Module - 3

Solved Question & Answers on Python Programming

1. Explain join() and split() method with examples [-8M-] [Jan-Feb 2023]

Ans For join() → refer pg no 9

For split() → refer pg no 10

2. Explain with examples (i) isalpha() (ii) isalnum() (iii) isspace(). [-6M-] [Jan-Feb 2023]

Ans For (i) isalpha → refer pg no 8
(ii) isalnum
(iii) isspace

3. Develop a python code to determine whether the given string is a palindrome / not a palindrome. [-6M-] [Jan-Feb 2023]

Ans
```
text = input ("Enter a string") → Input the string
if text == text [::-1]: → check if the string is the same when reversed
    print ("The string is a palindrome")
else :
    print ("The string is not a palindrome")
```

text[::-1] → This reverses the string using slicing

text == text[::-1] → Compares the original string with its reversed version.

If they are the same, the string is a palindrome, otherwise it's not.

Input : Enter a string :
madam

Output :

The string is a palindrome!

Input : Enter a string :
hello

Output :

The string is not a palindrome!

4. Explain the concept of file handling. Also explain reading and writing process with suitable example [-8M-] [Jan-Feb 2023]

Ans Refer chapter 8 → page no 18 - 21. Imp

5. Explain the concept of file path. Also discuss absolute and relative file path [-6M-] [Jan-Feb 2023]

Ans Refer chapter 8 → page no 18 - 21

For absolute v/s relative path refer page no 21 - 22 //

6. Briefly explain saving variables with shelve module [-6M-] [Jan-Feb 2023]

Refer page no 27 //

7. Explain the following methods with suitable examples :

(i) upper() (ii) lower() (iii) is_upper()

(iv) is_lower() [-8M-] [June-July 2023]

V.V.V Imp

Refer page no 7 to 9 //

8. Illustrate with example opening of a file with open() function, reading the contents of the file with read() and writing to files with write(). [10M] [June/July 2023] V.VV.Imp

Refer Chapter 8 → page no 18-21

9. Explain the steps involved in adding bullets to Wiki-Markup Support with appropriate code. [10M] → refer pg no 14 to 17

Ans In wiki markup, adding bullets (unordered list) is simple. You use an asterisk for each item. Indentation with additional asterisks creates sub-items (nested lists).

Steps to add Bullets :-

1. Use a single asterisk (*) for top level items
2. Use two asterisks (**) for second-level items (nested under)
3. Continue with more asterisks for deeper levels if needed.

Eg:-

```
* Item 1
* Item 2
** Sub-item 2.1
** Sub-item 2.2
* Item 3
```

→ Rendered output

- Item 1
- Item 2
  - Sub item 2.1
  - Sub-item 2.2
- Item 3

This structure keeps your content neat & easy to follow !!

10. Develop a program to sort the contents of a text file and write the sorted contents into a separate text file. [Use strip(), len(), list methods, sort(), append and file methods open(), readlines() & write()]. [8M]

Ans

1] Open the input file : Use open() to read all the file that contains the text you want to sort

2] Read the contents : Use readlines() to get all the lines as list

3] Strip whitespace : Use strip() to remove any extra spaces or newline characters from each line.

4] Sort the lines : Use the sort() method to arrange the lines in alphabetical order

5] Write the sorted lines : Open another file in write mode & write the sorted lines

```
with open ("input.txt", "r") as file :          # open the I/P file & read the lines.
    lines = file.readlines()
cleaned_lines = [line.strip() for line in lines if len(line.strip()) > 0]
                # Remove leading/trailing spaces from each line & filter out empty lines
cleaned_lines.sort()                            # sort the cleaned lines.
with open ("sorted_output.txt", "w") as sorted_file :   # write the sorted lines into a new file.
    for line in cleaned_lines :
```

```
        sorted_file.write(line + "\n")
    print("Contents sorted & written to c sorted_output.txt")
```

Ex:- Input File : input.txt

apple
banana
cherry
grape
date

Output File : sorted_output.txt

apple
banana
cherry
date
grape.

→ This program ensures the i/p files content is cleaned, sorted & written neatly into a new file.

11. (P) Briefly explain saving variables with shelve module [L6M]

Ans Refer pg no 27 to 29. [v.v.v imp]

12. Explain permanent delete & safe delete with a suitable python programming example to each.

Ans A permanent delete removes the item (like a file) immediately and can't be undone. If you accidently delete it, its gone unless you have a backup.

Eg: Using os.remove() to delete a file permanently

```
import os
file_name = "example.txt"   → file to be deleted
```

```
if os.path.exists(file_name):
    os.remove(file_name)          → permanently deletes the file
    print(f"{file_name} has been permanently deleted")
else:
    print(f"{file_name} does not exists")
```

2] Safe Delete :-

A safe delete involves precautions like moving the item to another location (like a "trash" folder) instead of permanently deleting it. This way, it can be recovered if needed

Eg:- Moving a file to a "Trash" folder instead of deleting it

```
import os
import shutil
file_name = "example.txt"
trash_folder = "trash"
os.makedirs(trash_folder, exists_ok = True)   ← ensures the trash folder exists before moving the file.
if os.path.exists(file_name):
    shutil.move(file_name, trash_folder)   ← moves the file to a trash folder instead of deleting it.
    print(f"{file_name} has been safely moved to the trash folder")
else:
    print(f"{file_name} doesn't exists)
```

Conclusion :-

Permanent delete : Deletes the file directly & can't be undone eg! os.remove(). Safe delete → moves the file to a safer location eg ("trash") so it can be recovered later eg shutil.move()

13. Explain string and useful string methods. [V.V. Imp]

(R) Refer page no 1 to 7, 8, 9, 10, 11, 12, 13.

14. Write a note on Indexing and slicing strings -

Refer page no 4 to 5

15. Write a note on IN and NOT IN operations with string.

(R) [V.V.V. Imp]

Refer page no 8

16 Justify the text with rjust(), ljust() and center()

(R) [V.V.V. Imp]

17. Write a note on the current working directory.

Refer pg no 21 to 22

18. Write a note on os path module.

Refer pg no 22 - 24

19. Write a program that uses the os.walk() function on the directory tree to show the folders subfolder & files with output [5M-]

Ans Python program that uses os.walk() to traverse a directory tree & display folders, subfolders & files -

```
import os
directory = "."
for root, dirs, files in os.walk(directory):
    print("Root : {root}")
    print("Subfolders")
    for d in dirs:
        print("    - {d}")
    print("Files")
    for f in files:
        print("    - {f}")
    print()
```

Specify the directory you want to walk through

Iterates through the given directory tree

current directory being traversed.

A list of subdirectories in the current directory.

A list of files in the current directory.

Output :-

```
.
├── folder 1
│   ├── file1.txt
│   └── file2.txt
├── folder 2
│   ├── subfolder 1
│   │   ├── file3.txt
│   │   └── file4.txt
└── file5.txt
```

↘ The program output :-

```
Root :
    Subfolders :
        - folder 1
        - folder 2
    Files :
        - file5.txt

Root : ./folder 1
    Subfolders :
    Files :
        - file1.txt
        - file2.txt

Root : ./folder 2
    Subfolders :
        - subfolder 1
```