Practice Set

OOPS - E2UC201C

1. Explain the procedural and object-oriented paradigm along with advantages and disadvantages.
2. What are pillars of OOP paradigm. Explain each one of them with example.
3. Develop a program for creating a class named Students, initializing attributes such as name, age, and grade during object instantiation.
4. Write a program to establish a valid empty class named Students, devoid of any properties.
5. Design a program for creating a child class Teacher that inherits properties from the parent class Staff.
6. Devise a Python program to define a Vehicle class with instance attributes max_speed and mileage.
7. Create a Python program to define a person class with attributes like name, country, and date of birth, and implement a method for determining the person's age.
8. Implement a Python class named BankAccount representing a bank account, incorporating attributes such as accountNumber, name, and balance. Provide a complete code for the BankAccount class with various methods like withdraw and deposit and check balance money.
9. Discuss the importance of access modifiers (e.g., public, private, protected) in achieving the objectives of Object-Oriented Programming (OOP) in Python.
10. Explain how a destructor can be defined and utilized in a Python class.
11. Explore the usage of the 'private' access modifier in Python and discuss its implications.
12. Enumerate the different types of polymorphism in Object-Oriented Programming (OOP) in Python.
13. Demonstrate method overriding in Python to achieve runtime polymorphism with a code example.
14. Discuss the advantages and disadvantages of compile-time polymorphism compared to runtime polymorphism in Python.
15. Explain how method overriding showcases runtime polymorphism in Python.
16. Create a Python code example illustrating data overloading by defining multiple attributes with the same name in a class.
17. Differentiate between data overloading and data overriding in Python.
18. Define data overriding and provide an example demonstrating it in Python.
19. Present a Python code example showcasing data overloading using multiple constructors with different parameters.
20. List and explain the exceptions in Python.
21. Describe how to handle exceptions using try, except, and finally blocks in Python.
22. Explain the purpose of the raise statement in Python.
23. Discuss the use of the except clause without an exception type in Python.
24. Highlight the differences between the except and else blocks in Python exception handling.
25. Explain how to handle multiple exceptions in a single except block in Python.
26. Determine if it is possible to include other statements between 'try,' 'catch,' and 'finally' blocks in Python.
27. Elaborate on the concept of an unreachable catch block error in Python.
28. Differentiate between errors and exceptions in Python.
29. Enumerate and explain the types of exceptions in Python.
30. Explain the purpose of the finally block in Python exception handling.
31. Discuss the concept of custom exceptions in Python.
32. Explain the purpose of the assert statement in Python.
33. Differentiate between a built-in exception and a custom exception.

34. Describe how to re-raise an exception in Python.
35. Clarify the purpose of inheritance in Python.
36. Define the role of the super() function in Python.
37. Explain the StackOverflowError in Python.
38. Explore whether it is possible to override a superclass method that throws an unchecked exception with checked exceptions in the sub-class.
39. Identify the class defined as a superclass for all types of errors and exceptions in Python.
40. Discuss the correct combination of try, catch, and finally blocks in Python.
41. Create a Bus child class that inherits from the Vehicle class, incorporating fare calculations with maintenance charges.
42. Imagine you are designing a system to model animals. Create a base class called Animal with attributes like name and age. Now, derive two subclasses Mammal and Bird. How would you handle specific attributes or methods for each subclass, such as fur_color for mammals and wing_span for birds?
43. Design a system to model vehicles. Start with a base class Vehicle with attributes like make, model, and year. Derive subclasses like Car, Truck, and Motorcycle. How would you handle specific methods or attributes for each subclass, such as cargo_capacity for trucks and seating_capacity for cars?
44. Imagine you are developing a file processing application. Design a scenario where the user provides a file name as input, and your program needs to handle exceptions such as file not found, permission denied, or a general I/O error while reading or writing to the file.
45. Create a scenario where a user inputs two numbers and selects an operation (addition, subtraction, multiplication, division). How would you handle exceptions like division by zero, invalid input (non-numeric values), or any other mathematical errors during the operation?
46. Create a base class Shape with methods to calculate area and perimeter. Derive subclasses like Rectangle, Circle, and Triangle. How would you ensure that each subclass implements its own area and perimeter calculation logic?
47. Is it possible to have multiple constructors in a Python class? If yes, how can you achieve this functionality? If not, what are the alternatives?
48. Create a class with a parameterized constructor that takes arguments to initialize instance variables. Provide an example and explain how object creation and initialization work in Python.
49. Explain the relationship between Python's garbage collection mechanism and destructors. How does Python manage memory deallocation, and when does the destructor get called?
50. How are constructors and destructors inherited in Python classes?
51. Create a class, Triangle. Its __init__() method should take self, angle1, angle2, and angle3 as arguments. Create two objects of Triangle class T1 and T2. For T1 object the value of angle1, angle2, and angle3 is 80,30 and 40 respectively. For T2 object the value of angle1, angle2, and angle3 is 70,50 and 60 respectively. Create a method named check_angles which returns True if the sum of angle1, angle2, and angle3 is equal to 180, and False otherwise.
52. Define a class named Souvenir with attributes title and author. Create an object of this class representing the souvenir “OOPS" written by "D. Mark" and published by “TMH”. Print out the title, author and publisher of the book.
53. Define classes Person, Student, and Professor. Implement methods specific to each class, such as enroll_course for Student and teach_course for Professor.
54. Create a class named 'Employee' with attributes 'name' for storing the Employee's name and ‘monthly salary' for storing a list of salaries .How do you make a method inside the 'Employee' class to figure out the average salary of a Employee using their list of monthly salary?

55. Define a class called MathOperations with a method called operate() that takes either one or two arguments. If one argument is provided, the method should return the cube of the argument. If two arguments are provided, the method should multiply the two arguments and return the product. Demonstrate the usage of this class by creating an object and calling the operate() method with both one and two arguments.
56. Design a class Product with attributes product_name and product_price. Create a class Order that can add products, delete products, and compute the total amount.

57. Imagine you are designing a software system for a zoo. One of the key features is managing different types of animals. You decide to use abstract classes and methods to represent these animals.

    1. Create an abstract class called Animal with the following abstract methods:
        - make_sound: This method should be implemented by subclasses to make the specific sound of each animal.
        - move: This method should be implemented by subclasses to define how each animal moves.
    2. Create two subclasses of Animal called Lion and Monkey.
        - For the Lion class, implement the make_sound method to roar and the move method to walk.
        - For the Monkey class, implement the make_sound method to chatter and the move method to climb.
    3. Instantiate objects of both Lion and Monkey classes and demonstrate their functionalities by calling their make_sound and move methods.
58. Explain the following with examples: Class methods, instance methods, static methods and abstract methods, class variables, instance variables, abstract classes, interface
59. Explain different types of inheritance supported in python.
60. Create a package named "shapes" that includes two modules: "circle.py" and "rectangle.py." In the "circle.py" module, define a class Circle with methods to calculate the area and circumference of a circle. In the "rectangle.py" module, define a class Rectangle with methods to calculate the area and perimeter of a rectangle. Use the formulae area=length×width, area=pi×r×r and perimeter=2×(length+width), circumference=2×pi ×radius. Create a script named "main.py" outside the "shapes" package. In this script, import the Circle and Rectangle classes from the "shapes" package, create instances of both classes, and demonstrate the calculation of area and circumference for a circle, as well as area and perimeter for a rectangle.
61. Explain the significance of following built-in modules: sys, random, math, os, datetime, itertools, functools, re, collections
62. Explain comprehensions(list,set,dictionary,tuple) in python with example.
63. Describe the following functions append(), extend(), pop(), reverse(),remove(), sort() and insert() of list by writing a code for each of them .
64. Differentiate between: a) sets, tuples, lists and dictionary b)Packages and Modules
65. Demonstrate each of the following with a piece of python code: lambda, zip, map, reduce, filter, pip, pypi
66. Use each of map, filter, and reduce to fix the broken code.

```
from functools import reduce

# Use map to convert temperatures from Celsius to Fahrenheit
celsius_temperatures = [0, 20, 37, 100]

# Use filter to print only the odd numbers
numbers = [1, 4, 9, 16, 25, 36, 49]
```

```
# Use reduce to find the sum of these numbers
more_numbers = [10, 20, 30, 40, 50]

#Fix all three respectively
map_result = list(map(lambda c: c, celsius_temperatures))
filter_result = list(filter(lambda x: x , numbers))
reduce_result = reduce(lambda x, y: x + y, more_numbers,10)

print(map_result)
print(filter_result)
print(reduce_result)
```

67. Explain the following: decorators and generators, staticmethod and classmethod decorators.
68. Illustrate with example re module along with its functions: findall(),match(),search()
69. What is Tkinter in python? Describe its role and advantages.
70. What are widgets in Tkinter? Explain the following widget along with their uses, syntax, options and methods: Frame, Label, Button, Entry, Text, Radiobutton, Checkbox, listbox, Combobox, , Spinbox, OptionMenu, Scale, Canvas, Checkbutton. Also write code for setting these widgets on the window.

71. How would you create a window in Tkinter and set its title and dimensions? Write code for the same.

72. The following program creates a graphical user interface that allows the user to enter a list of numbers and either find the smallest value or calculate the average. Assume that the functions minList and avgList each take a string as a parameter and return the appropriate value. For example, minList("2 5 1 4") returns 1, and avgList("2 5 1 4") returns 3.0.

```
from tkinter import * #1
window = Tk() #2
frame = Frame(window) #3
frame.pack() #4
def compute(): #5
    input_str = entry.get() #6
    choice = option.get() #7
    if choice == 1: #8
        result = minList(input_str) #9
    else: #10
        result = avgList(input_str) #11
    result_label.config(text=str(result)) #12
    result_label.update() #13
entry = Entry(frame) #14
entry.pack() #15
result_label = Label(frame, text="") #16
result_label.pack() #17
option = IntVar() #18
radio_min = Radiobutton(frame, text="Min", variable=option, value=1,
command=compute) #19
radio_min.pack() #20
radio_avg = Radiobutton(frame, text="Avg", variable=option, value=2,
command=compute) #21
radio_avg.pack() #22
```

```
exit_button = Button(frame, text="Exit", command=window.destroy) #23
exit_button.pack() #24
window.mainloop() #25
```

Explain the purpose of each line or group of lines in the program, and describe exactly how the user should interact with it. Ignore any errors that may occur due to inappropriate input.

73. Differentiate between create_line, create_oval and create_rectangle methods used in canvas widget.
74. How Create_line method can be used to draw different types of lines, rectangle, square and triangle?
75. What is geometry management? Differentiate pack(), grid() and place() methods.
76. Explain pack(), grid() and place() with their options with the help of suitable code.
77. How would you choose color and date in tkinter? Explain.
78. Differentiate between Iterator and iterable. How iter and next method works?
79. How do various operations and manipulations can be done on images in tkinter. Explain the method used by giving piece of code.
80. What are virtual environments? What is their importance and how they are created?
81. How multiple decorators can be applied to only one function?
82. Write a function that counts the number of occurrences of each word in a given string (Use re module). The function should ignore case and punctuation.
83. Write a function that uses regular expressions to find all words starting with a specific letter (case-insensitive) in a given string.
84. Write a function that uses regular expressions to validate if a given string is a valid email address.
85. Write a function that takes a list of tuples where each tuple contains the price and quantity of an item. Use map to calculate the total price for each item and reduce to calculate the grand total.
86. Write a function that takes a list of integers, filters out the odd numbers, and returns a list of squares of the even numbers. Use filter to filter out the odd numbers and map to square the even numbers.
87. Write a function that takes a list of integers, filters out the negative numbers, and returns the sum of the positive numbers. Use filter to filter out the negative numbers and reduce to sum the positive numbers.
88. Write a function that takes a list of elements and an integer n, and returns a new list with n randomly selected elements from the original list. The original list should be shuffled before selecting the elements.
89. What is Regex Module in Python? Discuss various functions and meta characters included in this module
90. How geometry method works in tkinter window?

** Prepare problems shared for Lab ETE.

**********All the very Best**********