QN 1: MAY 2024

Explain how to read numeric values from a file, perform some operations, and

then write the results back to the file?

ANS:

Reading numeric values from a file, performing operations, and writing the results back to the file involves the following steps:

**STEPS:-**

**1-OPEN THE FILE IN THE RAED MODE TO EXTRACT NUMERIC VALUES**

**2-CONVERT THE EXTRACTED DATA INTO A USABLE NUMERIC FORMATE(INTEGER or FLOAT)**

**3-PERFORM THE REQUIRED OPERATIONS IONS ON THE NUMBERS**

**4-OPEN THE FILE IN THE WRITE MODE AND STORE THE PROCESSED DATA**

**CONSIDER THE FOLLOWING PYTHON PROGRAM prgm.py WHICH OPENS A FILE NAMED number.txt**

**AND TAKE NUMBERS AND STORE THE SQUARES OF THE NUMBERS BACK TO THE FILE**

|  |
| --- |
| **number.txt** |
| **1**  **5**  **6**  **8**  **11**  **33**  **57**  **25**  **21**  **100** |

|  |
| --- |
| Prgm.py |
| file=open(“number.txt”,”r”)  numbers=[int(line.strip() for line in file]  sq\_num=[x\*\*2 for x in numbers]  file1=open(“number.txt”,”w”)  for num in sq\_num:  file1.write(str(num)+”\n”) |

Output:

|  |
| --- |
| number.txt |
| 1  25  36  64  121  1296  3249  625  441  10000 |

In the given program ,file “number.txt ” open in read mode and read the entire number from the file in to a list named”numbers” and then found the squares of each number and store back into another list of name “sq\_num” anf open the file”number.txt” write mode and write the content of the listb “sq\_num” into the file.

QN 2 : JUNE 2023

Assume that the variable data refers to the string "Python rules!". Use a string method to perform the following tasks:

a. Obtain a list of the words in the string.

b. Convert the string to uppercase.

c. Locate the position of the string "rules" .

d. Replace the exclamation point with a question mark.

ANS:

a)-

>>>string=”Python rules!”

>>>word=string.split()

b)-

>>>string=”Python rules!”

>>>s=string.upper()

c)-

>>>string=”Python rules!”

>>> pos=string.find(“rules”)

>>> print(“position is”,pos)

d)-

>>>string=”Python rules!”

>>>s=atring.replace(“!”,”?”)

>>>print(s)

QN 3:JUNE 2023

Write the output of following python code :

S = “Computer”

print(S[::2])

print(S[::-1])

print(S[:])

ANS:

1-cmue

2-retupmoc

3-computer

QN 4:MAY 2023

Explain the concepts namespace, scope, and lifetime in the case of Python

programming language.

These three concepts define how variables and objects are stored and accessed in Python.

**a. Namespace**

A **namespace** is a collection of names (variable names, function names, etc.) mapped to objects. Python has three types of namespaces:

* **Built-in Namespace:** Contains built-in functions and exceptions (e.g., print(), len()).
* **Global Namespace:** Includes variables and functions defined at the top level of a script/module.
* **Local Namespace:** Contains variables inside a function, valid only within that function.

**b. Scope**

**Scope** determines where a variable can be accessed. Python has four types of scope (LEGB Rule):

1. **Local Scope:** Variables defined inside a function.
2. **Enclosing Scope:** Variables in the enclosing function (for nested functions).
3. **Global Scope:** Variables defined at the module level.
4. **Built-in Scope:** Includes Python’s built-in functions and variables.

|  |
| --- |
| Example: |
| x = 10 # Global scope  def outer\_function():  y = 20 # Enclosing scope  def inner\_function():  z = 30 # Local scope  print(x, y, z) # Accessing global, enclosing, and local variables  inner\_function()  outer\_function() |

**c. Lifetime**

**Lifetime** refers to how long a variable exists in memory.

* **Global variables** exist throughout the program execution.
* **Local variables** exist only while the function is running and are destroyed after execution.

|  |
| --- |
| Example: |
| def my\_function():  temp\_var = 50 # Created when the function runs  print(temp\_var)  my\_function()  # temp\_var is destroyed after function execution |

QN 5:MAY2023

What are mutable and immutable properties in the case of Python data

structures? Give one example each for mutable and immutable data structures in

Python.

**a. Mutable Data Structures**

A **mutable** object can be changed after creation.

|  |
| --- |
| Example: **List** |
| my\_list = [1, 2, 3]  my\_list.append(4) # Modifies the list  print(my\_list) # Output: [1, 2, 3, 4] |

**b. Immutable Data Structures**

An **immutable** object cannot be changed after creation.

|  |
| --- |
| Example: **string** |
| my\_string =”helo” #creating string |

QN 6:June 2022

Illustrate the use of negative indexing of list with example.

ANS:

Negative indexing in Python allows accessing elements from the end of a list. The last element has an index of -1, the second last is -2, and so on.

Negative indexing provides an easy way to access elements from the end without needing to calculate the length of the list.

|  |
| --- |
| EXAMPLE : slicing |
| my\_list = [10, 20, 30, 40, 50]  # Accessing elements using negative indexing  print(my\_list[-1]) # Last element: 50  print(my\_list[-2]) # Second last element: 40  print(my\_list[-3]) # Third last element: 30  print(my\_list[-3:]) # Output: [30, 40, 50] (last 3 elements)  print(my\_list[:-2]) # Output: [10, 20, 30] (excluding last 2 elements) |