ICS4U Unit 1 Activity 11 - Unit Test Part 1

Phone Pyae Thet Khine
Teacher NEDA
ICS4U
14th March 2022

Knowledge/Understanding

Q1) Define the following terms with examples:

1. Modular Programming:
   Modular Programmins is the process of breaking up or separating codes into small components and store the code in a seperate file instead of the main program file. These separate files are called modules and they can be imported into the main program file by just using the `import` keyword. For example, in Python, we use `import foo` on the very top of the main script to import the whole module or `from foo import bar` to selectively import a single function or class. Modular Programming concept helps programmers to further structure their projects to a clearer and more organized way and makes code reuse-able.
2. One dimensional array:
   One dimensional array is an array or list of structured elements which are indexed according to thier position on the list. These elements in the one-dimensional array can be of a single data type in the most older programming languages such as Java or C++ but in Python, there can be a mix of more than one data type in a single array. Elements in these attays can be accessed by their index numbers which starts from zero. We can also further manipulate the elements in the arrays by using array methods which vary from language to language.
   Python Example:

```
arr = [1, 2, 3, 4]

print(arr[0])
```

Output

```
1
```

3. Encapsulation:
   Encapsulation is the restriction of access to variables or data inside an object. This helps to hide the inner implementations of methods inside of a class or

prevernting clients or users to see things that they are not supposed to see. Encapsulation hides values or states by using keywords like `private` to encapsulate states inside an object or the keyword `public` to make that variable accessable to any code that asks for it. In Python, variables are by default private but can be made public by the keyword `global`. Encapsulation is usually used in Object Oriented Programming but can be also used outside of OOP, for example to hide the value of a variable inside a function.
Encapsulationin Python wraps up the variables and methods into a single entity. These protect members inside the instance of a class so they cannot be accessed by code outside a class.

```
class Student:
    def __init__(self, name, grade, age) -> None:
        self.name = name # Public
        self._grade = grade # Protected
        self.__age = age # Private

aaron = Student("Aaron", 12, 19)
print(aaron.name)
print(aaron._grade)
print(aaron.__age)
```

Output

```
Aaron
12

Traceback (most recent call last):
  File "/Users/williamkhine/Desktop/ICS4U-202203/Unit 3 - Designing Modular
Programs/U1A11.py", line 17, in <module>
    print(aaron.__age)
AttributeError: 'Student' object has no attribute '__age'
```

4. Polymorphism:
   Polymorphism in OOP is the ability to make or code functions or methods in an object that will perform different takes or excute a different set of codes when called with a different data-type or a different order of paramenters or arguments. There are two ways to polymorphrize methods, method overriding and method overloading.
   Overloading makes a new method with the same name but with different parameters such as the order or with more parameters added to handle more arguments and logics. This method will react to each configuration.

```
class Computer:
    def __init__(self) -> None:
        print("This is a Computer")

class AdvancedComputer(Computer):
    def __init__(self, processor, memory) -> None:
        print(f"This is a computer with {processor} processor and {memory}
memory")

asus = Computer()
dell = AdvancedComputer("Intel", "32GB")
```

Output

```
This is a Computer.
This is a computer with Intel processor and 32GB memory.
```

Thinking/Inquiry

Q2a) Investigate the program which convert string into integer.

```
def stringToInt(string):
    x = int(string)
    return(x)

string = input("Enter a number: ")
integer = stringToInt(string)
print(integer, type(integer))
```

Output

```
Enter a number: 46246
46246 <class 'int'>
```

Q2b) Compute a program to print array in reverse order.

```
# Method 1
arr = [1, 2, 3, 4, 5]
arr.reverse()
print(arr)

# Method 2
arr = [1, 2, 3, 4, 5]
```

```
print(arr[::-1])

# Method 3
arr = [1, 2, 3, 4, 5]
revArr = []
for i in range(len(arr) - 1, -1, -1):
    revArr.append(arr[i])

print(revArr)
```

Output

```
[5, 4, 3, 2, 1]
[5, 4, 3, 2, 1]
[5, 4, 3, 2, 1]
```

Communication

Q3) Explain with an example of method of overloading or method of overriding by using modular design concept that support reusable code.

```
# In the file person.py
class Person:
    def __init__(self, name) -> None:
        self.name = name

    def move(self):
        print("Person moving.")
```

```
# In the file main.py
# import person
class RunningPerson(Person):
    def __init__(self, name) -> None:
        super().__init__(name)

    def move(self):
        print(f"{self.name} running.")

john = Person("John")
jess = RunningPerson("Jess")

john.move()
jess.move()
```

Output

```
Person moving.
Jess running.
```

In the file `person.py`, the object `Person` is made with the method `move`. This makes the person walk. But in the `main.py`, the class is imported and the method move is overridden so that the person now runs instead of walking.
The class `RunningPerson` inherits all the methods defined by the parent class `Person`. The `move` method is redefined with the same paramenters with the different code. This is overriding with modular design with support for reusable code.

Application

Q4a) Demonstrate user defined class with an example.

User defined classes are used commonly in OOP to represent objects that can be protected and accessed by the user.

```
class Human:
    def __init__(self, name, gender) -> None:
        self.name = name
        self.gender = gender

    def showSpecs(self):
        print(f"My name is {self.name}. I am {self.gender}.")

albert = Human("Albert", "Male")
albert.showSpecs()
```

Output

```
My name is Albert. I am Male.
```

Q4b) Show the program to calculate the number of characters in first name of the person.

```
def nameCount(name):
    ln = len(name)
    print(ln)

name = input("What's your first name?: ")
nameCount(name)
```

Output

```
What's your first name?: William
7
```