Sort, Recursive functions, lambda functions and OOP [MCQ] (Version : 0)

TEST

● **Correct Answer**

Answered in 8.47 Minutes

Question 1/10

What is the Big-O notation for the worst-case time complexity of the following function?

```
def find_element(arr, x):
    for element in arr:
        if element == x:
            return True
    return False
```

animal_list = ["Lion", "Elephant", "Tiger", "Giraffe", "Zebra", "Panda", "Kangaroo", "Penguin"]

find_element(animal_list , "Penguin")

- [✓ O(n)]
- O(1)
- O(n^2)
- O(log n)

Explanation:

O(1) is incorrect as the time complexity is not constant; it depends on the size of arr.
O(n^2) is incorrect as the function does not have nested loops.
O(log n) is incorrect as the function does not employ a divide-and-conquer approach like binary search.

Question 2/10

Rearrange the following lines to correctly implement a bubble sort algorithm for a list.

What is the output if…

```
def bubble_sort(items):
return items
for j in range(len(items)-1-i):
if items[j] > items[j+1]:
""" Implementation of bubble sort."""
items[j], items[j+1] = items[j+1], items[j]
for i in range(len(items)):
```

- [✗] Output = [11, 12, 22, 25, 34, 64, 90] for (64, 34, 25, 12, 22, 11, 90)
- [✓] Output = [11, 12, 22, 25, 34, 64, 90] for [64, 34, 25, 12, 22, 11, 90
- [ ] Output = [11, 12, 22, 25, 36, 64, 90] for 64, 34, 25, 12, 22, 11, 90
- [ ] Output = [64, 34, 25, 12, 22, 11, 90] for [64, 34, 25, 12, 22, 11, 90]

Explanation:

[11, 12, 22, 25, 34, 64, 90] for (64, 34, 25, 12, 22, 11, 90) is incorrect because a tuple is immutable and cannot be sorted and reassigned.
[11, 12, 22, 25, 36, 64, 90] for 64, 34, 25, 12, 22, 11, 90 the input of bubble_sort() expects a list, not multiple integers.
TypeError for [64, 34, 25, 12, 22, 11, 90] is incorrect because no error is produced when bubble sort is used this way.
[64, 34, 25, 12, 22, 11, 90] for [64, 34, 25, 12, 22, 11, 90] is incorrect because the code was not arranged correctly, returning the input list.

Question 3/10

Rearrange the following code to correctly define a Python class representing a BankAccount with a method to deposit funds.

Which of the following lines of code will open a new bank account with a balance of 100, deposit an additional 50, and print the balance?

```
def deposit(self, amount):
 self.balance += amount
class BankAccount:
 def __init__(self, balance):
 self.balance = balance
```

- ☐ 200 for account.deposit(50); BankAccount(100); print(account.balance)
- ☐ TypeError for account = BankAccount(150); account.deposit('50'); print(account.balance))
- ✓ 150 for account = BankAccount(100); account.deposit(50); print(account.balance)
- ☐ 150 for account.deposit(50); account = BankAccount(150); print(account.balance)

Explanation:

200 for account.deposit(50); BankAccount(100); print(account.balance) is incorrect because the bank account from another answer option was used and 50 added.
150 for account.deposit(50); account = BankAccount(150); print(account.balance) is incorrect because we are depositing funds into an account that does not exist, or creating an account with a starting balance of account = BankAccount(150)
TypeError for account = BankAccount(150); account.deposit('50'); print(account.balance)) is incorrect because while a TypeError is raised, this code does not deposit an additional 50 units in because the syntax is incorrect.

Question 4/10

Rearrange the following lines to create a function that uses recursion to reverse a string.

Which of the following answers is correct, given the correctly rearranged code and function calls provided?

```
if len(s) == 0:
return s
return r(s[1:]) + s[0]
def r(s)
```

☐ 'Error for r('programming')

☐ 'gnimmargorP' for r('programming')

✓ 'gnimmargorp' for r('programming')

☐ 'prgram' for r(r('programming')[4:])

Explanation:

'gnimmargorP' is incorrect because r('programming') does not contain a capitalised “P”.

'prgram' for r(r('programming')[4:]) is incorrect because calling . r('programming')[4:] produces ‘margorp’, which is then reversed to form ‘progam’

'Error' is incorrect as the rearranged function should not generate an error; it correctly reverses the string.

Question 5/10

What is the lambda expression required to filter out negative numbers from a list:

```
numbers = [-2, 5, -9, 8, -1, 10]
result = filter(, numbers)
print(list(result))
```

☐ lambda x: x == 0

☐ lambda x: -x

✓ lambda x: x >= 0

☐ lambda x: x < 0

Explanation:

‘lambda x: x < 0’ is incorrect because it only shows negative numbers, not positive.
‘lambda x: x == 0’ is incorrect as it filters out all numbers that are not equal to 0.

‘lambda x: -x’ is incorrect because it simply negates the value of x and does not return a boolean value as expected by the filter function, so all values are True in the list.

Question 6/10

Rearrange the following code to correctly implement a Python class for a Circle, including a method to calculate the circumference.

Which of the following options give the correct answer, given that the radius of the circle is 3?

```
def __init__(self, radius):
self.radius = radius
return 2 * 3.14159 * self.radius
class Circle:
def circumference(self):
```

- [ ] None for Circle().circumference(3)
- [✓] 18.849539999999998 for Circle(3).circumference()
- [ ] TypeError for Circle().circumference(3)
- [ ] SyntaxError for Circle(3).circumference()

Explanation:

SyntaxError for Circle(circumference) is incorrect because the output is 18.849539999999998. TypeError for Circle().circumference() is incorrect because the class call is formatted incorrectly. None for Circle().circumference() is incorrect because the class call is formatted incorrectly.

Question 7/10

Rearrange the following code to correctly define a Python class representing a rectangle, which includes methods for calculating area and perimeter.

What will be the output of the following code?

```
return 2 * (self.width + self.height)
def perimeter(self):
return self.width * self.height
class Rectangle:
def __init__(self, width, height):
self.width = width
self.height = height
def area(self):

rect = Rectangle(3, 4)
print("Area:", rect.area(), "Perimeter:", rect.perimeter())
```

[✓ Area: 12 Perimeter: None]

☐ Area: 14 Perimeter: 18

☐ TypeError

☐ Area: 12 Perimeter: 14

Explanation:

Area: 7 Perimeter: 18 is incorrect because 12 is the area of a rectangle with width 3 and height 4, but the perimeter is 14.
TypeError is incorrect as the code, when properly arranged, does not produce any type errors.
Area: 12 Perimeter: None is incorrect because the perimeter method is defined and returns a valid calculation.

Question 8/10

Rearrange the following lines of code to create a lambda function that squares each number in a list, using the map function:

```
print(list(result))
result = map(lambda x: x**2, numbers)
numbers = [1, 2, 3, 4, 5]
```

☐ SyntaxError

☐ [2, 4, 6, 8, 10]

☐ [1, 2, 3, 4, 5]

✓ [1, 4, 9, 16, 25]

Explanation:

[1, 2, 3, 4, 5] is incorrect because it shows the original list, not the squared values.
SyntaxError is incorrect because the syntax is correct when properly arranged.
[2, 4, 6, 8, 10] is incorrect because it doubles the numbers, not squaring them.

Question 9/10

Identify the line that violates the PEP 8 style guidelines:

```
class ExampleClass:
def __init__(self,value):
self.value = value
def get_value(self):
return self.value
```

- ☐ return self.value
- ☐ class ExampleClass:
- ✗ self.value = value
- ☐ def __init__(self,value):
- ☐ def get_value(self):

Explanation:

class ExampleClass: is incorrect because it is correctly formatted.

self.value = value is incorrect because it is correctly formatted.

def get_value(self): is incorrect because it is correctly formatted.

return self.value is incorrect because it is correctly formatted.

Question 10/10

The following Python code is meant to calculate the total area of two different shapes, but it contains an error. Identify and correct the error.

```
class Shape:
    def __init__(self, name):
        self.name = name

class Circle(Shape):
    def __init__(self, radius):
        super().__init__("Circle")
        self.radius = radius

    def area(self):
        return 3.14 * self.radius * self.radius
class Square(Shape):
    def __init__(self, side):
        super().__init__("Square")
        self.side = side

    def area(self):
        return self.side * self.side

circle = Circle(3)
square = Square(4)
total_area = circle.area + square.area
print("Total Area:", total_area)
```

- ☐ The error is a missing self parameter in the area method of the Square class.
- ☐ There is no error in the code.
- ■ The error is that circle.area and square.area are method references, not method calls.
- ✗ The error is a mistake in the Circle class's _init_ method. The super().__init__("Circle") call should come after initialising the radius attribute.

Explanation:

The error is a missing self parameter in the area method of the Square class: This is incorrect because both Circle and Square classes correctly define their area methods with the self parameter. The actual error is in calculating total_area, not in the method definitions.

The error is a mistake in the Circle class's init method. The super().init("Circle") call should come after initialising the radius attribute: This is incorrect as the order of super().__init__ and attribute initialisation in the Circle class is not a concern in this case. The real error is in the calculation of total_area.

There is no error in the code: This is incorrect because the error is in how total_area is calculated. The methods circle.area and square.area need to be called with parentheses to compute the total area.