Introduction to python final exam Part 1

duration: 60 MIN

Name : _____

Group No.:_____

PROBLEM 1 (Reading and Debugging Programs)

PART(A) Consider the following program:

```
1    list1 = ['red', 'black', 'green']
2    list2 = ['purple', 'gold']
3    print(list2)
4    list1[-1] = 'yellow'
5    print(list1 + list2)
6    print(list1)
7    list1.append(list2)
8    print(list1)
```

a) What is the value of list1 when the program completes?

b) What is the value of list2 when the program completes?

c) I want the result of list1 (when printed in line 8) to be [’red’, ’black’, ’yellow’, ’gold’].

What line of code would you modify?

PART (B) The following code checks if the last character in name2 is a substring of name1

```
1   name1 = 'North Carolina'
2   name2 = 'Sylva'
3   if name2[5] in name1:
4       print("It is there!!")
5   else:
6       print("It is NOT there!!")
```

This program doesn’t execute. It produces an IndexError: string index out of range

a) What line of code contains the error?

b) What is the reason for the error?

c) What line of code fixes this error to ensure it works correctly?

Problem 2: (Object Diagramming and Terminology)

(A) answer the following questions

1. In Python, a *class* is _____ for a concrete object.
   - a blueprint
   - a nuisance
   - a distraction
   - an instance
2.

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

The correct way to instantiate the above Dog class is:

- Python
```
Dog.create("Rufus", 3)
```

- Python
```
Dog()
```

- Python
```
Dog("Rufus", 3)
```

- Python
```
Dog.__init__("Rufus", 3)
```

3. Object and class attributes are accessed using _____ notation in Python.

4. In Python, a function within a class definition is called a:
    - a callable
    - a method
    - an operation
    - a class function
    - a factory
5. What's the output of the following code snippet?

```
Python >>>
>>> class Dog:
...     def walk(self):
...         return "*walking*"
...
...     def speak(self):
...         return "Woof!"
...
>>> class JackRussellTerrier(Dog):
...     def speak(self):
...         return "Arff!"
...
>>> bobo = JackRussellTerrier()
>>> bobo.walk()
```

    - Arff!
    - Woof!
    - AttributeError: 'JackRussellTerrier' object has no attribute 'walk'
    - *walking*

6. By using the same code at Q 5 what is the output of the next line ?

```
bobo.speak()
```

    - Arff!
    - CanineError: Dog malfunction
    - Woof!
    - *walking*

7. What's the output of the following code snippet?

```
>>> class Dog:
...     def walk(self):
...         return "*walking*"
...
...     def speak(self):
...         return "Woof!"
...
>>> class JackRussellTerrier(Dog):
...     def talk(self):
...         return super().speak()
...
>>> bobo = JackRussellTerrier()
>>> bobo.talk()
```

   - CanineError: Tail curvature exceeded
   - Woof!
   - *walking*

8. Which of the below is a getter method for the number of wheels?

```
class Car(object):
    def __init__(self, w, d):
            self.wheels = w
            self.doors = d
            self.color = ""
```

   - def get_wheels():
     return wheels
   - def get_wheels():
     return self.wheels
   - def get_wheels(self):
     return wheels
   - def get_wheels(self):
     return self.wheels

9. You create a car with mycar = Car(4, 2). Which is a line of code to change the color of mycar to "red"?

```
class Car(object):
    def __init__(self, w, d):
            self.wheels = w
            self.doors = d
            self.color = ""
    def paint(self, c):
            self.color = c
```

   - Car.paint("red")
   - mycar.paint("red")
   - mycar.paint(red)
   - mycar.paint(Car, "red")

(b) The questions on the right pertain to the code on the left. Some questions may have multiple correct answers. Write down only one answer

```
1  class A():
2      x = 1
3
4      def __init__(self, n):
5          self.y = n
6          A.x += 1
7
8      def p(self):
9          print(self.y)
10         self.y += 3
11         self.r()
12
13     def r(self):
14         self.y += 2
15         print(self.y)
16
17 class B(A):
18     x = 10
19
20     def __init__(self, n):
21         super().__init__(n)
22         sum = self.y + B.x
23         self.m = sum
24
25     def r(self):
26         self.y += self.x
27         print(self.m)
28
29 a = A(1)
30 b = B(2)
```

an **object folder** is created when Python executes line _____

a **class folder** is created when Python executes line _____

an **object attribute** is created on line _____

a **class attribute** is created on line _____

a **superclass** definition begins on line _____

a **class method** definition begins on line _____

an attribute definition **that overrides another** begins on line _____

a method definition **that overrides another** begins on line _____

a **local variable** is created on line _____

a **global variable** is created on line _____