COSC 1437, Introduction to Classes and Objects

Study Guide

True/False questions

T 1. Functions that are defined for objects are called methods. – instance methods

T 2. The first parameter of a Python instance method definition is referred to as self.

F 3. If an object has instance variables, it cannot use local variables.

T 4. In a Python class, we commonly refer to the constructor as `__init__`.

F 5. The `__string__` method in Python represents the class objects as a string

T 6. In Python, classes do not enforce information hiding.

F 7. Hiding the details of an object in a class definition is called instantiation.

T 8. An object is an instance of a class.

T 9. The `__str__` method is called when the following functions are invoked on the object and return a string:
`print()`
`str()`

---

10. What Python keyword starts a class definition?
(a) `def` (b) `class` (c) object (d) `__init__`

---

11. An instance method definition with two formal parameters is generally called with how many actual parameters?
(a) zero (b) one (c) two (d) three

---

12. A method definition is most similar to a(n)
(a) loop (b) module (c) import statement (d) function definition

---

13. The term applied to hiding details inside class definitions is
(a) obscuring (b) subclassing (c) documentation (d) encapsulation

---

14. What does the `__init__()` the function do in Python?
(a) Initializes the class for use
(b) called when a new object is instantiated.
(c) Initializes all the data attributes to zero
(d) None of the above.

---

15. The first parameter of each _____ in a class must be the `self` parameter which refers to the calling object.
instance method

---

16. The dunder method `__eq__` overloads the `==` operator.

---

17. The `__str__` dunder method in Python gets invoked when we `print` the object.

---

18. A Python convention for defining methods that are private to a class is to prefix the method name with
(a) private (b) a pound sign (#) (c) an underscore (_)
(d) a double underscore (__) (e) a hyphen (-) (f) Python has no such mechanism

19. What is the output of the following code?

```
class Point:
    def __init__(self, x = 0, y = 0):
      self.x = x + 1
      self.y = y + 1

if __name__ == "__main__":
    p1 = Point()
    print(p1.x, p1.y)
```

1 1
↑ space

20. What is the output of the following code?

```
class Point:
    def __init__(self, x = 0, y = 0):
      self.x = x + 1
      self.y = y + 1

    def __str__(self):
          return f"({self.x}, {self.y})"

if __name__ == "__main__":
    point1 = Point(2, 3)
    print(point1)
```

(3, 4)
↑ space

21. What is the output of the following code?

```
class Point:
    def __init__(self, x = 0, y = 0):
      self.x, self.y = x, y

    def __sub__(self, other):
        x = self.x + other.x
        y = self.y + other.y
        return Point(x,y)

if __name__ == "__main__":
      p1     = Point(3, 4)
      p2     = Point(1, 2)
      result = p1 - p2
      print(result.x, result.y)
```

Output
4 6
↑ space

$p_1.x = 3$
$p_1.y = 4$
$p_2.x = 1$
$p_2.y = 2$

self.x = p1.x = 3
self.y = p1.y = 4
other.x = p2.x = 1
other.y = p2.y = 2

x = 4, y = 6
result.x = 4 result.y = 6

22. What is the output of the following code?

```
class Foo:
      def printLine(self, line = 'Python'):
            print(line)

if __name__ == "__main__":
      o1 = Foo()
      o1.printLine('Java')
```

Java

23. Which of the following statements is incorrect about the following code?

```
class People():

  def __init__(self, name):
    self.name = name

  def namePrint(self):
    print(self.name)

if __name__ == "__main__":
      person1 = People("Sally")
      person2 = People("Louise")
      person1.namePrint()
```

(a) `person1` and `person2` are two different instances of the `People` class.
(b) The `__init__` method is used to set initial values for attributes.
(c) the parameter `self` is not needed in `def namePrint(self):`
(d) `person2` has a different value for 'name' than `person1`.

---

24. Which of the following statements is true?
(a) In Python, same operator may behave differently depending upon operands.
(b) You can change the way operators behave in Python.
(c) Special method `__add()__` is called when **+** operator is used. (on user-defined objects)
(d) All of the above.

↑ technically, when the left operand in user defined

---

25. The _____ keyword defines a template indicating the data that will be in an object of the class and the functions that can be called on an object of the class.
(a) class (b) object (c) Class (d) instance

---

26. In Python, __add__ is a magic (dunder) method which should be defined in the user class to enable the use of the **+** operator with its objects.

---

27. The __eq__ dunder method in Python gets called when the `==` operator is called for object comparison.

---

28. When adhering to Pythonic conventions, the first parameter of the constructor `__init__` must be self.

---

29. The first parameter of each _____ in a class must be the `cls` parameter which refers to the class.

class method

self → instance method

---

30. The dunder method __lt__ overloads the `<` operator.

---

31. The class constructor creates an object in the memory and invokes the `__init__` method.

↑ technically, __new__

---

32. In a user defined class, which function would we write to overload the **+** operator?
(a) __plus__ (b) __add__ (c) __sub__ (d) __mul__ (e) none of the others

33. In a user defined class, which function would we write to overload the * operator?
(a) __plus__ (b) __add__ (c) __sub__ (d) __mul__ (e) none of the others

34. In a user defined class, which function would we write to overload the / operator?
(a) __truediv__ (b) __div__ (c) __floor__ (d) __mul__ (e) none of the others

35. In a user defined class, which function would we write to overload the - operator?
(a) __neg__ (b) __minus__ (c) __sub__ (d) __plus__ (e) none of the others

36. _____ is not a keyword, but by convention it is used to refer to the current instance (object) of a class.
(a) `class` (b) `def` (c) `self` (d) `init`

37. Which of the following would be a correct way to define an initializer method?
(a) `def __init__(title, author):`
(b) `def __init__(self, title, author):`
(c) `def __init__():`
(d) `__init__(self, title, author):`

38. In Python, all the members of the class are _____ by default.
(a) public (b) private (c) protected (d) internal

39. Which of the following statement is correct?
(a) Class attributes are the variables defined directly in the class that is shared by all objects of the class.
(b) Class attributes are the variables defined inside the class method.
(c) Class attributes are objects of the class.
(d) None of the above

40. When correctly implementing the OOP, which of the following represents a distinctly identifiable real-world entity?
(a) class (b) object (c) method (d) data attribute

41. Which of the following is required to create a new instance of the class?
(a) A constructor (b) A class (c) A value-returning method (d) A `None` method

42. Which of the following statements can be used to check, whether an object obj is an instance of class A or not?
(a) `obj.isinstance(A)` (b) `A.isinstance(obj)`
(c) `isinstance(obj, A)` (d) `isinstance(A, obj)`

example: isinstance(frac2, int)

43. Strictly speaking, a new instance of class is created by the _____ method and initialized by the _____ method.
(a) `__new__` `__init__` (b) `__init__` `__new__`
(c) `__new__` `__new__` (d) `__init__` `__init__`

44. What is the output of the following code?

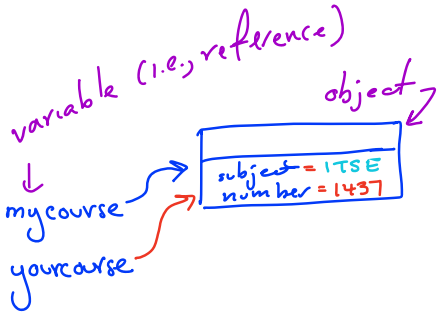
```
class courseName:
    def __init__(self, pre, num):
        self.subject = pre
        self.number = num

    def setSubject(self, value):
        self.subject = value

    def setNumber(self, value):
        self.number = value

if __name__ == "__main__":
      subject     = "COSC"
      mycourse    = courseName(subject, 1437)
      yourcourse  = mycourse
      subject     = "ITSE"
      yourcourse.setSubject(subject)
      print(mycourse.subject, mycourse.number)
```

OUTPUT:
ITSE 1437
↑ space



45. What is the output of the following code?

```
class A:
    def __init__(self):
        self.x      = 1
        self.__y    = 1

    def getY(self):
        return self.__y

    def getX(self):
        return self.x

if __name__ == "__main__":
    a = A()
    print(a.__y)
    print(a.getX())
```

← Error

46. What convention does Python employ to effectively declare members to be private?
(a) It doesn't have any such convention
(b) name mangling
(c) leading single underscore
(d) trailing single underscore
(e) leading double underscore

both techniques are employed

47. What are the dunder (magic) methods in Python?
(a) Methods that start with a double underscore.
(b) Methods that start and end with a double underscore
(c) Methods that start with a single underscore
(d) Methods that start and end with a single underscore

48. The `__add__` dunder method _____.
(a) Returns addition of two numbers
(b) Overloads * operator
(c) Should be overridden to overload + operator
(d) Only a) and c) are correct

49. In order to overload `==` operator to apply to a user defined class, which magic (dunder) method must be written?
    (a) `__comp__` (b) `__eq__` (c) `__equal__`
    (d) `__ne__` (e) `__init__` (f) `__add__`

50. Which of the following statements is most accurate for the declaration `x = Circle()`?
    (a) x contains an int value. (b) x contains an object of the Circle type.
    (c) x contains a reference to a Circle object. (d) You can assign an int value to x.

    ↑ some say x points to a Circle object

51. What will be the output of the following code snippet?

```
class Sales:
    def __init__(self, id):
        self.id = id
        id      = 100

if __name__ == "__main__":
      val = Sales(123)
      print (val.id)
```

local variable

val.id = 123

    (a) SyntaxError (b) 100 (c) 123 (d) None of the above

52. What will be the output of the following?

```
s = "\t\tWelcome\n"
print(s.strip())
```

    (a) `\t\tWelcome\n` (b) `Welcome\n` (c) `\t\tWELCOME` (d) `Welcome`

53. Which of the following is the output of the following Python code?

```
list1    = [1, 3]
list2    = list1
list1[0] = 4
print(list2)
```

    (a) [1, 3] (b) [4, 3] (c) [1, 4] (d) [1, 3, 4]

54. Which of the following is the output of the following code?

```
class A:
    def __init__(self, i = 2, j = 3):
        self.i, self.j = i, j

    def __str__(self):
        return f"i = {self.i}, j = {self.j}"

    def __eq__(self, other):
        return self.i * self.j == other.i * other.j

if __name__ == "__main__":
    x = A(1, 2)
    y = A(2, 1)
    print(x == y)
```

    (a) `True` (b) `False` (c) 1 (d) `None` (e) 2

55. What will be the output of the following Python code?

```
class Test:
    def __init__(self, a = "Hello World"):
        self.a = a

    def display(self):
        print(self.a)

if __name__ == "__main__":
    obj = Test()
    obj.display()
```

(a) The program has an error because constructor can't have default arguments
(b) Nothing is displayed
(c) "Hello World" is displayed
(d) The program has an error display function doesn't have parameters

---

56. What will be the output of the following Python code?

```
class Test:
    def __init__(self,a):
        self.a = a

    def display(self):
        print(self.a)

if __name__ == "__main__":
    obj = Test()
    obj.display()
```

look here

(a) Runs normally, doesn't display anything
(b) Displays 0, which is the automatic default value
(c) Error as one argument is needed when creating the object
(d) Error as display function requires an argument

---

57. What will be the output of the following Python code?

```
class Test:
    def __init__(self):
        self.variable = 'Old'
        self.Change(self.variable)

    def Change(self, var):
        var = 'New'

if __name__ == "__main__":
    obj = Test()
    print(obj.variable)
```

obj.variable = 'Old'

(a) Run-time error
(b) 'New' is printed
(c) 'Old' is printed
(d) Nothing is printed
(e) TypeError

58. Assuming that the following code is not being run as an import from another module, what will be the output of the following Python code?

```
class Demo:
    def __init__(self):
        pass

    def test(self):
        print(__name__)

if __name__ == "__main__":
    obj = Demo()
    obj.test()
```

(a) A run-rime error (i.e., exception) is thrown
(b) `__main__`
(c) Demo
(d) test

59. The assignment of more than one function to a particular operator is known as _____.
(a) Operator over-assignment
(b) Operator overriding
(c) Operator overloading
(d) Operator instance

60. What will be the output of the following Python code?

```
def add(c, k):
    c.test = c.test + 1
    k      = k + 1

class A:
    def __init__(self):
        self.test = 0

if __name__ == "__main__":
    Count = A()
    k     = 0

    for i in range(0,25):
        add(Count,k)

    print("Count.test =", Count.test, end = ", ")
    print("k =", k)
```

a) A run-rime error (i.e., exception) is thrown
b) Count.test = 25, k = 25
c) Count.test = 25, k = 0
d) Count.test = 20, k = 0

61. Which of the following Python code creates an empty class `A`?
a) `class A: return`
b) `class A: pass`
c) `class A: ellipsis`
d) `class A: None`

62. What are the methods which begin and end with two underscore characters called?
(a) Dunder methods, magic methods
(b) In-built methods
(c) User-defined methods
(d) Additional methods

63. The special (dunder) method `__init__` needs to be explicitly called during object creation.
(a) True (b) False

↳ constructor

64. What will be the output of the following Python code?

```
class Student:
   def __init__(self, roll_num, grade):
      self.num           = roll_num
      self.grade   = grade

   def display (self):
      print("Roll number : ", self.num,  ", Grade: ", self.grade)

stud1        = Student(34, 'A')
stud1.age    = 27
print(hasattr(stud1, 'age'))
```

+ public
− private
protected

a) Error as `age` isn't defined (b) `True` (c) `False` (d) 7

65. Write a Python `Person` class which conforms to the following UML diagram.

Person
+first: str +last: str +age: int
Person() +setAge(age:int) +setFirst(first:str) +setLast(last:str) +getAge() :int +getFirst() : str +getLast() : str

Person() represents the constructor (i.e., __init__ method).

66. Write a Python `Employee` class which conforms to the following UML diagram.

Employee
-first: str -last: str -id: int
Employee() +setFirst(first:str) +setLast(last:str) +getId() :int +getFirst() : str +getLast() : str -setId(id:int)

private

Employee() represents the constructor (i.e., __init__ method).

66.

```
class Employee:
    def __init__(self, first, last, id):
        self.__first = first
        self.__last  = last
        self.__id    = id

    def setFirst(self, first):
        self.__first = first

    def setLast(self, last):
        self.__first = last

    def getId(self):
        return self.__id

    def getFirst(self):
        return self.__first

    def getLast(self):
        return self.__last
    def __setId(self, id):
        self.__id = id
```

67. Write a Python `Dog` class which conforms to the following UML diagram.

Dog
+name: str +breed: str +age: int
Dog() +speak() +setAge(age:int) +getAge() :int

Dog() represents the constructor (i.e., __init__ method).
When the speak method is invoked, "Woof Woof Woof" should be printed if the age is less than 2 (years) and "Woof" should be printed if the age is greater than or equal to 2 years.

---

68. Write a Python `Point` class which conforms to the following UML diagram.

Point
+x: float +y: float
Point() +distance(): float

Point() represents the constructor (i.e., __init__ method).
The distance method should compute the distance of the point from the origin (i.e., $\sqrt{x^2 + y^2}$)

67.

```
class Dog:
    def __init__(self, name, breed, age):
        self.name = name
        self.breed = breed
        self.age = age

    def speak(self):
        if self.age < 2:
            print("Woof Woof Woof")
        else:
            print("Woof")

    def setAge(self, age):
        self.age = age

    def getAge(self):
        return self.age
```

68.

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance(self)
        return (self.x * self.x + self.y * self.y) ** (0.5)
```