python-unit-iv

August 18, 2024

[ ]:
```
1.Mention the significance of 'sys' module. List out its main features␣
 ↪including 'sys.argv' variable.

The sys module in Python provides access to some variables used or maintained␣
 ↪by the Python interpreter and to functions that interact strongly with the␣
 ↪interpreter.
It is a built-in module that allows you to interact with the interpreter in␣
 ↪various ways, making it significant for controlling the execution␣
 ↪environment.
=>sys.argv
=>sys.path
=>sys.exit
=>sys.stdin
=>sys.stdout
=>sys.stderr
```

[ ]:
```
2.What is the role of the __init__ method in a Python class? Provide an example␣
 ↪to illustrate its usage.

The __init__ method in a Python class is a special method called a constructor.␣
 ↪Its primary role is to initialize the newly created object of the class.
The __init__ method is automatically invoked when a new instance of a class is␣
 ↪instantiated.
class Student:
    def __init__(self, roll_no, name):
        self.roll_no = roll_no
        self.name = name

    def display_details(self):
        print(f"Roll No: {self.roll_no}")
        print(f"Name: {self.name}")

# Creating an instance of the Student class
student1 = Student(101, "Alice")

# Displaying the details of the student
student1.display_details()
```

[ ]:
```
3.Write a python program to handle the divide by zero exception

def divide_numbers(numerator, denominator):
    try:
        result = numerator / denominator
        print(f"The result of the division is: {result}")
    except ZeroDivisionError:
        print("Error: Cannot divide by zero.")

numerator = float(input("Enter the numerator: "))
denominator = float(input("Enter the denominator: "))
divide_numbers(numerator, denominator)
```

[ ]:
```
4.Distinguish between files and modules

Files
=>Store code or data
=>Contains Python code or data
=>Functions, classes, variables, data
=>Text files, JSON files, Python script files

Modules
=>Encapsulate reusable code
=>Used to import and reuse code
=>Python definitions and statements
=>Standard library modules ('math', 'os') or custom modules
```

[ ]:
```
5.What is meant by Garbage collection in python. How is it performed?

Garbage collection in Python refers to the automatic management of memory by␣
 ↪the Python interpreter. It is the process by which Python reclaims memory␣
 ↪that is no longer in use,
ensuring efficient memory usage and preventing memory leaks.
=>Automatic Process
=>Cyclic Garbage Collection
=>Reference Counting
=>Memory Management
```

[ ]:
```
i)Write a python program to create a text file "MyFile.txt" and ask the user to␣
 ↪write separate 3 lines with three input statements from the user

with open("MyFile.txt", "w") as file:
    line1 = input("Enter the first line: ")
    line2 = input("Enter the second line: ")
    line3 = input("Enter the third line: ")
    file.write(line1 + "\n")
    file.write(line2 + "\n")
```

```
    file.write(line3 + "\n")
print("The lines have been written to MyFile.txt successfully.")

ii)Write a python program to find the total occurrences of a specific word from␣
 ↪a text file

filename = input("Enter the filename: ")
word_to_count = input("Enter the word to count its occurrences: ")
word_count = 0
with open(filename, "r") as file:
    for line in file:
        words = line.split()
        word_count += words.count(word_to_count)
print(f"The word '{word_to_count}' occurs {word_count} times in the file␣
 ↪'{filename}'.")

iii)Write a Python program to read first n lines of a file

filename = input("Enter the filename: ")
n = int(input("Enter the number of lines to read: "))
with open(filename, "r") as file:
    count = 0
    for line in file:
        print(line, end="")
        count += 1
        if count >= n:
            break
```

[ ]:
```
i)Write a Python program to know the cursor position and print the text␣
 ↪according to below-given specifications (8 marks)
1.        Print the initial position
2.        Move the cursor to 4th position
3.        Display next 5 characters
4.        Move the cursor to the next 10 characters
5.        Print the current cursor position
6.        Print next 10 characters from the current cursor position

with open("sample.txt", "w") as file:
    file.write("This is a sample text file used to demonstrate cursor␣
 ↪operations.")

with open("sample.txt", "r") as file:
    initial_position = file.tell()
    print(f"Initial cursor position: {initial_position}")
    file.seek(3)
    next_five_chars = file.read(5)
    print(f"Next 5 characters from 4th position: {next_five_chars}")
```

```
    file.seek(file.tell() + 10)
    current_position = file.tell()
    print(f"Current cursor position: {current_position}")
    next_ten_chars = file.read(10)
    print(f"Next 10 characters from current cursor position: {next_ten_chars}")

ii)Write a python program that uses class to store and display the employee's␣
 ↪name and salary and print the total employee count whose details are␣
 ↪available.

class Employee:
    employee_count = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.employee_count += 1

    def display_employee(self):
        print(f"Name: {self.name}, Salary: {self.salary}")

    def display_employee_count(cls):
        print(f"Total Employee Count: {cls.employee_count}")

emp1 = Employee("John Doe", 50000)
emp2 = Employee("Jane Smith", 60000)
emp3 = Employee("Emily Davis", 55000)

emp1.display_employee()
emp2.display_employee()
emp3.display_employee()

# Display total employee count
Employee.display_employee_count()
```

[ ]:
```
Provide a detailed explanation, along with examples, on the steps involved in␣
 ↪opening, closing, reading, and writing to a file using Python.

1. Opening a File:
To open a file in Python, you typically use the open() function. The open()␣
 ↪function returns a file object that allows you to interact with the file.
The basic syntax for opening a file is:

file_object = open(filename, mode)

where:
```

```
filename: This is the name of the file you want to open, including the path if␣
 ↪the file is not in the current directory.
mode: This specifies the mode in which the file is opened. It can be:
'r': Open for reading (default).
'w': Open for writing, truncating the file first.
'a': Open for writing, appending to the end of the file if it exists.
'b': Binary mode (e.g., 'rb' or 'wb').
'+': Open for updating (reading and writing).
Example:
file = open("example.txt", "r")
file = open("output.txt", "w")
file = open("log.txt", "a")

2. Reading from a File
Once you have opened a file for reading ('r' mode), you can read its contents␣
 ↪using methods like read(), readline(), or readlines().

read(): Reads the entire file as a single string.
readline(): Reads one line from the file.
readlines(): Reads all lines into a list where each line is an element.
Examples:
with open("example.txt", "r") as file:
    content = file.read()
    print(content)
with open("example.txt", "r") as file:
    line = file.readline()
    while line:
        print(line.strip())
        line = file.readline()

with open("example.txt", "r") as file:
    lines = file.readlines()
    for line in lines:
        print(line.strip())

3. Writing to a File
When a file is opened in write ('w') or append ('a') mode, you can write data␣
 ↪to it using the write() method.

Example:
with open("output.txt", "w") as file:
    file.write("This is a sample line.\n")
    file.write("Writing another line.\n")

4. Closing a File
It's important to close a file after you've finished working with it to free up␣
 ↪system resources and ensure that all data is written to the file properly.
```

```
In Python, you can close a file automatically by using the with statement,␣
 ↪which ensures that the file is properly closed when the block inside with␣
 ↪completes.

Example:
with open("example.txt", "r") as file:
    content = file.read()
    print(content)

Full Example: Reading and Writing
with open("output.txt", "w") as file:
    file.write("This is a sample line.\n")
    file.write("Writing another line.\n")

with open("output.txt", "r") as file:
    content = file.read()
    print(content)
```

[ ]:
```
i)Write a python program to count the total number of uppercase, lower case,␣
 ↪and digits used in the text file "merge.txt".

uppercase_count = 0
lowercase_count = 0
digit_count = 0

with open("merge.txt", "r") as file:
    content = file.read()
    for char in content:
        if char.isupper():
            uppercase_count += 1
        elif char.islower():
            lowercase_count += 1
        elif char.isdigit():
            digit_count += 1

print(f"Total uppercase letters: {uppercase_count}")
print(f"Total lowercase letters: {lowercase_count}")
print(f"Total digits: {digit_count}")

ii)Write a program to count a total number of lines and count the total number␣
 ↪of lines starting with 'A', 'B', and 'C'

total_lines = 0
count_A = 0
count_B = 0
count_C = 0
```

```
with open("sample.txt", "r") as file:
    lines = file.readlines()
    total_lines = len(lines)
    for line in lines:
        if line.startswith('A'):
            count_A += 1
        elif line.startswith('B'):
            count_B += 1
        elif line.startswith('C'):
            count_C += 1

print(f"Total number of lines: {total_lines}")
print(f"Number of lines starting with 'A': {count_A}")
print(f"Number of lines starting with 'B': {count_B}")
print(f"Number of lines starting with 'C': {count_C}")
```