**Assessment type (🗹):**

Arvand Eftekhar 20108797

Questioning (Oral/Written)

Practical Demonstration

3rd Party Report

Other – Project/Portfolio (*please specify)*

**Assessment Resources:**

|  |
| --- |
| You do not need specific resources to answer the questions in this assessment.  You may, however, use a Python shell and the online Python documentation if you think that helps. |

**Assessment Instructions:**

|  |
| --- |
| This assessment consists of multiple questions. Answer all questions to the best of your ability. Think about your spelling and grammar. If a question has a minimum or maximum word count, take that into account.  Answer all questions in your own words. If you use external resources, including ChatGPT, please provide references.  If provided, you should use the provided template to answer your questions.  Do NOT zip the Word document before uploading it. |

**Assessment Instrument:**

|  |
| --- |
| Question 1 Fundamentals of OOP (100-200 words)  1. **In your own words** describe what an object is and its relationship to a class. 2. An object contains behaviours, state, and boundaries. How do specify each in Python? 3. What are two key problem that OOP is trying to solve? |
| An object is like a thing in the real world, like a car, a dog, or a person. It has characteristics and things it can do. A class is like a blueprint or a template for creating objects. It defines what the object will be like and what it can do.  1. In Python, you specify behaviours of an object by defining functions inside the class. These functions are called methods. The state of an object refers to its current condition or data. You specify the state by defining variables inside the class, which are called attributes. 2. 1-Complexity Management: OOP helps manage complexity by breaking down a program into smaller, more manageable parts called objects. Each object has its own data and behaviours, which makes it easier to understand and maintain the code.   2-Code Reusability: OOP promotes code reusability by allowing objects to be reused in different parts of a program or in different programs altogether. Once a class is defined, it can be used to create multiple objects with similar characteristics and behaviours, saving time and effort in writing new code.  **Resource: https://www.wikipedia.org/** |
| Question 2 – Language constructs (20-50 words) In your own words, describe what a sequence is. Illustrate with an example. |
| A sequence is an ordered collection of elements. In programming, it's often used to store multiple values in a specific order. For example, a list in Python is a sequence: my\_list = [5, 6, 7, 8, 9]  Here, my list is a sequence of numbers in the order they appear. |
| Question 3 – Language constructs (20-50 words)  1. In your own words, describe what iteration is. 2. Provide at least two iteration constructs that are supported in Python and provide an example of one. |
| a. Iteration is the process of repeatedly executing a set of instructions. It allows you to perform actions multiple times, typically over a sequence of elements.Two iteration constructs in Python are for loops and while loops. Here's an example of a for loop: # Iterating over a list  my\_list = [5, 6, 7, 8, 9]  for num in my\_list:  print(num)  This loop iterates over each element in my list and prints it. |
| Question 4 – Language constructs (20-50 words) In your own words, describe what a selection is. Illustrate with an example. |
| Selection, also known as conditional execution, involves making decisions based on certain conditions. It allows a program to choose between different paths of execution based on whether a condition is true or false.Example: # Checking if a number is positive or negative  num = 10  if num > 0:  print("The number is positive.")  else:  print("The number is negative.")  In this example, the program selects different paths based on whether the variable num is greater than zero or not. If the condition num > 0 is true, it executes the code inside the if block, otherwise, it executes the code inside the else block. |
| Question 5 – Language syntax rules (30-70 words)  1. In your own words, describe the importance of indentation in Python. 2. Provide at least one example of code where a difference in indentation changes the meaning of the program. |
| Indentation is important in Python because it defines the structure and scope of the code. It is used to group statements inside loops, conditional statements, and functions. Proper indentation makes the code more readable and helps developers understand the logic of the program.  1. # Incorrect indentation   def my\_function():  print("This is inside the function")  # Correct indentation  def my\_function():  print("This is inside the function")  # Difference: In the first example, the print statement is not inside the function due to incorrect indentation. In the second example, the print statement is correctly indented inside the function. |
| Question 6 – Data types (20-50 words)  1. In your own words, describe how a user might input a number (in a terminal, from the keyboard) and how you can use that number within a calculation. 2. You may illustrate with a small code snippet. Be precise and make sure Word’s auto correct does not change the meaning of the program. |
| To input a number from the keyboard in a terminal, a user can use the input() function in Python. This function reads a line of text from the standard input (keyboard) and returns it as a string. To use the input number in a calculation, you can convert the input string to an integer or float using the int() or float() functions, respectively.  1. # Input a number from the user   num\_input = input("Enter a number: ")  # Convert the input string to an integer  num = int(num\_input)  # Use the number in a calculation  result = num \* 2  # Print the result  print("Twice the number is:", result) |
| Question 7 – Data types (20-50 words)  1. Describe the following Python data types and give an example of each: a) Integer b) Float c) String d) Boolean |
|  |
| Question 8 – OOP Principles (20-50 words)  1. Explain the concept of polymorphism and inheritance in object-oriented programming. Provide an example in Python to illustrate these concepts. |
| a. Integer: An integer is a whole number without any decimal point. For example, 6, -11, and are integers.b. Float: A float (floating-point number) is a number that has a decimal point. For example, 4.15, -3.2, and 0.0 are floats.c. String: A string is a sequence of characters enclosed in single or double quotes. For example, "HI", 'Python', and "234" are strings.d. Boolean: A boolean is a data type that has only two possible values: True or False. It is used to represent the truth values of logic expressions. For example, True and False are booleans. |
| Question 9 – Organisational documentation (20-50 words) In your own words, describe the importance of PEP-8 and why organisations often use this as their preferred coding style for Python. |
| PEP 8 is important as it provides guidelines for writing clean, readable Python code, making it easier for developers to understand and maintain. Organizations prefer PEP 8 because it promotes consistency across projects, improves code quality, and enhances collaboration among team members. |
| Question 10 – Constructors (30-70 words) Explain the purpose of constructors in Python classes. Provide an example of a constructor in a Python class. For the purposes of this question, the \_\_init\_\_ method can be considered a constructor. |
| Constructors in Python classes are special methods used for initializing newly created objects. They allow you to set initial values for object attributes. The \_\_init\_\_ method serves as the constructor in Python classes. Here's an example: class Person:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age  # Creating a new Person object  person1 = Person("Arvand", 23)  print(person1.name) # Output: Arvand  print(person1.age) # Output: 23 |
| Question 11 – Aggregation (30-70 words)  1. Describe the concept of object aggregation and provide a Python example to illustrate this. |
| Object aggregation in Python refers to a relationship where one class contains references to one or more instances of another class. It represents a "has-a" relationship. class Engine:  def \_\_init\_\_(self, horsepower):  self.horsepower = horsepower  class Car:  def \_\_init\_\_(self, make, model, engine):  self.make = make  self.model = model  self.engine = engine  engine = Engine(300)  car = Car("BMW", "M3", engine)  print(car.engine.horsepower) # Output: 300 |
| End of the questions |