|  |  |  |  |
| --- | --- | --- | --- |
| Instructor |  | Due Date |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Part** | **1** | **2** | **3** | **4** | **Total** |
| *Maximum Points* | **25** points | **25** points | **25** points | **25** points | **100**G101010 pointsG |
| ***Your Score*** |  |  |  |  |  |

**Textbook Reading Assignment**

Thoroughly read Chapter(s) on Classes and Objects in your **Python** textbook.

**Part 1 Glossary Terms**

Define, in detail, each of these glossary terms from the realm of computer programming logic and design and computer topics, in general. If applicable, use examples to support your definitions. Consult your notes or course textbook(s) as references or the Internet by visiting Web sites such as:

|  |  |  |
| --- | --- | --- |
| http://www.askjeeves.com | http://www.webopedia.com | http://www.bing.com |

**(a) class**

|  |
| --- |
| Code that defines data attributes and methods of a “class” of object that can have distinct instances of the class, as well as extensions of the superclass and its methods. |

**(b) constructor**

|  |
| --- |
| A constructor or factory function returns a class of object whenever it is called. |

**(c) instantiate**

|  |
| --- |
| To create a new instance of a class is to instantiate that class, in Python we do this by calling that class name with parentheses, e.g,. my\_new\_class = SomeClass() |

**(d) object - oriented programming**

|  |
| --- |
| A paradigm that guides how develops design programs, using certain principles, including encapsulation, polymorphism, inheritance, and abstraction among others. Typically this involves the creation and extension of classes of objects with public / private methods. |

**(e) self**

|  |
| --- |
| The self keyword in Python classes refers to the instance of the class. In the class, methods use “self” as a parameter, e.g., `def someMethod(self)`, and data attributes use the “self” keyword to set data properties, e.g., `self.something = “someString”` |

**Part 2 Textbook Exercises - OOP Classes and Objects**

For each of the following, select the correct answer.

**(1)** A(n) \_\_\_\_\_ is a set of real - world objects, parties and major events related to the problem.

(a) class **(b) problem domain**

(c) object (d) instance

**(2)** What does the acronym UML stand for?

**(a) Unified Modeling Language** (b) United Modeling Language

(c) Unified Model Language (d) United Model Language

**(3)** Which section in the UML holds the list of the class’s data attributes?

(a) First **(b) Second**

(c) Third (d) Fourth

**(4)** Which section in the UML holds the list of the class’s methods?

(a) First (b) Second

**(c) Third** (d) Fourth

**(5)** What type of method provides a safe way for code outside a class to retrieve the values of attributes, without exposing the attributes in a way that they could be changed by the code outside the method?

**(a) Accessor** (b) Mutator

(c) Setter (d) Class

**(6)** What attributes belong to a specific instance of the class?

(a) Instance (b) Self

(c) Object **(d) Data**

**(7)** What is the special name given to the method that returns a string containing the object’s state?

(a) \_\_state\_\_ (b) \_\_obj\_\_

**(c) \_\_str\_\_** (d) \_\_init\_\_

**(8)** Which method is automatically executed when an instance of the class is created in memory?

(a) \_\_state\_\_ (b) \_\_obj\_\_

(c) \_\_str\_\_ **(d) \_\_init\_\_**

**(9)** When a method is called, what does Python make to reference the specific object on which the method is supposed to operate?

(a) state variable **(b) self parameter**

(c) object data (d) init procedure

**(10)** What type of programming contains class definitions?

(a) Procedural (b) Object

**(c) Object - oriented** (d) Modular

**Part 3 Textbook Exercises - OOP Classes and Objects**

**Mix and Match**

Match each glossary term on the left with its appropriate description on the right.

**e (1) \_\_init\_\_** (a) a group of functions and data placed inside a container

**f** **(2) self** (b) can be passed as an argument to a function

**i (3)** class variable (c) a function defined inside a class

**g** **(4)** instantiate an object (d) any name following a dot

**a (5)** class (e) a method used to initialize an object

**b (6)** object (f) a variable name commonly used as the first parameter in a method definition

**d (7)** attribute (g) illustrated with this example:

**myTri = Triangle(6, 8, 10)**

**j (8)** instance variable (h) a variable that is available everywhere within the program

**c** **(9)** method (i) a variable shared by all instances of a class

**h** **(10)** global variable (j) a variable that is only available to particular instances of a class

**Part 4 Textbook Exercises - OOP Classes and Objects**

**(1) ( Class Construction )**

Consider the definition of class Employee that is given in the code segment below.

|  |
| --- |
| class Employee:  empCount = 0  def \_\_init\_\_(*self*, *name*, *salary*):  self.name = name  self.salary = salary  Employee.empCount += 1  def displayCount(*self*):  print("employee tally %d" % Employee.empCount)  def displayEmployee(*self*):  print("name : ", self.name, ", salary: ", self.salary)  emp1 = Employee("James", 1500)  emp2 = Employee("Katherine", 2250)  emp3 = Employee("Pauline", 2050)  emp4 = Employee("Esteban", 2550)  emp1.displayCount()  emp1.displayEmployee()  emp2.displayEmployee()  emp3.displayEmployee()  emp4.displayEmployee() |

Modify the above code by instantiating two more employee objects and then display a count of the employees, display all of the employees and their total salaries.