Workbook 1: Object Orientated Programming Introduction

Please complete the tasks in this workbook.

For any code created a screenshot 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.

**oop1\_<student number>\_<student name>.pdf**

**For example:**

**oop1\_123456\_fred\_blogs.pdf**  
  
**Place it, with your python code files, in a zipped folder and submit the zipped folder.**

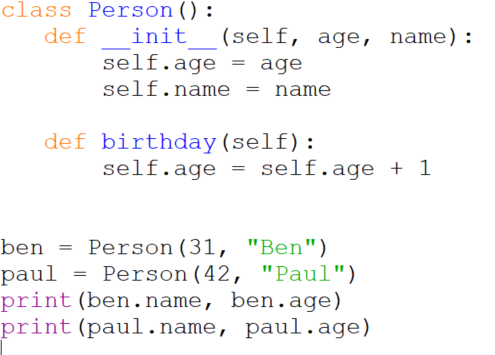
# Task 1

## A Basic Class

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*, its constructor used to set up 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:

Add a student ID number to the constructor (any random six digit number will be fine). This student number should be provided when creating the object.

Write a method to be able to view this number when requested.

Note: This method will be what’s known as a getter as it *gets* a value. Remember…an objects attributes should not be directly accessible.

#### 1.A Answer: class Person():

#### def \_\_init\_\_(self, age, name, idnum):

#### self.age = age

#### self.name = name

#### self.idnum = idnum

#### def birthday(self):

#### self.age = self.age

#### def idcard(self):

#### self.idnum = self.idnum

#### 

#### ben = Person(31, "Ben", 123456)

#### paul = Person(42, "Paul", 123456)

#### print (ben.name, ben.age , ben.idnum)

#### print (paul.name, paul.age, paul.idnum)

### Task 1.B

Create a course ID number and make it available within the class as an attribute.

Write a method that will display this course ID when requested.

#### 1.B Answer:

### class Person:

### def \_\_init\_\_(self, age, name, Idnum):

### self.age = age

### self.name = name

### self.Idnum = Idnum

### def birthday(self):

### self.age = self.age + 1

### def get\_idnum(self):

### # This method can be used to retrieve the idnum

### return self.Idnum

### def set\_idnum(self, new\_idnum):

### # This is the "setter" method to update the idnum

### self.Idnum = new\_idnum

### Task 1.C

Write a method that will allow the course ID to be updated.

while True:

print("do you want to update ID numbers?")

print("1. yes")

print("2. no")

choice = input("Enter your choice: ")

if not choice.isdigit():

print("Invalid input. Please enter a number.")

time.sleep(2)

continue

choice = int(choice)

if choice == 1:

print("Who's ID number do you want to update?")

print("1.", ben.name)

print("2.", paul.name)

person\_choice = input("Enter your choice (1 or 2): ")

if not person\_choice.isdigit():

print("Invalid input. Please enter a number.")

time.sleep(2)

continue

person\_choice = int(person\_choice)

if person\_choice == 1 or person\_choice == 2:

new\_idnum = input("Enter the new ID number: ")

if person\_choice == 1:

ben.set\_idnum(new\_idnum)

print(f"Updated {ben.name}'s ID number.")

print(ben.name, ben.age, ben.get\_idnum())

elif person\_choice == 2:

paul.set\_idnum(new\_idnum)

print(f"Updated {paul.name}'s ID number.")

print(paul.name, paul.age, paul.get\_idnum())

else:

print("Invalid input. Please select 1 or 2.")

time.sleep(2)

elif choice == 2:

print("No IDs updated")

break # Exit the loop if the user chooses not to update

else:

print("Invalid choice. Please choose 1 or 2.")

time.sleep(2)Note: This method is what is known as a *setter* as it provides an interface to set an attribute.

#### 1.C Answer:

### Task 1.D

Using your class write some code that will add 20 students and then display each student’s number, name and course.

(Think about iteration and data structures….how can you iterate over the objects…where/how can you store such a collections of objects to refer to….what would be the best way of identifying a given object?)

#### 1.D Answer:

# Store Person objects in a list

people = [

Person(31, "Ben", "123456"),

Person(42, "Paul", "987654"),

Person(22, "Sarah", "456789"),

Person(60, "John", "789012"),

Person(23, "johanson", "789014"),

Person(21, "zdislav", "789016"),

Person(21, "wladislav", "789024"),

Person(22, "jakub", "789015"),

Person(23, "gregosz", "789011"),

Person(26, "Finn", "789112"),

Person(25, "Marissa", "789212"),

Person(23, "Jaxon", "789312"),

Person(22, "Matej", "789412"),

Person(19, "Ivana", "789512"),

Person(17, "Ivan", "789612"),

Person(17, "Zora", "789712"),

Person(27, "Luke", "789812"),

Person(22, "Boris", "789912"),

Person(19, "Petra", "789992"),

Person(17, "Pjotr", "789129")

]

if choice == 1:

print("Whose ID number do you want to update?")

for index, person in enumerate(people, 1):

print(f"{index}. {person.name}")

person\_choice = input("Enter your choice (number): ")

try:

person\_choice = int(person\_choice)

except ValueError:

print("Invalid input. Please enter a number.")

time.sleep(2)

continue

if 1 <= person\_choice <= len(people):

new\_idnum = input("Enter the new ID number: ")

people[person\_choice - 1].set\_idnum(new\_idnum)

print(f"Updated {people[person\_choice - 1].name}'s ID number.")

print(people[person\_choice-1])

### Bonus: Task 1.E

Can you adapt a version of your 1.D code to be able to save the data to a file?

#### 1.E Answer:

# Save the game state to a file

def save(people\_list):

with open("people", "w") as f:

for person in people\_list:

f.write(f"{person.name},{person.age},{person.Idnum}\n")

print("Saved successfully.")

input("Press Enter to continue...")

def load():

global people

people = [] # Create an empty list to store loaded data

if os.path.exists("people"):

with open("people", "r") as f:

for line in f:

name, age, Idnum = line.strip().split(",")

people.append(Person(int(age), name, Idnum))

print("Loaded successfully.")

else:

print("No saved game found.")

input("Press Enter to continue...")

# Task 2:

## Cats and Dogs

As part of another software project, you have been asked to start development of some classes for a simulated cat and dog.

Your task is to provide some class diagrams showing the design of each class as well as example code showing your design working.

Each animal should be showing behaviors specific to that animal, eg. A cat should not bark, and a Dog should not purr.

You are free to implement this how you wish as long as it is using Python.

### Task 2 Answer:

# Task 2a:

Cats and Dogs and other animals

Now create other animal classes showing relavent behavior.