|  |  |  |  |
| --- | --- | --- | --- |
| 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, Objects and Inheritance 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) OOP Inheritance**

|  |
| --- |
| Inheritance in OOP classes involves the extension of a super class into sub classes, in which the super class methods and data attributes are accessible in the sub classes |

**(b) method overriding**

|  |
| --- |
| In line with polymorphism, extended subclasses can override superclass methods by defining methods with the same name as the superclasses’ |

**(c) subclass**

|  |
| --- |
| A class extended from a superclass, e.g., class Dog(Animal) is a subclass of Animal |

**(d) superclass**

|  |
| --- |
| A class from which a subclass is extended from, e.g., `class Animal` in the previous example |

**(e) UML Class Diagram**

|  |
| --- |
| Unified Modeling Language diagram is a method for outlining the name, attributes, and methods a class might have. |

**Part 2 Textbook Exercises - OOP Inheritance**

For each of the following, select the correct answer.

**(1)** New attributes and methods may be added to a subclass.

**(a) True** (b) False

**(2)** It is not possible to indicate inheritance in a UML diagram.

(a) True **(b) False**

**(3)** Each subclass has a method named \_\_init\_\_ that overrides the superclass’s \_\_init\_\_.

**(a) True** (b) False

**(4)** When a class inherits another class, it is required to use all the data attributes and methods of the superclass.

(a) True **(b) False**

**(5)** An " is a " relationship exists between a grasshopper and a bumblebee.

(a) True **(b) False**

**(6)** In a UML diagram depicting inheritance, you only write the name of the subclass.

(a) True **(b) False**

**(7)** A subclass may not override any method other than the \_\_init\_\_ method.

(a) True **(b) False**

**(8)** Polymorphism works on any two class methods that have the same name.

**(a) True** (b) False

**(9)** One of the ingredients of polymorphic behavior is the ability to define a method in a subclass, and then define a method with the same name in a superclass.

**(a) True** (b) False

**(10)** A superclass inherits attributes and methods from its subclasses without any of them having to be rewritten.

(a) True **(b) False**

**Part 3 Textbook Exercises - OOP Inheritance**

**Mix and Match**

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

**i (1) isinstance** (a) allows subclasses to have methods with the same names as methods in their superclasses

**f (2)** relationship (b) another name for a subclass

**h (3)** subclass (c) is used to create an " is a " relationship among classes

**b (4)** derived (d) a version of the superclass whereby new attributes and methods may be added to a subclass

**j (5)** form (e) another name for a superclass

**c (6)** inheritance (f) in an inheritance \_\_\_\_\_ , a television is a(n) subclass of the electronics class

**d (7)** specialized (g) action that occurs when a subclass method has the same name as a superclass method

**a (8)** polymorphism (h) another name for an extended class in an inheritance relationship

**e (9)** base (i) this function determines whether an object is an instance of a specific class or an instance of a subclass of that class

**g (10)** overrides (j) polymorphism refers to an object’s ability to take different \_\_\_\_\_

**Part 4 Textbook Exercises - OOP Inheritance**

**(1) ( Class Construction )**

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

|  |
| --- |
| class Person:  def \_\_init\_\_(*self*, *first*, *last*):  self.firstname = first  self.lastname = last  def Name(*self*):  *return* self.firstname + " " + self.lastname  class Employee(Person):  def \_\_init\_\_(*self*, *first*, *last*, *staffnum*):  Person.\_\_init\_\_(self, first, last)  self.staffnumber = staffnum  def GetEmployee(*self*):  *return* self.Name() + ", " + str(self.staffnumber)  x = Person("Sammy", "Student")  y = Employee("Penny", "Peters", "805")  class PartTime(Employee):  def \_\_init\_\_(*self*, *first*, *last*, *staffnum*, *hourlyRate*):  Employee.\_\_init\_\_(self, first, last, staffnum)  self.hourlyRate = hourlyRate  def GetEmployee(*self*):  *return* f"{Employee.GetEmployee(self)}, is a part time employee who makes: {str(self.hourlyRate)}"  partTimeEmployee = PartTime("Hayes", "Crowley", 123, 18.00)  partTimeEmployee.GetEmployee() |

Modify the above code by deriving a new class from class Employee named as class PartTime and derive some objects based on this new class.