Practice Theory

Problem 1

Question 1

A blueprint created by a programmer for an object. [MCQ]
(a) Class
(b) Attribute
(c) Object
(d) OOP

Answer

(a)

Solution

Class is the blueprint which is used to create objects. These objects have same interfaces but different attributes and behaviour.

Question 2

Which of the following is an instance of a class. [MCQ]
(a) Method
(b) Attribute
(c) Object
(d) OOP

Answer

(c)

Solution

Refer to the solution of question 1.

Question 3

Select all the correct options. A `private` variable and method of a class can be accessed. [MSQ]

(a) Outside the class

(b) Inside a class method of the same class

(c) Inside the class

(d) In child/subclass

Answer

(b), (c)

Solution

`private` variables and methods of a class can be accessed by the members of the class only. No subclass or any other class can access these variables.

Question 4

A class variable `x` is defined inside the class `A`. We wish to update the class variable value of this class. Which of the following holds true. [MSQ]

(a) can be updated in the class method/functions of the same class as `self.x=10`

(b) can be updated in the method/functions of child class `Y` as `Y.x=10`

(c) can be updated outside the class using `A.x=10`

(d) can be updated using an instance `obj` of the class `A` as `obj.x=10`

Answer

(b), (c)

Solution

Option (b) and (c) are valid ways. The variable is a class attribute. Therefore it can be accessed by class or subclass only, not using their objects.

Question 5

Select the most appropriate option which holds true. A variable defined inside the class method is accessible: [MSQ]

(a) in other method of the same class

(b) in the subclass method

(c) not accessible in other methods of the same class

(d) not accessible outside the class

Answer

(c), (d)

Solution

A variable defined inside the class method is called instance variable. This is accessible only inside the method only and specific to the object. It can not be accessed outside class or by any other object.

Question 6

Match the `Name` with the correct `Description`. [MCQ]

Name	Description
1. class	A. a blueprint from which instance objects are created
2. object	B. a class function/method with first parameter as `self` (representing an object of the class) in its definition, It only operate on specific object data not on the class data
3. class variable	C. an instance of a class, it has own copy of member variables(instance variable) and methods(instance methods) to operate on
4. instance variable	D. a function/method declared inside class, it can operate on class data
5. class method	E. a variable or attribute declared inside class body and outside method, all object shares class variables
6. instance method	F. a variable declared inside the instance function/method, belongs to and can be accessed only from the object
7. class member	G. variables declared inside a function
8. local variables	H. variables and methods declared inside class

(a) 1-E, 2-C, 3-A, 4-D, 5-F, 6-B, 7-H, 8-G

(b) 1-A, 2-C, 3-E, 4-F, 5-D, 6-B, 7-H, 8-G

(c) 1-C, 2-A, 3-E, 4-D, 5-B, 6-F, 7-H, 8-G

(d) 1-F, 2-C, 3-E, 4-A, 5-G, 6-B, 7-H, 8-D

Answer

(b)

Solution

Self explanatory.

Question 7

Match the `Name` with the correct `Description`. [MCQ]

Name	Description
1. method overloading	B. Two methods are said to be overloaded when they have the same name but different signatures (number, type and order of parameters), a method overloading can happen in the same class or in the subclass. Python does not support method overloading rather it keeps only the latest defined method leaving previous definitions.
2. method overriding	A. Two methods are said to be overridden when they have the same name and signature (number, type and order of parameters), a method overriding can only happen between parent class and child class.
3. Hierarchical Inheritance	E. Multiple child classes `A, B, C` inherits from same parent class `X`
4. Multiple Inheritance	D. A child class `A` inherits from multiple parent class `X, Y, Z`
5. Multilevel Inheritance	C. A child class `P` inherits from its parent `Q`, which again inherits from its parent `R` (forming child-grandfather like relationship)

(a) 1-D, 2-A, 3-D, 4-E, 5-C

(b) 1-A, 2-E, 3-B, 4-C, 5-D

(c) 1-A, 2-B, 3-C, 4-D, 5-E

(d) 1-B, 2-A, 3-E, 4-D, 5-C

Answer

(d)

Solution

Self explanatory.

Problem 2

```python
class Words:
    count = 0
    def __init__(self, seqNo, word, partOfSpeech):
        Words.count += 1
        self.seqNo = seqNo
        self.word = word
        self.partOfSpeech = partOfSpeech
```

Question 1

What does `w0 = Words(0, 'the', 'Article')` do?

(a) Creates a variable `w0` having string value `'the'`
(b) Creates a `Words` object `w0` having `word` attribute of `'the'`
(c) Creates a variable `w0` having string value `'Article'`
(d) Creates a `Word` object with no attributes values.

Answer

(b)

Solution

`w0 = Words(0, 'the', 'Article')` creates an object of class `Words` with the value of attributes `seqNo`, `word` and `partOfSpeech` be `0`, `'the'` and `'Article'` respectively.

Question 2

Does the below code create a new `Words` object?

```
w77 = Words()
w77.seqNo = 77
w77.word = 'book'
w77.partOfSpeech = 'Noun'
```

(a) True
(b) False

Answer

(b) False

Solution

`Words` object requires 3 parameters.

Question 3

The `Words` class requires a method `getLetterCount` to get the letter count of the respective object. Choose the correct way of implementation.

(a)

```
def getLetterCount():
    return len(self.word)
```

(b)

```
def getLetterCount():
    return len(word)
```

(c)

```
def getLetterCount(self):
    return len(self.word)
```

(d)

```
def getLetterCount(self):
    return len(word)
```

Answer

(c)

Solution

Option	Comment
(a)	The parameter `self` is required for method of object
(b)	The parameter `self` is required for method of object
(c)	Correct Implementation. `word` is accessed through `self.word`
(d)	Here, `word` is not accessed through `self.word`

Question 4

The below code-snippet is executed after the class definition. What will be the printed value?

```python
w0 = Words(0, 'the', 'Article')
w1 = Words(1, 'a', 'Article')
w2 = Words(2, 'an', 'Article')
print(Words.count)
```

(a) 0
(b) 1
(c) 2
(d) 3

Answer

(d) 3

Solution

`count` is a class attribute initialized to zero and incremented by 1 on every object creation of the class `Words`. Here, three `Words` objects `w0`, `w1` and `w2` are created. Hence, `count` will be `3`.

Problem 3

Questions 1- 12 are based on a common theme. Use below class definitions to answer these questions. Instance and objects represent the same thing and can be used interchangeably.

```python
class Vehicle:
    units = 0
    def __init__(self, name, wheels):
        Vehicle.units += 1
        self.name =  name
        self.wheels = wheels
    def spec(self):
        print(self.name)
        print(self.wheels)
    def start():
        print("starting")
    def stop():
        print("stopping")

class Motor(Vehicle):
    speed = 0
    def __init__(self, name, wheels, fuel_type, capacity):
        super().__init__(name, wheels)
        self.fuel_type = fuel_type
        self.capacity = capacity
    def spec(self):
        super().spec()
        print(self.fuel_type)
        print(self.capacity)
    def raise_speed(self, num):
        self.speed = speed + num
    def slow_down_speed(self, num):
        self.speed = speed - num
    def apply_break(self):
        self.speed = 0
    def check_speed(self):
        print(self.speed)

class Car(Motor):
    def __init__(self, name="car", wheels=4, fuel_type='diesel', capacity=10, car_type="sedan"):
        super().__init__(name, wheels, fuel_type, capacity)
        self.__car_type = car_type
    def spec(self):
        super().spec()
        print(self.__car_type)
    def raise_speed(self, num, times):
        self.speed = self.speed + num * times
    def slow_down_speed(self, num, times):
        self.speed = self.speed - num * times
    def apply_break(self):
        self.speed = self.speed//10
    def get_speed(self):
        super().check_speed()
```

Question 1

Which of the following are class attributes of class `Vehicle`. [MCQ]

(a) `wheels`

(b) `name`

(c) `units`

(d) None of these

Answer

(c)

Solution

A class attribute is a variable declared inside the class but outside all the methods. Hence, `units` is a class variable of class `Vehicle`.

Question 2

Which of the following are instance attributes of class `Vehicle`. [MSQ]

(a) `wheels`

(b) `name`

(c) `units`

(d) None of these

Answer

(a), (b)

Solution

Any variable defined inside a method of a class is known as instance variables. Here `name` and `wheels` are instance variables.

Question 3

An object is created for each class `Vehicle`, `Motor` and `Car` in any order. There is no objects created prior to this from these classes. Assume `v`, `m` and `c` are the objects of the class `Vehicle`, `Motor` and `Car` respectively. What will be the value of `Vehicle.units` after creating these objects? [NAT]

Answer

3

Solution

Class `Vehicle` is inherited by class `Motor` and class `Motor` is then inherited by class `Car`. The variable `units` is a class variable and is increased by 1 every time an object of `Vehicle` class, or an object of subclass of `Vehicle` is created.

Question 4

Which of the following are valid and error free calls to instance methods? `v1` and `v2` are two instances of the class `Vehicle`.

Please answer based on the class definition given before. [MCQ]

(a) `Vehicle.get_speed()`

(b) `v1.get_speed()`

(c) `Vehicle.spec()`

(d) `v2.spec()`

(e) `start(v1)`

Answer

(d)

Solution

Instance methods are the methods which are called with respect to an instance (object). Since, `Vehicle` class defines `spec()` method and `v2` is its object, option (d) `v2.spec()` is the only valid option.

Question 5

Which of the following is a valid and error free call? `v1` and `v2` are two objects of the class `Vehicle`.

Please answer based on the class definition given before. [MCQ]

(a) `Vehicle.spec()`

(b) `Vehicle.start()`

(c) `v1.stop()`

(d) `v2.start()`

Answer

(b)

Solution

The method `start()` and `stop()` do not have any parameters. This means it does take any object as its argument. In other words, these two methods can not be called from any object of `Vehicle`. These should be called only from the class `Vehicle`. Hence, the valid option is (b) `Vehicle.start()`. Method `spec()` takes an argument which is an object of `Vehicle`, this indicates the method is an instance method. It should be called from the class with an object as an argument such as `Vehicle.spec(v1)`. Therefore, option (a) is incomplete and incorrect. `v1.spec()` and `v2.spec()` would have been valid calls.

Question 6

Which of the following are valid and error free calls? `c1` is an object of the class `Car`. [MSQ]

(a) `c1.spec()`
(b) `Car.get_speed(c1)`
(c) `c1.check_speed(c1)`
(d) `c1.start()`

Answer

(a), (b)

Solution

The `car` class has `spec()` and `get_speed()` as an instance method. While calling an instance method from the class, an object should be passed as an argument to the method.

Question 7

Which of the following code produces below output. `v` , `m` and `c` are objects of the class `Vehicle`, `Motor` and `Car` respectively. [MSQ]

```
starting
stopping
```

(a)

```
Car.start()
Car.stop()
```

(b)

```
Car.start()
Motor.stop()
```

(c)

```
Vehicle.stop()
Vehicle.start()
```

(d)

```
v.start()
v.stop()
```

Answer

(a), (b)

Solution

Class `Vehicle` is inherited by class `Motor` and class `Motor` is then inherited by class `Car`. `start()` and `stop()` method can be called only from the class itself as it does not accept any argument. Therefore, option (a) and (b) are valid options.

Question 8

Which of the following relationships holds True. [MSQ]

(a) `Vehicle` is the child/subclass of `Motor`

(b) `Motor` is the parent/super class of `Car`

(c) `Car` is parent/superclass of `Vehicle`

(d) `Motor` is a child/subclass of `Parent`

Answer

(b), (d)

Solution

Please refer the common code snippet.

Question 9

Which type of inheritance exists among the class `Vehicle`, `Motor` and `Car`. [MCQ]

(a) Simple inheritance

(b) Hierarchical inheritance

(c) Multiple inheritance

(d) Multilevel inheritance

Answer

(d)

Solution

The relationship is called `Multilevel` inheritance. Class `Vehicle` is inherited by class `Motor` and class `Motor` is then inherited by class `Car`.

Question 10

Which of the following variables can not be inherited further in the child class or can not be accessed outside class. [MCQ]

(a) `name`

(b) `wheels`

(c) `speed`

(d) `__car_type`

Answer

(d)

Solution

A private variable is accessible only within the class attributes and methods not outside the class. Hence, (d) is the correct answer.