSV data from a file and use it to create records for students and staff.

#### 1.C Answer:

# Task 2:

## Pizza Code Express

Read the attached Pizza Code assignment.

Put simply, a national chain of Pizza Shops and they want you to create the software.

To ensure backwards compatibility, they require that your code meets the following:

* A way to handle user interactions
* A way to handle the order process
* A way to handle order calculations for the order (sub total, final price etc.)
* A way to generate a receipt and store it as a file

Produce two separate versions of it:

* One in the procedural paradigm
* One in the OOP paradigm:
  + A class to handle the user interactions.
  + A class to handle the order process
  + A class to handle calculations for the order.
  + A class to handle file operations for the receipt.

The class handling calculations should be used as part of the order class.

Along with your solution answer the following questions:

1. Explanation of the key features of object-oriented programming.
2. Explanation of the importance of encapsulation, inheritance and polymorphism in object-oriented programming
3. What is a Paradigm in respect to Programming? (Provide some examples.)
4. Which have you found better (OOP or Procedural) to write your solution in?
5. Why?

### Task 2 Answer: