Experiment No. 4

Aim: To implement programs based on basic I/O operations and string methods.

Problem statement:

A. To implement programs based on basic I/O operations.
B. To implement programs based on String methods.

A. To implement programs based on basic I/O operations.

Input Operations in Python 3:

- Use input() function.
- Whatever you enter as input, the input function converts it into a string.
- If you enter an integer value still input() function convert it into a string.

Syntax:

```
input(prompt)
```

Prompt: (optional): The string that is written to standard output without newline.
Return: String object

- By default input() function takes the user's input in a string.
- So, to take the input in the form of int/float, you need to use int()/ float() along with input function.

Output Operations in Python 3:

Python provides the print() function to display output to the console.

Syntax:

print(value(s), sep= ' ', end = '\n', file=file, flush=flush)

Parameters:

value(s):

- Single/multiple values can be written.
- Will be converted to string before printing.

sep='separator': (Optional)

- Specify how to separate the objects if there is more than one.
- Default :' '

end='end': (Optional):

- Specify what to print at the end.
- Default : '\n'

file: (Optional):

- An object with a write method.
- Default :sys.stdout (screen)

flush : (Optional):

- A Boolean, specifying if the output is flushed (True) or buffered (False).
- Default: False

Returns: It returns output to the screen.

Formatting Output:

- We can format the string literals, by starting a string with f or F before opening quotation marks or triple quotation marks. In this string, we can write Python expressions between { and } that can refer to a variable or any literal value.
- We can also use format() function to format our output to make it look presentable. The curly braces { } work as placeholders. We can specify the order in which variables occur in the output.

Example:

Input() function:

```
In [1]: name = input("Enter your name: ")
        print(name)

        Enter your name: mrunal
        mrunal

In [2]: str=input('enter a message')
        print(str)

        enter a messagehello
        hello

In [3]: a=input('enter first number')
        b=input('enter second number')
        c=a+b
        print('addition of two numbers is',c)

        enter first number23
        enter second number34
        addition of two numbers is 2334

In [4]: a=int(input('enter first number'))
        b=int(input('enter second number'))
        c=a+b
        print('addition of two numbers is',c)

        enter first number23
        enter second number34
        addition of two numbers is 57
```

print() function:

In [16]:
```
#using separators and end
print('M', 'R', 'U', sep ='#')
print("Hii", end = '@')
print("Hello")
```

```
M#R#U
Hii@Hello
```

In [22]:
```
#String formatting using f
name = 'mrunal'
print('Hello {name}')
print(f'Hello {name}')
```

```
Hello {name}
Hello mrunal
```

In [25]:
```
#formatting using format() function
a = 20
b = 10
print('The value of a is {} and b is {}'.format(a,b))
```

```
The value of a is 20 and b is 10
```

Questions:

1. Implement a python program to take an input message from the user and print it.
2. Implement a python program to take two values from user and apply all arithmetic operators on those values.
3. Implement a python program to accept a string from user and create a tuple containing that single string.
4. Implement a python program to create a list containing elements 10, 20, 30, 40, 50. Take an input from user and find out whether that value is present in a list or not.
5. Implement a python program to take name of the student from user and create a dictionary with key ‘name’ and value accepted from user. Apply possible dictionary methods on the same.
6. Implement a python program to create a set containing values ‘CSE’, ‘IT’, “ETRX’, ‘AERO’. Accept an input from user and add that element inside the set.
7. Implement a python program to show different print() function options.

8. Implement a python program to show different print() function formatting options.

B. To implement programs based on String methods.

Python String methods:

- Python has a set of built-in methods that you can use on strings.
- All string methods return new values.
- They do not change the original string.

Question: Implement a python program to accept a string from user and apply all the string methods on it.

Method	Description	Example	Output
capitalize()	Converts the first character to upper case.	str='university' str.capitalize()	'University'
casefold()	Converts string into lower case.	str1='University' str1.casefold()	'university'
center()	Returns a centered string. Padding is done using the specified fill char. Default filler is a space.	str1='University' str1.center(20,'@')	'@@@@@University @@@@@'
count()	Returns the number of times a specified value occurs in a string.	str1='University' str1.count('i')	2
encode()	Returns an encoded version of the string.	str1='University' str1.encode()	b'University'
endswith()	Returns true if the string ends with the specified value.	str1='University' str1.endswith('t')	False
expandtabs()	Sets the tab size of the string. Default tab size=8	str2='SGU\tAtigre' print(str2) str2.expandtabs(2)	SGU Atigre 'SGU Atigre'
find()	Searches the string for a specified value and returns the	str1='University' print(str1)	9

	position of where it was found.	str1.find('y')	
format()	Formats specified values in a string. (It copies a dictionary during method call)	str3 = 'My name is {}'.format('Gurunath') print(str3)	My name is Gurunath
format_map( )	Formats specified values in a string. (It makes a new dictionary during method call).	student = {'name':'Gurunath','addr':'Kolhapur'} print('{name} {addr}'.format_map(student))	Gurunath Kolhapur
index()	Searches the string for a specified value and returns the position of where it was found.	print(str1) str1.index('v')	University 3
isalnum()	Returns True if all characters in the string are alphanumeric (letters or numbers or both)	str1=’University’ str1.isalnum()	True
isalpha()	Returns True if all characters in the string are letters.	str1=’University’ str1.isalpha()	True
isascii()	Returns True if all characters in the string are ASCII values	str1=’University’ str1.isascii()	True
isdecimal()	Returns True if all characters in the string are decimals. It supports only Decimal Numbers.	str1=’University’ str1.isdecimal()	False
isdigit()	Returns True if all characters in the string are digits. It supports Decimals, Subscripts, Superscripts.	str1=’University’ str1.isdigit()	False
isidentifier()	Returns TRUE if the string is a valid identifier according to the language definition, and FALSE	str1=’University’ str1.isidentifier()	True

	otherwise.		
islower()	Returns True if all characters in the string are lower case	str1='University' str1.islower()	False
isnumeric()	Returns True if all characters in the string are numeric. It supports Digits, Subscripts, Superscripts, Roman Numerals, Currency Numerators.	str1='University' str1.isnumeric()	False
isprintable()	Returns True if all characters in the string are printable. It returns False if the string contains at least one non-printable character. Line feed \n and tab \t are examples of nonprintable characters.	str1='University' str1.isprintable()	True
isspace()	Returns True if all characters in the string are whitespaces.	str1='University' str1.isspace()	False
istitle()	Returns TRUE if the string is nonempty and a titlecased string. Otherwise, it returns FALSE. In titlecased string each word starts with an uppercase character and the remaining characters are lowercase.	str1='University' str1.istitle()	True
isupper()	Returns True if all characters in the string are upper case.	str1='University' str1.isupper()	False
join()	Joins the elements of an iterable (list, string, tuple) to the end of the string.	str1='University' str1.join(('SGU','Kolhapur','Atigre'))	'SGUUniversityKolhapurUniversityAtigre'
ljust()	Returns left-justified string of	str1='University'	'University@@'

	length width. Padding is done using the specified fillchar (default is an ASCII space). The original string is returned as it is, if width is less than or equal to string length.	str1.ljust(12,'@')	
lower()	Converts a string into lower case.	str1=’University’ str1.lower()	'university'
lstrip()	Returns a left trim version of the string. Removes whitespace from the beginning (leading) of the string by default.	str3=' University' str3.lstrip()	'University'
maketrans()	Returns a translation table to be used in translations. It creates a one to one mapping of a character to its translation/replacement. It creates a Unicode representation of each character for translation. This translation mapping is then used for replacing a character to its mapped character when used in translate() method.	dict1 = {"a": "123", "b": "456", "c": "789"} str4 = "abc" print(str4.maketrans(dict1)) firstString = "abc" secondString = "def" str4 = "abc" print(str4.maketrans(firstString ,secondString))	{97: '123', 98: '456', 99: '789'} {97: 100, 98: 101, 99: 102}
partition()	Returns a tuple where the string is parted into three parts.	str1=’University’ str1.partition('ver')	('Uni', 'ver', 'sity')

	Splits the string at the first occurrence of the argument string and returns a tuple containing the part the before separator, argument string and the part after the separator.		
replace()	Returns a string where a specified value is replaced with a specified value.	str1=’University’ str1.replace('U','u')	'university'
rfind()	Searches the string for a specified value and returns the last position of where it was found.	str1=’University’ str1.rfind('i')	7
rindex()	Searches the string for a specified value and returns the last position of where it was found.	str1=’University’ str1.rindex('i')	7
rjust()	Returns a right justified version of the string.	str1=’University’ str1.rjust(12,'@')	'@@University'
rpartition()	Returns a tuple where the string is parted into three parts. Partitions where the last occurrence of separator is found.	str4='UniversityUniversity' str4.rpartition('ni')	('UniversityU', 'ni', 'versity')
rsplit()	Splits the string at the specified separator, and returns a list.	str1=’University’ str1.rsplit('ver')	['Uni', 'sity']
rstrip()	Returns a right trim version of the string. Removes whitespace from the end of the string by default.	str4='University ' str4.rstrip()	'University'

split()	Splits the string at the specified separator, and returns a list.	str1=’University’ str1.split('ver')	['Uni', 'sity']
splitlines()	Splits the string at line breaks and returns a list.	str4='University\nSGU\nAtigre' str4.splitlines()	['University', 'SGU', 'Atigre']
startswith()	Returns true if the string starts with the specified value.	str1=’University’ str1.startswith('N')	False
strip()	Returns a trimmed version of the string. (will remove spaces from left and right side)	str4=' Uni versity ' str4.strip()	'Uni versity'
swapcase()	Swaps cases, lower case becomes upper case and vice versa	str1=’University’ str1.swapcase()	'uNIVERSITY'
title()	Converts the first character of each word to upper case.	str4='university' str4.title()	'University'
translate()	Returns a translated string. Takes the translation table to replace/translate characters in the given string as per the mapping table. The translation table is created by the static method maketrans().	firstString = "abc" secondString = "def" str4 = "abc" str5=str4.maketrans(firstString, secondString) print(str5) str4.translate(str5)	{97: 100, 98: 101, 99: 102} 'def'
upper()	Converts a string into upper case.	str1=’University’ str1.upper()	'UNIVERSITY'
zfill()	Fills the string with a specified number of 0 values at the beginning.	str1=’University’ str1.zfill(12) str1.zfill(2)	'00University' 'University'