String Processing and File Handling

UNIT - V

Unit-5: String Processing and File Handling

CO5 -Perform string manipulation and file operations to solve a given problem.

- Introduction to String
- Access String elements using index operator
- String functions
  - Basic functions: len, max, min
  - Testing functions: isalnum, isalpha, isdigit, isidentifier, islower, isupper, and isspace
  - Searching functions: endswith, startswith, find, rfind, count
  - Manipulation functions: capitalize, lower, upper, title, swapcase, replace, lstrip, rstrip, strip
  - Formatting functions: format, center, ljust, rjust
- Introduction to Text files
- File Handling functions:
  - Basic functions: open, close
  - Reading file: read, readline, readlines
  - Writing file: write, append, writelines

String

- String is the collection of the characters that is enclosed between single quotes, double quotes, or triple quotes.
- String is a immutable.
- The computer does not understand the characters; internally, it stores manipulated character as the combination of the 0's and 1’s.

Examples:

```
a = 'Hello'
print(a)

a = "Hello"
print(a)

a = """ABCDEFGHIJK"""
print(a)
```

Some More Examples of String:

```
s = 'hi'
print s[1]                  ## i
print len(s)                ## 2
print s + 'there'           ## hi there
```

```
for x in "college":
  print(x)
```

```
pi = 3.14
text = 'The value of pi is ' + pi                    ## NO, does not work
text = 'The value of pi is '  +str(pi)               ## yes
```

String Operations and Access String Elements Using Index Operator

- Access String
  - We can use square brackets along with index number to access individual character of string.
  - Index always start with 0.

```
Examples:

a = 'Hello'
print(a[0])
       Output: H
```

String Operations and Access String Elements Using Index Operator

- Traversing a String
  - We can traverse a string using for loop and while loop.

Examples : Using For Loop

```
a ='Hello'
for x in a:
       print(x, end=' ')
```

Output: Hello

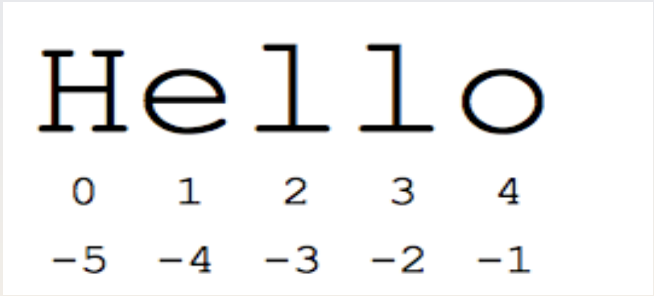
Examples : Using While Loop

```
a = "hello"
x = 0
while x < len(a):
       letter = a[x]
       print(letter, end='')
       x = x + 1
```

Output: Hello

String Operations and Access String Elements Using Index Operator

- Slicing a String
  - It means extract specific part of a string.
  - We can do this by providing starting and ending index by : in square bracket.



Examples:

```
s ='Hello’
pirnt(s[1:4])        #ell
pirnt(s[1:])         #ello
pirnt(s[:])          #Hello
```

String Operations and Access String Elements Using Index Operator

- Searching a String
  - You can search a single character or a word from a given string using a membership operator.

Examples:

```
s ='Hello’
pirnt(“llo” in s)          #true
pirnt(“llo” not in s)      #false
pirnt(“Llo” in s)          #false
pirnt(“o” in s)            #true
```

String Operations and Access String Elements Using Index Operator

- String Concatenation
  - Multiple string can be concat using + operator.

```
Examples:

s ='Hello'
a ='AVPTI’
Mystring = s + a
pirnt(Mystring)                #Hello AVPTI
```

String Operations and Access String Elements Using Index Operator

- String Repetition
  - You can repeated a string using * operator.

```
Examples:

s ='Hello'
Mystring = s * 2
pirnt(Mystring)                #HelloHello
```

String Basic Functions

len( )

- It returns the length of a string.
- `a = "Hello"`
- `print(len(a))  #5`

max( )

- Return the name with the lowest ASCII value, ordered alphabetically
- `x = min("Mahesh", "Suresh", "Ramesh")`
- `print(x)  #Mahesh`

min( )

- Return the name with the Highest ASCII value, ordered alphabetically
- `x = min("Mahesh", "Suresh", "Ramesh")`
- `print(x)  #Suresh`

Testing Functions

isalnum

- Returns True if all characters in the string are alphanumeric.
- s="Yagnik1"
- print(s.isalnum()) #True
- s="Yagnik@"
- print(s.isalnum()) #False

isalpha

- Returns True if all characters in the string are in the alphabet.
- print("zyz".isalpha()) # True
- print(“1ab".isalpha()) # False

isdigit

- Returns True if all characters in the string are digits.
- print("101.129".isdigit()) # False
- Print("101".isdigit()) # True

Testing Functions

isidentifier

- Returns True if the string is an identifier
- print("2bring".isidentifier()) #False
- print("bring".isidentifier()) #True

islower

- Returns True if all characters in the string are lower case
- print("abc".islower()) #True
- print("abX".islower()) #False

isupper

- Returns True if all characters in the string are upper case
- print("abc".isupper()) #False
- print("ABX".isupper()) #True

Testing Functions

isspace

- Returns True if all characters in the string are whitespaces
- print(" \n\t".isspace()) #True
- print(" Yagnik".isspace()) #False

Searching Functions

endswith

- Returns True if the string ends with the specified value, otherwise False.
- ```
  txt = "Hello, welcome to my world."
  x = txt.endswith("my world.")
  print(x)                    # True
  ```

startwith

- Returns True if the string start with the specified value, otherwise False.
- ```
  txt = "Hello, welcome to my world."
  x = txt.startswith("Hello")
  print(x)                    # True
  ```

Searching Functions

find

- The find() method finds the first occurrence of the specified value.
- The find() method returns -1 if the value is not found.
-
```
txt = "Hello, welcome to my world."
x = txt.find("e")
print(x)                    # 1
```

rfind

- The rfind() method finds the last occurrence of the specified value.
- The rfind() method returns -1 if the value is not found.
-
```
txt = "Hello, welcome to my world."
x = txt.rfind("e")
print(x)                    # 13
```

Searching Functions

count

- Returns the number of times a specified value appears in the string
- ```
  txt = "I love apples, apple are my favorite fruit"
  x = txt.count("apple")
  print(x)                    # 1
  ```
  ```

Manipulation Functions

capitalize

- Returns a copy of the original string and converts the first character of the string to a capital **(uppercase)** letter, while making all other characters in the string **lowercase** letters.
- ```
  txt = "good day"
  print(txt.capitalize())          # Good day
  ```

lower

- Returns a copy of string converted into lowercase.
- ```
  txt = "Good Day"
  print(txt.lower())       # good day
  ```

upper

- Returns a copy of string converted into uppercase.
- ```
  txt = "Good Day"
  print(txt.upper())       # GOOD DAY
  ```
  ```

Manipulation Functions

title

- Returns a string in which first character of each word is capital.
- ```
  txt = "good day"
  print(txt.title())                # Good Day
  ```

swapcase

- Returns a string by converting lowercase letters into uppercase and uppercase letters into lowercase.
- ```
  txt = "Good Day"
  print(txt.swapcase())             # gOOD dAY
  ```

replace

- Returns a string by replacing each occurance of search string by replacing string.
- ```
  txt ="Good Day"
  ```
- ```
  print(txt.replace('Good','food’)      # food Day
  ```

Manipulation Functions

lstrip

- Returns a string by removing white space from beginning of string.
- ```
  txt = "  good day"
  print(txt.lstrip())              #good day
  ```

rstrip

- Returns a string by removing white space from ending of string.
- ```
  txt = "good day    "
  print(txt.rstrip())              #good day
  ```

strip

- Returns a string by removing white space from beginning as well as ending of string.
- ```
  txt = "   good day    "
  print(txt.strip())               #good day
  ```
  ```

Formatting Functions

center

- The center() method will center align the string, using a specified character (space is default) as the fill character.
- Syntax: *string*.center(*length, character*)
- txt ="Good Day"
- print(txt.center(20)) # Good Day
- txt ="Good Day"
- print(txt.center(20,'*') # ******Good Day******

Formatting Functions

ljust

- The ljust() method will left align the string, using a specified character (space is default) as the fill character.
- Syntax: *string.ljust(length, character)*
- txt ="Good Day"
- print(txt.ljust(20)) # Good Day
- txt ="Good Day"
- print(txt.ljust(20,'*') # Good Day**********

Formatting Functions

rjust

- The ljust() method will rigth align the string, using a specified character (space is default) as the fill character.
- Syntax: *string.rjust(length, character)*
- txt ="Good Day"
- print(txt.rjust(20)) # Good Day
- txt ="Good Day"
- print(txt.rjust(20,'*’) # **********Good Day

Formatting Functions

format

- The format() method formats the specified value(s) and insert them inside the string's placeholder.
- The placeholder is defined using curly brackets: {}

- txt ="For only {price:.2f} Rs"
- print(txt.format(price = 49)) # For only 49.00 Rs

- txt ="For only {price:,} Rs"
- print(txt.format(price = 100000)) # For only 100,000 Rs

Introduction to Text files

- Files are named locations on disk to store related information. They are used to permanently store data in a non-volatile memory (e.g. hard disk).
- There are **two types** of files that can be handled in python, normal text files and binary files (written in binary language, 0s, and 1s).
- **Text files:**
  - In this type of file, Each line of text is terminated with a special character called EOL (End of Line), which is the new line character (‘\n’) in python by default.
  - "t" - Text - Default value. Text mode
- **Binary files:**
  - In this type of file, there is no terminator for a line, and the data is stored after converting it into machine-understandable binary language.
  - "b" - Binary - Binary mode (e.g. images)

File Handling functions : Basic functions: open, close

open()

- Python has a built-in open() function to create a new file or open existing file.
- `f = open("test.txt")` `# open file in current directory`
- `f = open("C:/Python38/README.txt")` `# specifying full path`

- Modes:
  - "r" - Read - Default value. Opens a file for reading, error if the file does not exist
  - "a" - Append - Opens a file for appending, creates the file if it does not exist
  - "w" - Write - Opens a file for writing, creates the file if it does not exist
  - "x" - Create - Creates the specified file, returns an error if the file exists
- `f = open("demofile.txt", "rt")` `# open file with specific mode`

File Handling functions : Basic functions: open, close

close()

- close() function closes the file and frees the memory space acquired by that file.
- It is used at the time when the file is no longer needed or if it is to be opened in a different file mode.
- `file1 = open("MyFile.txt","a")`
- `file1.close()`

File Handling functions : Reading File : read, readline, readlines

read()

- The function read() ,read some contents of a file based on the optional size and return the contents as a string.
- If you omit the size, the read() method reads from where it left off till the end of the file.
- If the end of a file has been reached, the read() method returns an empty string.

```
f = open("demofile.txt", "r")
print(f.read())
      #Hello! Welcome to demofile.txt This file is for testing
      purposes.Good Luck!


f = open("demofile.txt", "r")
print(f.read(33))
      #Hello! Welcome to demofile.txt Th
```

File Handling functions : Reading File : read, readline, readlines

readline()

- Read a single line from a text file and return the line as a string.
- If the end of a file has been reached, the returns an empty string

```
f = open("demofile.txt", "r")
print(f.readline())
      #Hello! Welcome to demofile.txt

f = open("demofile.txt", "r")
print(f.readline(5))
      #Hello
```

File Handling functions : Reading File : read, readline, readlines

readlines()

- Read all the lines of the text file into a list of strings.
- This method is useful if you have a small file and you want to manipulate the whole text of that file.

```
# open the data file
file = open("employees.txt")
# read the file as a list
data = file.readlines()
# close the file
file.close()
print(data)
```

```
#Hello! Welcome to employees.txt This file is data of
employees Details….
```

readline() vs readlines()

- **readlines()**
  - This method will read the entire content of the file at a time.
  - This method reads all the file content and stores it in the list.
  - This method reads up to the end of the line with readline () and returns a list.
- **readline()**
  - This method will read one line in a file.
  - A new line character is left at the string end and is ignored for the last line provided the file does not finish in a new line.
  - This method reads up to the end of the line with readline() and returns a list.

File Handling functions : Writing file: write

write()

- Inserts the string str1 in a single line in the text file.
- File_object.write(str1)
- Example :

```
file = open(“demo.txt", "w")
file.write(“Welcome to 75th Anniversary Celebration")
file.write(“AVPTI Amrit Mahotsav")
print(file.read())
file.close()
```

File Handling functions : Writing file: writelines

writelines()

- The writelines() method writes the items of a list to the file.
- Example :

```
f = open("demofile3.txt", "a")
f.writelines(["See you soon!", "Over and out."])
f.close()

#open and read the file after the appending:
f = open("demofile3.txt", "r")
print(f.read())
```

File Handling functions : Writing file: append

append Only (‘a’)

- Open the file for writing.

append and read (‘a+’):

- Open the file for reading and writing.

append( )

- When the file is opened in append mode in Python, the handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.
- Example:

```
# Append-adds at last
file1 = open("myfile.txt", "a")  # append mode
file1.write("Today \n")
file1.close()
```