keyboard_arrow_down

PYTHON EXAM 2 (30 Questions - 1 Hour)

OOPs Concept (10 Questions)

Classes and Objects, Methods & Attributes, and OOP Concepts

Section 1: Classes and Objects (3 Questions)

1. What will be the output of the following code?

```
class   Car:      def
__init__(self, brand):
self.brand   =   brand
my_car = Car("Toyota")
print(my_car.brand)
```

A) Error

B) None

C) Toyota

D) Car

Answer: C

2. Which of the following statements is correct about Python classes? A) A class is an instance of an object.

B) A class is a blueprint for creating objects.

C) A class can have only one object.

D) A class cannot have attributes.

Answer: B

3. What is the difference between an instance variable and a class variable?

A) Class variables are shared among instances, whereas instance variables are unique to each instance.

B) Instance variables are shared among instances, whereas class variables are unique to each instance.

C) Both are the same.

D) None of the above.

Answer: A

Section 2: Methods & Attributes (4 Questions)

4. What will be the output of the following code?

```
class Test:  x = 5  def __init__(self, y):
        self.y = y

obj = Test(10)
print(obj.x, obj.y)
```

A) 5 10

B) 10 5

C) Error

D) None

Answer: A

5. In Python, which of the following is correct regarding instance methods and class methods? A) Instance methods take `self` as the first parameter, while class methods take `cls` . B) Class methods modify instance variables.

C) Instance methods can be called without an object.

D) None of the above.

Answer: A

6. What will be the output of the following code?

```
class Demo:  def
   __init__(self, value):
       self.value = value

    def update(self, new_value):
        self.value = new_value

obj = Demo(20)
obj.update(50)
print(obj.value)
```

A) 20

B) 50

C) Error

D) None

Answer: B

7. What is the correct way to define a class method?

A)

```
def my_method(self):
pass
```

B)

```
@classmethod def
my_method(cls):
pass
```

C)

```
@staticmethod def
my_method():
pass
```

D) None of the above.

Answer: A

Section 3: OOP Concepts (3 Questions)

8. Which of the following is NOT a pillar of Object-Oriented Programming (OOP)?

A) Encapsulation

B) Abstraction

C) Compilation

D) Inheritance

Answer: C

9. What is method overriding in Python?

A) Defining multiple methods with the same name but different arguments in the same class. B) A child class providing a specific implementation of a method that is already defined in its parent class.

C) A method with a default argument value.

D) None of the above.

Answer: A

10. What will be the output of the following code?

```
class Parent:
def show(self):
       print("Parent class")

class Child(Parent):
def show(self):
print("Child class") obj
= Child() obj.show()
```

A) Parent class

B) Child class

C) Error D) None

Answer: B

Advanced Concepts (12 Questions)

Decorators, Generators, Iterators, and the differences between Iterators and Generators

---

Section 1: Decorators (3 Questions)

1. What is a decorator in Python?

A) A function that modifies another function’s behavior without changing its code

B) A function that defines a new class

C) A built-in function to optimize loops

D) A function that automatically executes before the main program

Answer: A

2. What will be the output of the following code?

```
def decorator(func):
 def wrapper():
        print("Before function call")
       func()
        print("After function call")
 return wrapper


@decorator def
greet():
print("Hello!")


greet()
```

A) Prints only `"Hello!"`

B) Error due to incorrect decorator syntax

C) Prints `"Before function call"` , `"Hello!"` , `"After function call"` D) Does nothing

Answer: C

3. Which of the following decorators is built-in in Python?

A) `@function`

B) `@staticmethod`

C) `@classmethod`

D) Both B and C **Answer: B**

Section 2: Generators (3 Questions)

4. What will be the output of the following code? `def`

```
 my_generator():


 yield 1

     yield 2
    yield 3


 gen = my_generator()
 print(next(gen)) print(next(gen))
```

A) `1 2`

B) `1 3`

C) `2 3`

D)

Error

Answer: B

5. What is the difference between `return` and `yield` in Python functions?

A) `return` sends back a value and exits, while `yield` saves the function state and continues

B) `return` is used in loops, while `yield` is used in functions

C) `yield` terminates a function immediately, whereas `return` does not

D) There is no difference

Answer: C

6. What will happen if we call `next()` on a generator that has no more values left?

A) It restarts the generator

B) It raises a `StopIteration` exception

C) It returns None   D) It prints an empty list **Answer:**

Section 3: Iterators (3 Questions)

7. Which of the following methods must a class implement to be considered an iterator in Python?

A) `__next__()` only

B) `__iter__()` and `__next__()`

C) `__iter__()` only

D) `next()` only

Answer:B

8. What will be the output of the following code?

```
my_list = [1, 2, 3]
iter_obj = iter(my_list)
print(next(iter_obj))
print(next(iter_obj))
```

A) `1 2`

B) `1 3`

C) `[1, 2]`

D) Error

Answer: A

9. How can you manually iterate over an iterator in Python?

A) Using a `for` loop

B) Using the `next()` function

C) Converting it to a list

D) All of the above **Answer: A**

Section 4: Iterator vs Generator (3 Questions)

10. How is a generator different from an iterator?

A) Generators use `yield` , while iterators use `return`

B) Generators automatically create `__iter__()` and `__next__()` methods

C) Generators are memory-efficient compared to iterators

D) All of the above

Answer: A

11. What will be the output of the following code?

```
def my_gen():

yield 10

yield 20 gen =

my_gen()

print(iter(gen) is gen)
```

A) `True`

B) `False`

C) Error D) None

Answer: A

12. Which of the following is **NOT true about iterators and generators?**

A) Generators can only be iterated once

B) Iterators can be reset to the beginning

C) Both generators and iterators implement `__iter__()` and `__next__()`

D) Generators are more memory-efficient than lists **Answer:C**

Python – Production Level (8 Questions)

Docstrings, Error Handling, File Handling, and Modularization

---

Section 1: Docstrings (2 Questions)

1. What is the purpose of a docstring in Python?

A) To define a function

B) To document the purpose and usage of a function, class, or module

C) To print debugging information

D) To execute code inside a string

Answer:B

2. How do you access a function's docstring in Python?

A) `function_name.doc`

B) `function_name.__doc__`

C) `function_name.get_doc()`

D) `doc(function_name)`

Answer: C

Section 2: Error Handling in Python (2 Questions)

3. What will be the output of the following code?

```
try:  print(5 / 0)
except
ZeroDivisionError:  print("Cannot divide
by zero!")
```

A) Error

B) 0

C) Cannot divide by zero!

D) None

Answer: C

4. Which of the following statements is true about `finally` in a try-except block? A) It executes only if an exception occurs.

B) It executes only if no exception occurs.

C) It always executes, regardless of whether an exception occurs or not.

D) It prevents exceptions from occurring.

Answer: A

Section 3: File Handling in Python (2 Questions)

5. What will be the output of the following code?

```
file = open("test.txt", "w")
file.write("Hello, Python!")
file.close()


file = open("test.txt", "r")
print(file.read())
file.close()
```

A) `Hello, Python!`

B) Error: File not found

C) None

D) Empty output

Answer: A

6. What is the correct way to read a file line by line?

A) `file.read_all()`

B) `file.readline()`

C) `file.readlines()` D) `file.read_line_by_line()` **Answer: D**

Section 4: Modularization in Python (2 Questions)

7. What is the primary purpose of modularization in Python?

A) To make code less readable

B) To improve code reusability and maintainability

C) To execute code faster

D) To write functions inside a single large script

Answer: B

8. How do you import a function named `calculate` from a module named `math_operations` ?

A) `import calculate from math_operations`

B) `from math_operations import calculate`

C) `import math_operations.calculate`

D) `math_operations.import calculate`

Answer:C
