Ms.M.Kamala Malar
AP/IT

Files: A named storage area, can store data or program

Types of files :

Text file : can handle alphanumeric characters (Text)

Binary files : can handle audio, video , images and text

File operations:

1)Open

file object = open(file_name , access_mode)

file_name: That contains the name of the file that you want to access.

access_mode: The access_mode determines the mode in which the file has to be opened

Sno	Modes	Description
1	r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
2	rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
3	r+	Opens a file for both reading and writing. The file pointer is placed at the beginning of the file.
4	rb+	Opens a file for both reading and writing in binary format. The file pointer is placed at the beginning of the file.
5	w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
6	wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
7	w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
8	wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
9	a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
10	ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
11	a+	Opens a file for both appending and reading. The file pointer is at the end of the

		file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
12	ab+	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

2)The close() Method
The close() method of a file object flushes any unwritten information and closes the file object, after which no more writing can be done.
Syntax
fileObject.close();

3)**Read and Write**
The read() Method – reads the content from file
fileObject.read([count]);
The write() Method
The write() method writes any string to an open file.
fileObject.write(string]);

File attributes

file.closed()	# Returns true if file is closed, false otherwise.
file.mode()	# Returns access mode with which file was opened.
file.name()	# Returns name of the file.

File Methods

SNo.	**Methods**	**Description**
1	file.close()	Close the file.
2	file.flush()	Flush the internal buffer.
3	file.next()	Returns the next line from the file each time it is being called.
4	file.read([size])	Reads at most size bytes from the file
5	file.readline([size])	Reads one entire line from the file.
6	file.readlines([sizehint])	Reads until EOF using readline() and return a list containing the lines.
7	file.seek(offset[,whence])	Sets the file's current position

		file.seek(10,0) – 10 bytes from the beginning file.seek(10,1) – 10 bytes from the current position file.seek(10,2) – 10 bytes from the end of the file
8	file.tell()	Returns the file's current position
9	file.truncate([size])	Truncates the file's size. If the optional size argument is present, the file is truncated to (at most) that size.
10	file.write(str)	Writes a string to the file. There is no return value.
11	file.writelines(sequence)	Writes a sequence of strings to the file. The sequence can be list of strings.

Example Problem 1 : Count no of word in a file

```
fw=open(“sample.txt”,”w”)
str=input(“Enter the content to write in to the file “sample.txt”)
fw.write(str)
fw.close()
fr=open(“sample.txt”,”r”)
content=fr.read()
words=content.split()
wc=len(words)
print(“totoal no of words”,wc)
```

Example Problem 2 : Copy the content of one file (in.txt) to another (out.txt)

```
fw=open(“in.txt”,”w”)
str=input(“Enter the content to write in to the file “in.txt”)
fw.write(str)
fw.close()
with open("in.txt",”r”) as f:
   with open("out.txt", "w") as f1:
      for line in f:
         f1.write(line)
```

Format operator(%)

```
print "My name is %s and weight is %d kg!" % ('Zara', 21)
```

output: My name is Zara and weight is 21 kg!

Here is the list of complete set of symbols which can be used along with %:

Format Symbol

%c character
%s string
%i signed decimal integer
%d signed decimal integer

%u unsigned decimal integer
%o octal integer
%x hexadecimal integer (lowercase letters)
%X hexadecimal integer (UPPERcase letters)
%e exponential notation (with lowercase 'e')
%E exponential notation (with UPPERcase 'E')
%f floating point real number
%g the shorter of %f and %e
%G the shorter of %f and %E

Python Dictionary

Python's dictionaries are work like associative arrays consist of key-value pairs.
A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
Dictionaries have no concept of order among elements. They are simply unordered.

Accessing dictionary elements:

```
tinydict = {'name': 'john','code':6734, 'dept': 'sales'}
print( tinydict)                    # Prints complete dictionary
print( tinydict.keys())  # Prints all the keys
print( tinydict.values())           # Prints all the values

bigdict={‘rno’:[1,2,3,4],’name’:[‘raja’,’rani’,’john,’jeny’]}
print (bigdict[‘name’] #  Prints all the values under the key ‘name’)
```

Updating dictionaries:

```
bigdict[‘rno’]=[11,12,13,14]
```

Deleting dictionary elements:

```
del bigdict[‘rno’]       # delete all the rno
bigdict.clear()  # delete all the entries
del bigdict              # delete the entire dictionary
```

Dictionary Functions:

1 len(dict) # Gives the total length of the dictionary.
2 str(dict) # Convert dictionaries to string
43type(dict) # Returns a dictionary type.

Dictionary methods with description:

1 dict.clear() # Removes all elements of dictionary *dict*
2 dict.copy() # Returns a shallow copy of dictionary *dict*
3 dict.fromkeys() # Create a new dictionary with keys and values.
4 dict.get(key, default=None) # For *key* key, returns value or default if key not in dictionary
5 dict.has_key(key) # Returns *true* if key in dictionary *dict*, *false* otherwise
6 dict.items() # Returns a list of *dict*'s (key, value) tuple pairs
7 dict.keys() # Returns list of dictionary dict's keys
8 dict.setdefault(key, default=None)
# Similar to get(), but will set dict[key]=default if *key* is not in dict
9 dict.update(dict2) # Adds dictionary *dict2*'s key-values pairs to *dict*
10 dict.values() # Returns list of dictionary *dict*'s values

3.1 Python Program to Count the Occurrences of Each Word in a Given String Sentence

Problem statement

The program takes a text and counts the occurrence of each word in the given sentence.

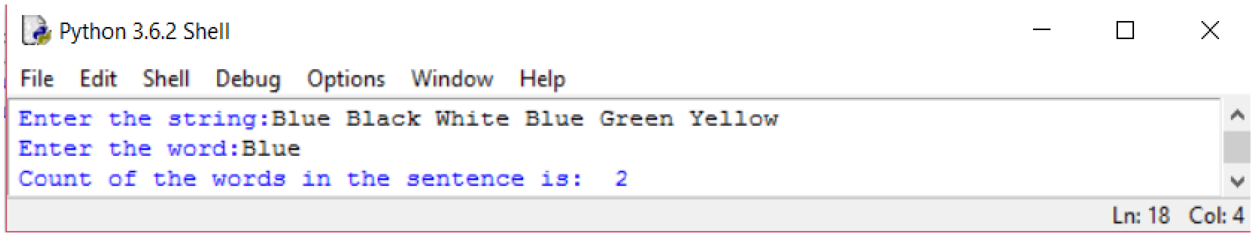
Algorithm

1. Read a string and a word from the user.
2. Initialize a count variable to 0.
3. Split the string using space as the reference and store the words in a list.
4. Use a for loop to traverse through the words in the list and use an if statement to check if the word in the list matches the word given by the user and increment the count.
5. Print the total count of the variable.
6. Exit.

Program/Source Code

```
string=input("Enter the string:")
word=input("Enter the word:")
a=[]
count=0
a=string.split(" ")
for i in range(0,len(a)):
    if(word==a[i]):
        count=count+1
print("Count of the words in the sentence is: ", count)
```

Sample Input / Output



```
Enter the string:Blue Black White Blue Green Yellow
Enter the word:Blue
Count of the words in the sentence is:  2
```

3.2 Python Program to Replace all Occurrences of ‘a’ with $ in a String

Problem statement

The program takes a string and replaces all occurrences of ‘t’ with ‘#’.

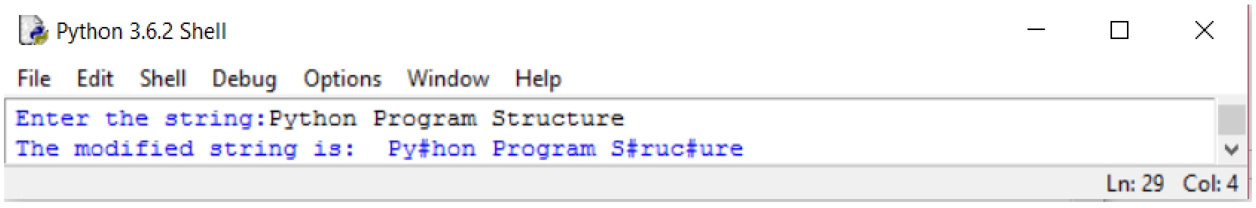
Algorithm

1. Read a string and store it in a variable.
2. Using the replace function, replace all occurrences of ‘t’ and ‘T’ with ‘#’ and store it back in the variable.
3. Print the modified string.
4. Exit.

Program/Source Code

```
string=input("Enter the string:")
string=string.replace('t','#')
string=string.replace('T','#')
print("The modified string is: ",string)
```

Sample Input / Output



```
Enter the string:Python Program Structure
The modified string is:  Py#hon Program S#ruc#ure
```

3.3 Python Program to Detect if Two Strings are Anagrams

Problem statement

The program takes two strings and checks if the two strings are anagrams.

Algorithm

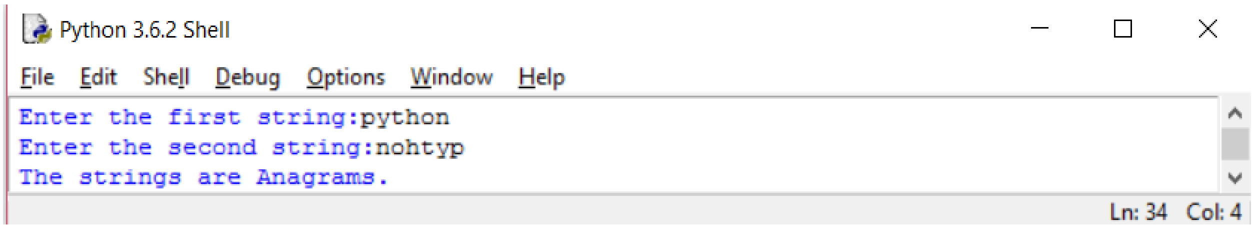
1. Read two strings from the user.
2. Then use sorted() to sort both the strings into lists.
3. Compare the sorted lists and check if they are equal.
4. Print the final result.
5. Exit.

Program/Source Code

```
string1=input("Enter the first string:")
string2=input("Enter the second string:")
if(sorted(string1)==sorted(string2)):
      print("The strings are Anagrams.")
```

```
else:
    print("The strings aren't Anagrams.")
```

Sample Input / Output



```
Enter the first string:python
Enter the second string:nohtyp
The strings are Anagrams.
```

3.4 Python Program to Form a New String where the First Character and the Last Character have been Exchanged

Problem statement

The program takes a string and swaps the first character and the last character of the string.

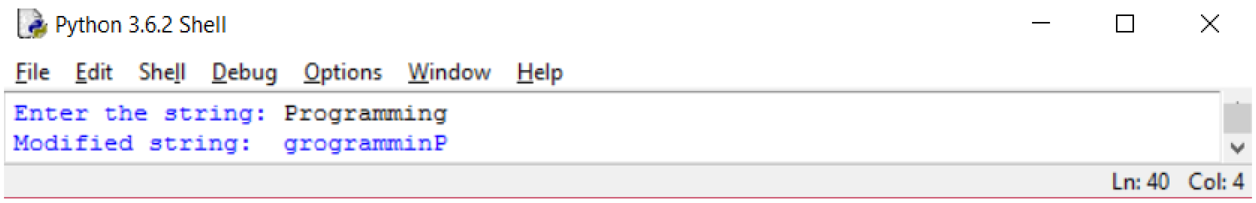
Algorithm

1. Read a string from the user.
2. Pass the string as an argument to a function.
3. In the function, split the string.
4. Then add the last character to the middle part of the string which is in turn added to the first character.
5. Print the modified string.
6. Exit.

Program/Source Code

```
string=input("Enter the string: ")
print("Modified string: ",string[-1:] + string[1:-1] + string[:1])
```

Sample Input / Output



```
Enter the string: Programming
Modified string:  grogramminP
```

3.5 Python Program to Remove the nth Index Character from a Non-Empty String

Problem statement

The program takes a string and removes the nth index character from the non-empty string.

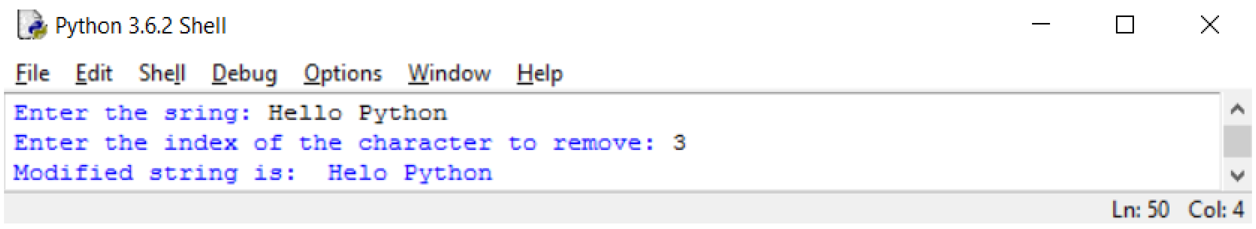
Algorithm

1. Read a string from the user.
2. Read the index of the character to remove.
3. Pass the string and the index as arguments to a function named remove.
4. In the function, the string should then be split into two halves before the index character and after the index character.
5. These two halves should then be merged together.
6. Print the modified string.
7. Exit.

Program/Source Code

```
string=input("Enter the sring: ")
n=int(input("Enter the index of the character to remove: "))
first = string[:n]
last = string[n+1:]
print("Modified string is: ", first+last)
```

Sample Input / Output



```
Enter the sring: Hello Python
Enter the index of the character to remove: 3
Modified string is:  Helo Python
```

3.6 Python Program to Count the Number of Vowels in a String

Problem statement

The program takes a string and counts the number of vowels in a string.

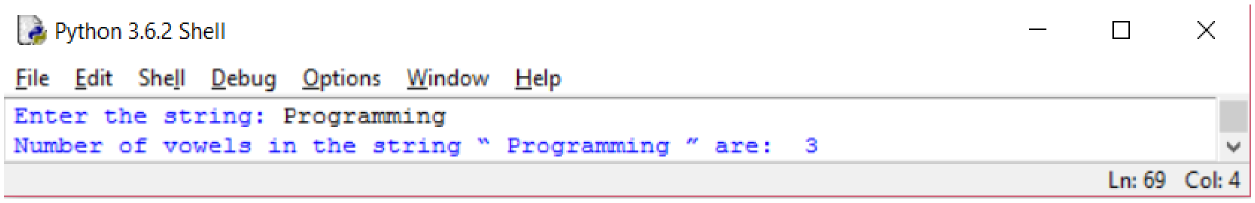
Algorithm

1. Read a string from the user.
2. Initialize a count variable to 0.
3. Use a for loop to traverse through the characters in the string.
4. Use an if statement to check if the character is a vowel or not and increment the count variable if it is a vowel.
5. Print the total number of vowels in the string.
6. Exit.

Program/Source Code

```
string=input("Enter the string: ")
vowels=0
for x in string:
    if(x=='a' or x=='e' or x=='i' or x=='o' or x=='u' or x=='A' or x=='E' or x=='I' or x=='O'
                                                                              or x=='U'):
        vowels=vowels+1
print("Number of vowels in the string “",string,"” are: ", vowels)
```

Sample Input / Output



```
Enter the string: Programming
Number of vowels in the string “ Programming ” are:   3
```

3.7 Python Program to Take in a String and Replace Every Blank Space with Hyphen

Problem statement

The program takes a string and replaces every blank space with a hyphen.

Algorithm

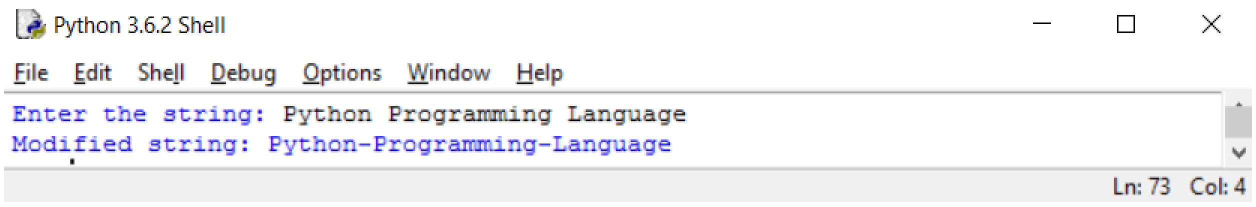
1. Take a string and store it in a variable.
2. Using the replace function, replace all occurrences of ‘ ‘ with ‘-‘ and store it back in the variable.
3. Print the modified string.
4. Exit.

Program/Source Code

```
string=input("Enter the string: ")
string=string.replace(' ','-')
```

```
print("Modified string:", string)
```

Sample Input / Output



```
Enter the string: Python Programming Language
Modified string: Python-Programming-Language
```

3.8 Write a python program to compare two strings

Problem Statement

The program takes two strings and displays the comparison result

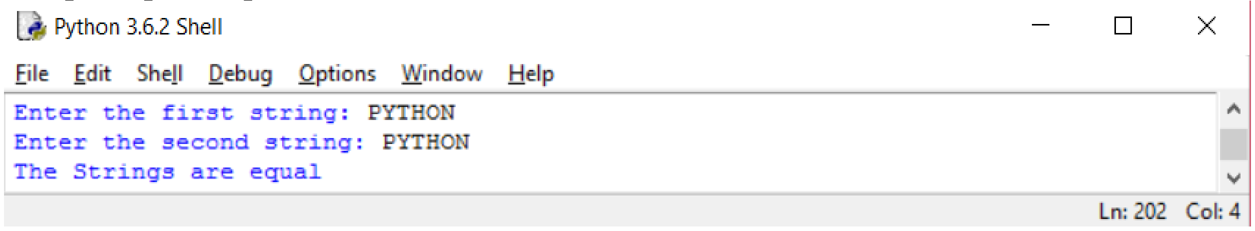
Algorithm

1. Read two string from the user
2. Compare two strings
3. If the comparison result is 0, the strings are equal
4. If the comparison result is greater than 0, First string is larger than second string
5. If the comparison result is lesser than 0, First string is smaller than second string
6. Exit

Program/Source Code

```
String1=input("Enter the first string: ")
String2=input("Enter the second string: ")
if (String1==String2):
    print("The Strings are equal")
elif (String1>String2):
    print("First string is larger")
else:
    print("Second string is larger")
```

Sample Input/output



```
Enter the first string: PYTHON
Enter the second string: PYTHON
The Strings are equal
```

3.9 Write a python program to count the common characters in the two inputted strings.

Problem Statement

The program reads two strings and checks common characters in both the strings

Algorithm

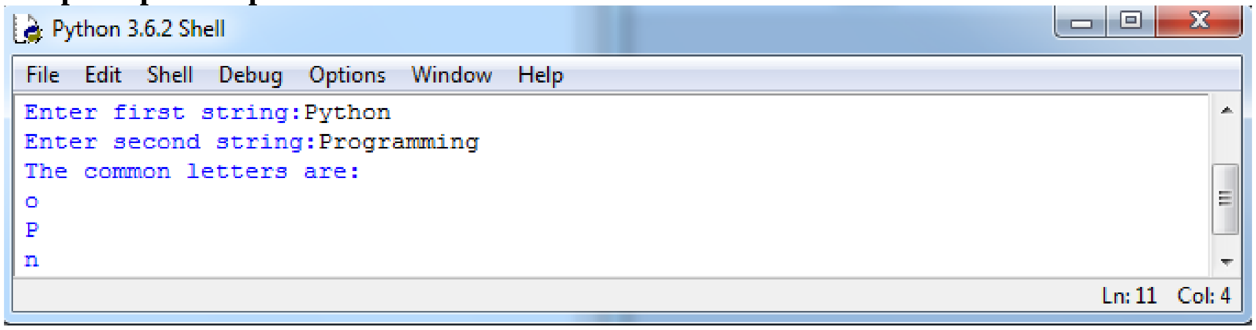
1. Enter two input strings and store it in separate variables.

2. Convert both of the strings into sets and find the common letters between both the sets.
3. Store the common letters in a list.
4. Use a for loop to print the letters of the list.
5. Exit.

Program/Source Code:

```
s1=input("Enter first string:")
s2=input("Enter second string:")
x=list(set(s1)&set(s2))
print("The common letters are:")
for i in x:
    print(i)
```

Sample Input/Output



```
Enter first string:Python
Enter second string:Programming
The common letters are:
o
P
n
```

3.10 Write a python program to concatenate two strings

Problem Statement

The Program reads two string and concatenates them

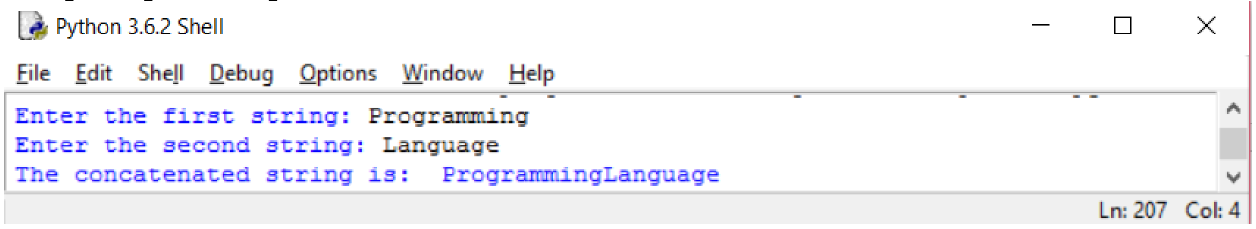
Algorithm

1. Read two string from the user
2. Concatenate the two string
3. Display the concatenated String
4. Exit

Program/Source Code

```
String1=input("Enter the first string: ")
String2=input("Enter the second string: ")
print("The concatenated string is:", String1+String2)
```

Sample Input / Output



```
Enter the first string: Programming
Enter the second string: Language
The concatenated string is:  ProgrammingLanguage
```

3.11 Write a python program to reverse the string

Problem Statement

The Program reads a string and reverse it

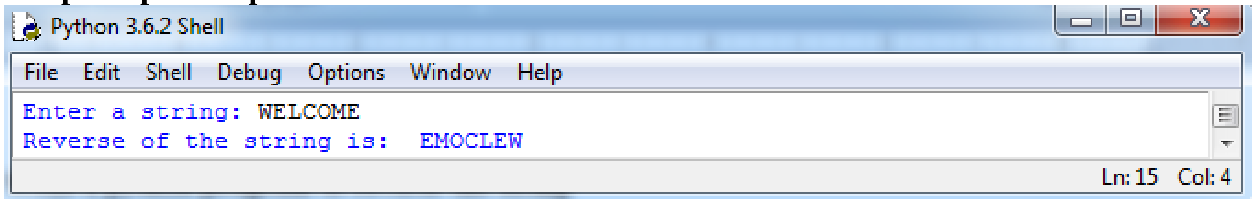
Algorithm

1. Read a string from the user
2. Use string slicing to reverse the string.
3. Print the reversed string.
4. Exit.

Program/Source Code

```
s1=str(input("Enter a string: "))
print("Reverse of the string is: ",s1[::-1])
```

Sample Input/Output



```
Enter a string: WELCOME
Reverse of the string is:  EMOCLEW
```

3.12 Write a python program to copy string1 to string2

Problem Statement

The Program reads a string and copy it to the other string

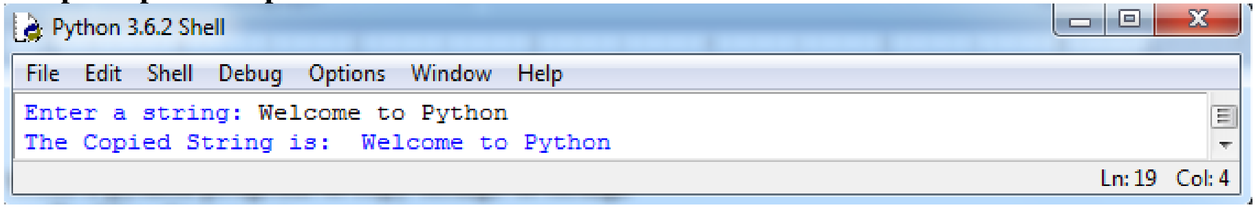
Algorithm

1. Read a string from the user
2. Assign one string to another.
3. Print the copied string.
4. Exit.

Program/Source Code

```
s1=str(input("Enter a string: "))
s2=s1
print("The Copied String is: ", s2)
```

Sample Input / Output



```
Enter a string: Welcome to Python
The Copied String is:  Welcome to Python
```

3.13 Python Program to calculate the length of a string without using a library function

Problem statement

The program reads a string and calculates the length of the string without using library functions.

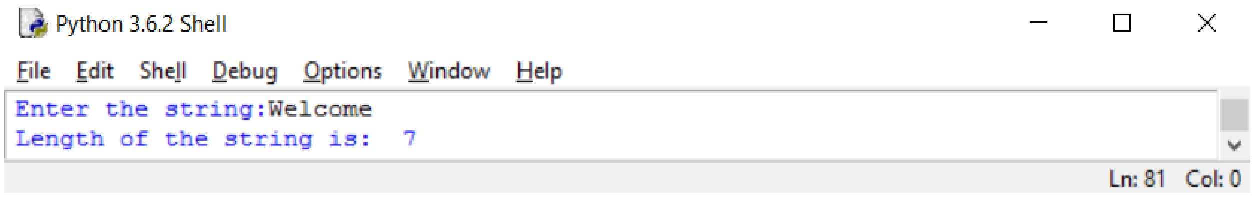
Algorithm

1. Read a string from the user.
2. Initialize a count variable to 0.
3. Use a for loop to traverse through the characters in the string and increment the count variable each time.
4. Print the total count of the variable.
5. Exit.

Program/Source Code

```
string=input("Enter the string:")
count=0
for i in string:
      count=count+1
print("Length of the string is: ", count)
```

Sample Input / Output



```
Enter the string:Welcome
Length of the string is:  7
```

3.14 Python Program to Remove the Characters of Odd Index Values in a String

Problem statement

The program reads a string and removes the characters of odd index values in the string.

Algorithm

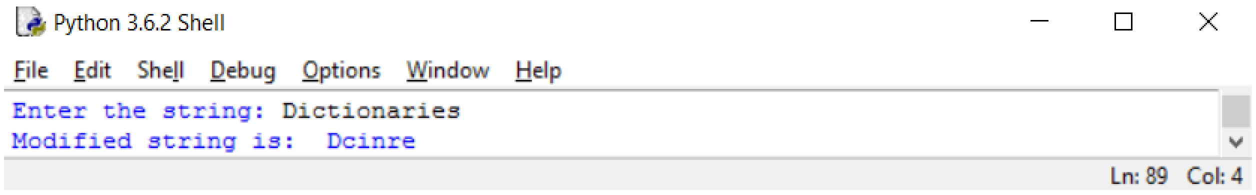
1. Read a string from the user.
2. Pass the string as an argument to a function.
3. In the function, initialize a variable to an empty character.
4. Use a for loop to traverse through the string.
5. Use an if statement to check if the index of the string is odd or even.

6. If the index is odd, append the no character to the string.
7. Then print the modified string.
8. Exit.

Program/Source Code

```
string=input("Enter the string: ")
final = ""
for i in range(len(string)):
    if i % 2 == 0:
        final = final + string[i]
print("Modified string is: ", final)
```

Sample Input / Output



```
Enter the string: Dictionaries
Modified string is:  Dcinre
```

3.15 Python Program to Calculate the Number of Words and the Number of Characters Present in a String

Problem statement

The program reads a string and calculates the number of words and characters present in the string.

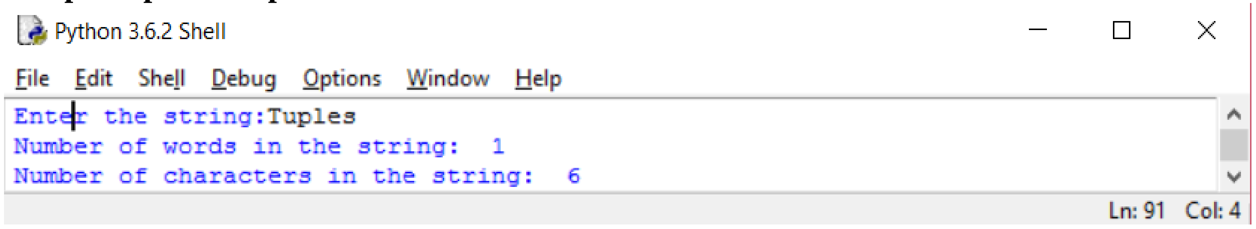
Algorithm

1. Read a string from the user and store it in a variable.
2. Initialize the character count variable to 0 and the word count variable to 1.
3. Use a for loop to traverse through the characters in the string and increment the character count variable each time.
4. Increment the word count variable only if a space is encountered.
5. Print the total count of the characters and words in the string.
6. Exit.

Program/Source Code

```
string=input("Enter the string:")
char=0
word=1
for i in string:
    char=char+1
    if(i==' '):
        word=word+1
print("Number of words in the string: ", word)
print("Number of characters in the string: ",char)
```

Sample Input / Output



```
Enter the string:Tuples
Number of words in the string:  1
Number of characters in the string:  6
```

3.16 Python Program to Take in Two Strings and Display the Larger String without Using Built-in Functions

Problem statement

The program reads in two strings and displays the larger string without using built-in function.

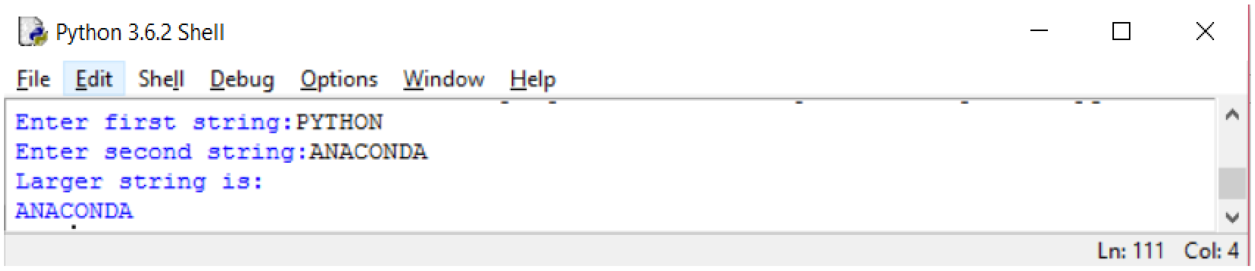
Algorithm

1. Read in two strings from the user.
2. Initialize the two count variables to zero.
3. Use a for loop to traverse through the characters in the string and increment the count variables each time a character is encountered.
4. Compare the count variables of both the strings.
5. Print the larger string.
6. Exit.

Program/Source Code

```
string1=input("Enter first string:")
string2=input("Enter second string:")
count1=0
count2=0
for i in string1:
    count1=count1+1
for j in string2:
    count2=count2+1
if(count1<count2):
    print("Larger string is:",string2)
elif(count1==count2):
    print("Both strings are equal.")
else:
    print("Larger string is:", string1)
```

Sample Input / Output



```
Enter first string:PYTHON
Enter second string:ANACONDA
Larger string is:
ANACONDA
```

3.17 Python Program to Count Number of Lowercase Characters in a String

Problem statement

The program reads a string and counts number of lowercase characters in a string.

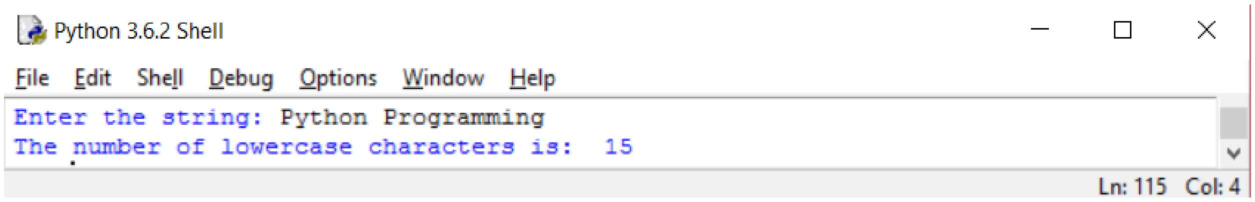
Algorithm

1. Read a string from the user.
2. Initialize a count variable to 0.
3. Use a for loop to traverse through the characters in the string and increment the count variable each time a lowercase character is encountered.
4. Print the total count of the variable.
5. Exit.

Program/Source Code

```
string=input("Enter the string:")
count=0
for i in string:
    if(i.islower()):
        count=count+1
print("The number of lowercase characters is: ",count)
```

Sample Input / Output



```
Enter the string: Python Programming
The number of lowercase characters is:  15
```

3.18 Python Program to Check if a String is a Palindrome or Not

Problem statement

The program reads a string and checks if a string is a palindrome or not.

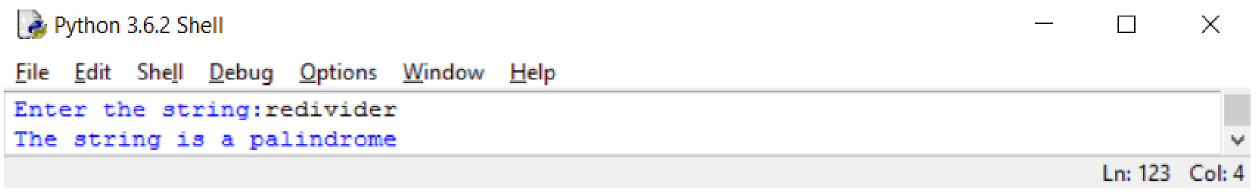
Algorithm

1. Read a string from the user.
2. Reverse the string using string slicing and compare it back to the original string.
3. Print the final result
4. Exit.

Program/Source Code

```
string=input("Enter the string:")
if(string==string[::-1]):
    print("The string is a palindrome")
else:
    print("The string isn't a palindrome")
```

Sample Input / Output



```
Enter the string:redivider
The string is a palindrome
```

3.19 Write a python program to convert strings in lower case to upper case and vice-versa

Problem statement

The program reads a string and convert it to all upper case letters and vice-versa.

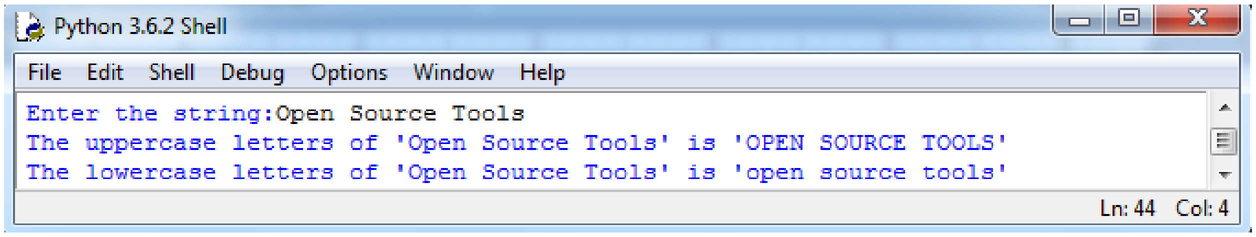
Algorithm

1. Read a string from the user.
2. Convert all cases to upper case letter using built-in string function *upper* and print it.
3. Convert all cases to lower case letter using built-in string function *lower* and print it.
4. Exit.

Program/Source Code

```
string=input("Enter the string:")
print("The uppercase letters of '{0}' is '{1}'".format(string,string.upper()))
print("The lowercase letters of '{0}' is '{1}'".format(string,string.lower()))
```

Sample Input / Output



```
Enter the string:Open Source Tools
The uppercase letters of 'Open Source Tools' is 'OPEN SOURCE TOOLS'
The lowercase letters of 'Open Source Tools' is 'open source tools'
```

3.20 Write a python program to swap cases in the given string

Problem statement

The program reads a string and convert all its cases.

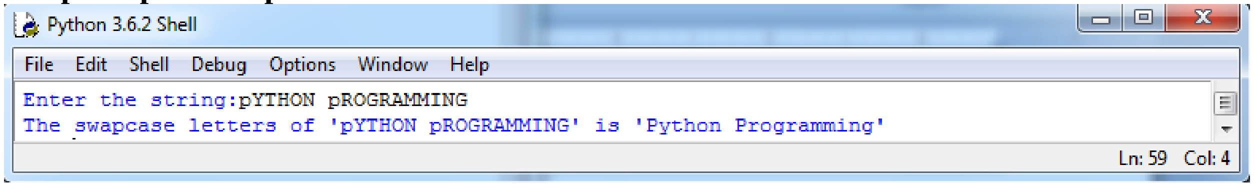
Algorithm

1. Read a string from the user.
2. Convert all it cases to new cases using built-in string function *swapcas()*and print it.
3. Exit.

Program/Source Code

```
string=input("Enter the string:")
print("The swapcase letters of '{0}' is '{1}'".format(string,string.swapcase()))
```

Sample Input / Output



```
Enter the string:pYTHON pROGRAMMING
The swapcase letters of 'pYTHON pROGRAMMING' is 'Python Programming'
```

3.21 Write a python program to demonstrate string function for joining, finding length and replacing the string

Problem statement

The program reads two strings and demonstrate string function for joining, finding length and replacing the string.

Algorithm

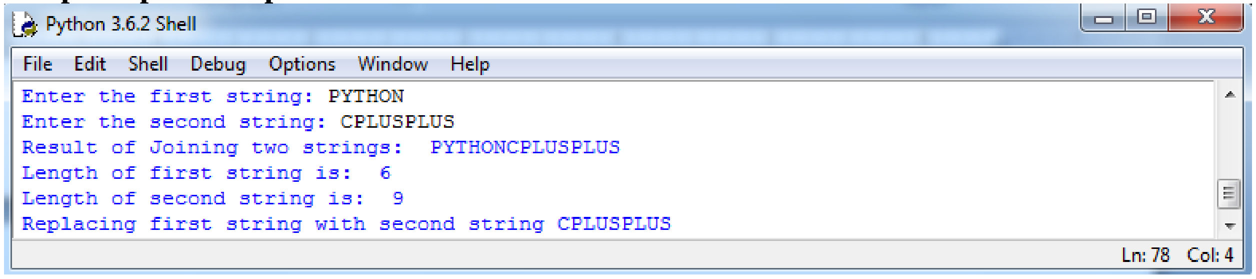
1. Read a string from the user.
2. Join two strings using '+' operator and print it.
3. Find the length of two string using the built-in *len*() and print it.
4. Replace the string using the built-in *replace*() and print it
5. Exit.

Program/Source Code

```
string1=input("Enter the first string: ")
string2=input("Enter the second string: ")
```

```
print("Result of Joining two strings: ",string1+string2)
print("Length of first string is: ", len(string1))
print("Length of second string is: ", len(string2))
print("Replacing first string with second string",string1.replace(string1,string2))
```

Sample Input / Output



```
Enter the first string: PYTHON
Enter the second string: CPLUSPLUS
Result of Joining two strings:  PYTHONCPLUSPLUS
Length of first string is:  6
Length of second string is:  9
Replacing first string with second string CPLUSPLUS
```

3.22 Python Program to Calculate the Number of Upper Case Letters and Lower Case Letters in a String

Problem statement

The program takes a string and counts the number of lowercase letters and uppercase letters in the string.

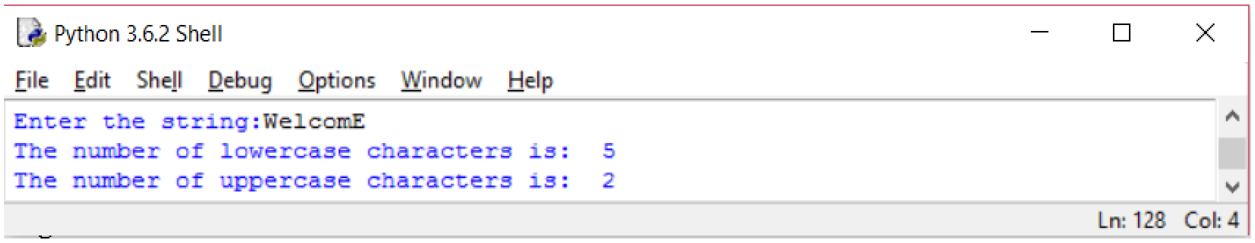
Algorithm

1. Read a string from the user and store it in a variable.
2. Initialize the two count variables to 0.
3. Use a for loop to traverse through the characters in the string and increment the first count variable each time a lowercase character is encountered and increment the second count variable each time a uppercase character is encountered.
4. Print the total count of both the variables.
5. Exit.

Program/Source Code

```
string=input("Enter the string:")
count1=0
count2=0
for i in string:
    if(i.islower()):
        count1=count1+1
    elif(i.isupper()):
        count2=count2+1
print("The number of lowercase characters is: ", count1)
print("The number of uppercase characters is: ", count2)
```

Sample Input / Output



```
Enter the string:WelcomE
The number of lowercase characters is:  5
The number of uppercase characters is:  2
```

3.23 Write a python program to check whether the characters of an Inputted String in Alphabetical order

Problem statement

The program reads a string and check whether the characters of an Inputted String in Alphabetical order.

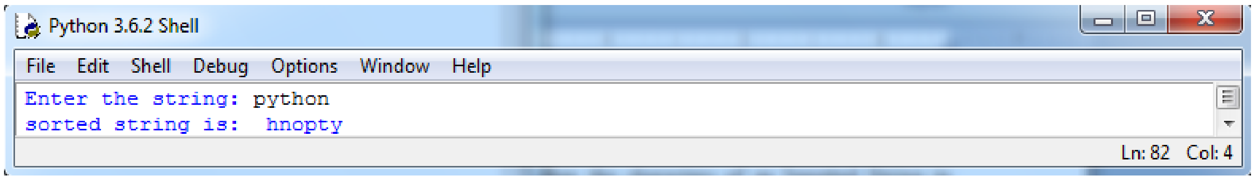
Algorithm

1. Read a string from the user and store it in a variable.
2. Use join() and sorted() functions in python to print the string in alphabetical order
3. Exit.

Program/Source Code

```
string=input("Enter the string: ")
print("sorted string is: ",''.join(sorted(string)))
```

Sample Input / Output



```
Enter the string: python
sorted string is:  hnopty
```

3.24 Write a python program to check if a string is a Pangram or Not

Problem statement

The program takes a string and checks if it is a pangram or not.

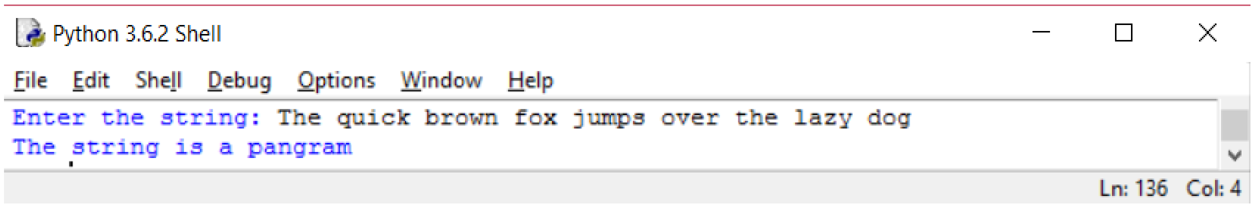
Algorithm

1. Read a string from the user and store it in a variable.
2. Pass the string as an argument to a function.
3. In the function, form two sets- one of all lower case letters and one of the letters in the string.
4. Subtract these both sets and check if it is equal to an empty set.
5. Print the final result.
6. Exit.

Program/Source Code

```
from string import ascii_lowercase as asc_lower
def check(s):
   return set(asc_lower) - set(s.lower()) == set([])
strng=input("Enter the string: ")
if(check(strng)==True):
     print("The string is a pangram")
else:
     print("The string isn't a pangram")
```

Sample Input / Output



```
Enter the string: The quick brown fox jumps over the lazy dog
The string is a pangram
```

3.25 Write a python program to accept a hyphen separated sequence of words as input and print the words in a hyphen-separated sequence after sorting them alphabetically

Problem statement

The program accepts a hyphen separated sequence of words as input and print the words in a hyphen-separated sequence after sorting them alphabetically.

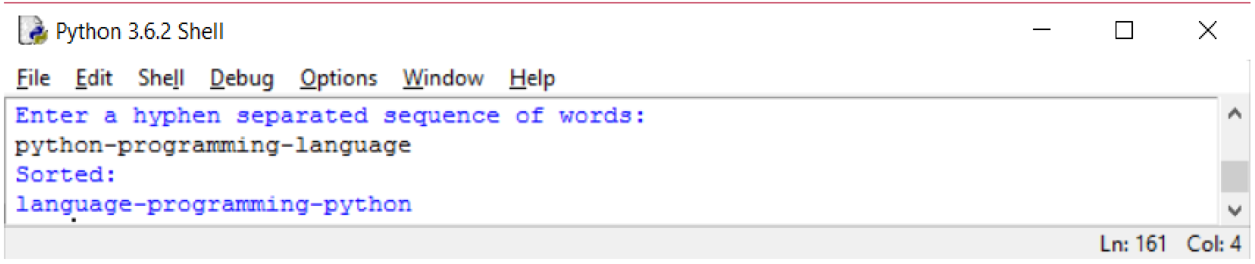
Algorithm

1. Read a hyphen separated sequence of words from the user.
2. Split the words in the input with hyphen as reference and store the words in a list.
3. Sort the words in the list.
4. Join the words in the list with hyphen between them and print the sorted sequence.
5. Exit.

Program/Source Code

```
print("Enter a hyphen separated sequence of words:")
lst=[n for n in input().split('-')]
lst.sort()
print("Sorted:")
print('-'.join(lst))
```

Sample Input / Output



```
Enter a hyphen separated sequence of words:
python-programming-language
Sorted:
language-programming-python
```

3.26 Write a python program to calculate the number of digits and letters in a string

Problem statement

The program takes a string and calculates the number of digits and letters in a string.

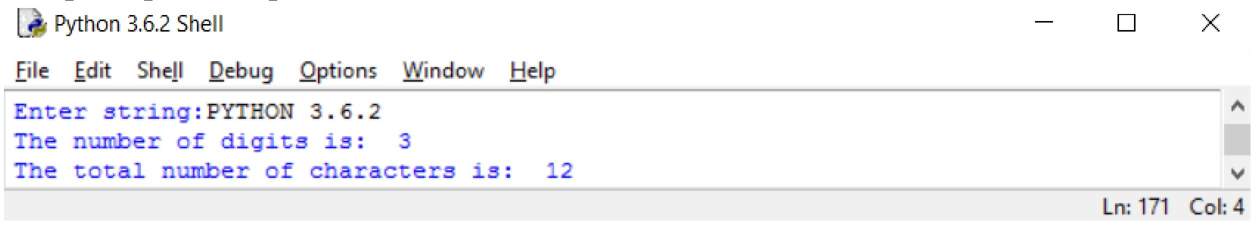
Algorithm

1. Read a string from the user and store it in a variable.
2. Initialize the two count variables to 0.
3. Use a for loop to traverse through the characters in the string and increment the first count variable each time a digit is encountered and increment the second count variable each time a character is encountered.
4. Print the total count of both the variables.
5. Exit.

Program/Source Code

```
string=input("Enter string:")
count1=0
count2=0
for i in string:
    if(i.isdigit()):
        count1=count1+1
    count2=count2+1
print("The number of digits is: ", count1)
print("The total number of characters is: ", count2)
```

Sample Input / Output



```
Enter string:PYTHON 3.6.2
The number of digits is:  3
The total number of characters is:  12
```

3.27 Write a python program to form a new string made of the first 2 and last 2 characters from a given string

Problem statement

The program takes a string and forms a new string made of the first 2 characters and last 2 characters from a given string.

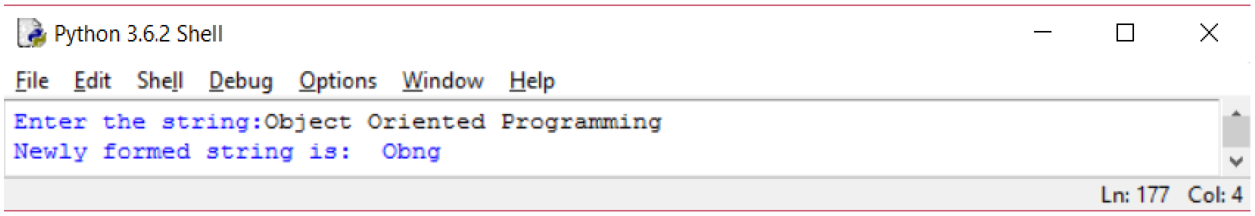
Algorithm

1. Read a string from the user and store it in a variable.
2. Initialize a count variable to 0.
3. Use a for loop to traverse through the characters in the string and increment the count variable each time.
4. Form the new string using string slicing.
5. Print the newly formed string.
6. Exit.

Program/Source Code

```
string=input("Enter the string:")
count=0
for i in string:
    count=count+1
newstr=string[0:2]+string[count-2:count]
print("Newly formed string is: ", newstr)
```

Sample Input / Output



```
Enter the string:Object Oriented Programming
Newly formed string is:  Obng
```

3.28 Write a python program to check if a substring is present in a given string

Problem statement

The program takes a string and checks if a substring is present in the given string.

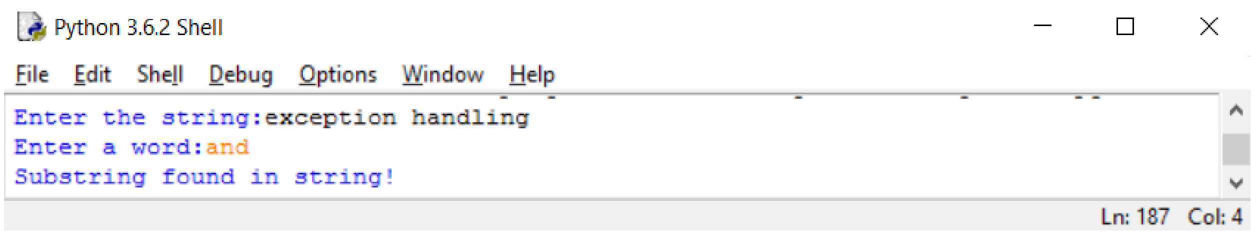
Algorithm

1. Read a string and a substring from the user and store it in separate variables.
2. Check if the substring is present in the string using find() in-built function.
3. Print the final result.
4. Exit.

Program/Source Code

```
string=input("Enter the string:")
sub_str=input("Enter a word:")
if(string.find(sub_str)==-1):
    print("Substring not found in string!")
else:
    print("Substring found in string!")
```

Sample Input / Output



```
Enter the string:exception handling
Enter a word:and
Substring found in string!
```

3.29 Write a python program to demonstrate string functions for splitting string and removing white spaces

Problem statement

The program reads a string demonstrate string functions for splitting string and removing white spaces.

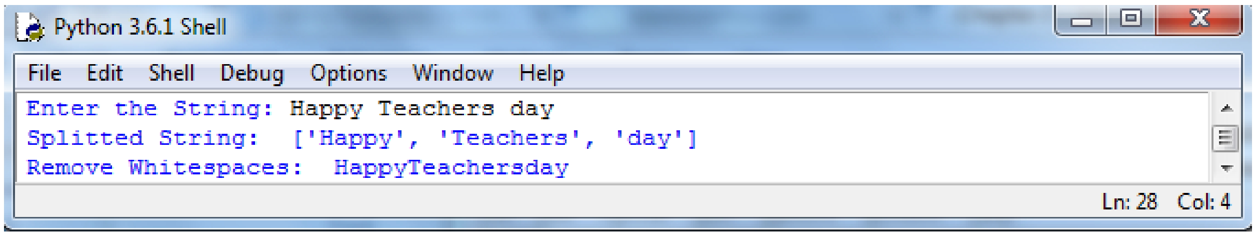
Algorithm

1. Read a string from the user.
2. Use string built-in *split*() to split the given string based on whitespaces .
3. Remove the whitespaces from the given string using the string built-in replace().
4. Exit.

Program/Source Code

```
string=input("Enter the String: ")
print("Splitted String: ",string.split())
print("Remove Whitespaces: ",string.replace(' ',''))
```

Sample Input / Output



```
Enter the String: Happy Teachers day
Splitted String:  ['Happy', 'Teachers', 'day']
Remove Whitespaces:  HappyTeachersday
```

3.30 Write a python program to remove consecutively repeated word in a sentence

Problem statement

The program reads a sentence and removes consecutively repeated word in a sentence.

Algorithm

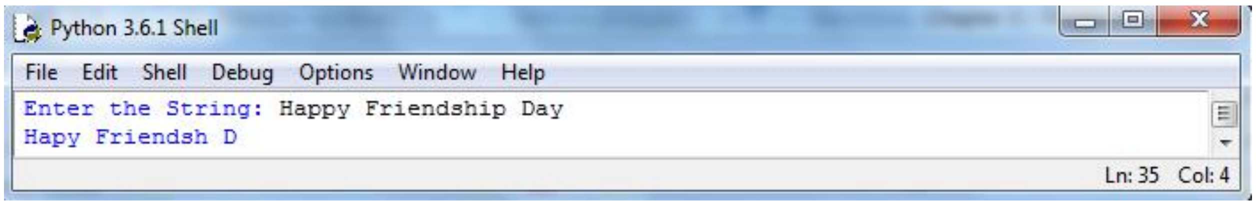
1. Read a sentence from the user.
2. Remove consecutively repeated word in a sentence using list and set
3. Print the resultant sentence after the removal of repeated words.
4. Exit.

Program/Source Code

```
def removeDuplicates(string,ignoreSpaces=True):
    result = []
    seen = set()
    for char in string:
        if char not in seen:
            seen.add(char)
            result.append(char)
        elif char==' ' and ignoreSpaces:
            result.append(char)
    return ''.join(result)

string = input("Enter the String: ")
print ("String after removing duplicates:", removeDuplicates(string))
```

Sample Input / Output



```
Enter the String: Happy Friendship Day
Hapy Friendsh D
```

3.31 Write a python program to return a quoted word from the sentence

Problem statement

The program reads a sentence and returns the quoted word from the sentence.

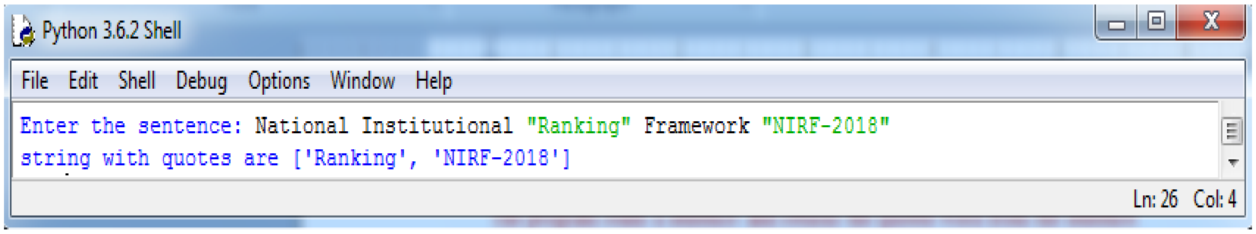
Algorithm

1. Read a sentence from the user.
2. Import the regular expression module, *re*
3. Use *findall*() method from the *re* module to find all quoted word from the sentence
4. Exit.

Program/Source Code

```
import re
inputString = input("Enter the sentence: ")
print("string with quotes are",re.findall(r'"([^"]*)"', inputString))
```

Sample Input / Output



```
Enter the sentence: National Institutional "Ranking" Framework "NIRF-2018"
string with quotes are ['Ranking', 'NIRF-2018']
```

3.32 Write a python program to capitalize each word in a given sentence

Problem statement

The program reads a sentence and capitalizes each word in a given sentence.

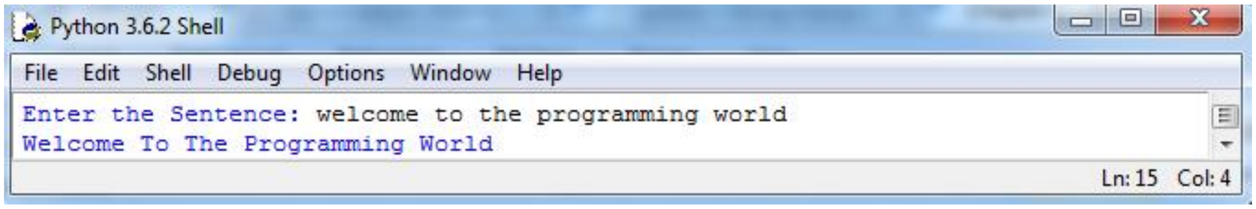
Algorithm

1. Read a string from the user.
2. Capitalize each word in the sentence using the built-in *title*()
3. Print the resultant sentence.
4. Exit.

Program/Source Code

```
string=input("Enter the Sentence: ")
print (string.title())
```

Sample Input / Output



```
Enter the Sentence: welcome to the programming world
Welcome To The Programming World
```

3.33 Write a Python program to return words beginning with specified letter in a sentence. Also display the count.

Problem statement

The program reads a sentence and return words beginning with specified letter in a sentence.

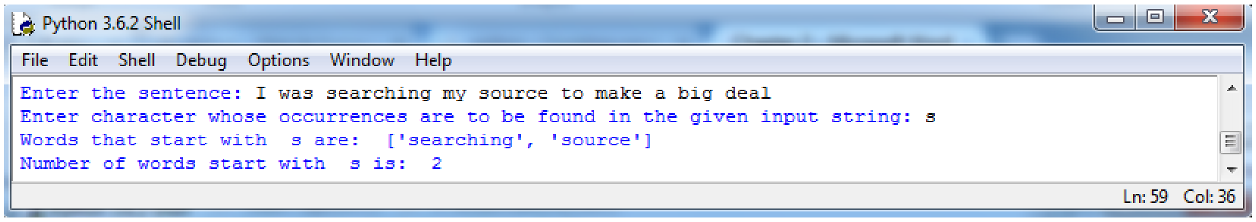
Algorithm

1. Read a sentence from the user.
2. Read the character whose occurrences are to be found in the given input string
3. Use *split*() method and *startswith*() method to return words beginning with specified letter in a sentence
4. Update the resultant of the step3 in a list
5. Display the items in the list and also print the count of elements in the list.
6. Exit

Program/Source Code

```
text=input("Enter the sentence: ")
c = input("Enter character whose occurrences are to be found in the given input string: ")
l=[t for t in text.split() if t.startswith(c)]
print("Words that start with ",c,"are: ",l)
print("Number of words start with ",c, "is: ", len(l))
```

Sample Input / Output



```
Enter the sentence: I was searching my source to make a big deal
Enter character whose occurrences are to be found in the given input string: s
Words that start with  s are:  ['searching', 'source']
Number of words start with  s is:  2
```

3.34 Write a python program to insert extra spaces before and after the string

Problem statement

The program reads a string and inserts extra spaces before and after the string.

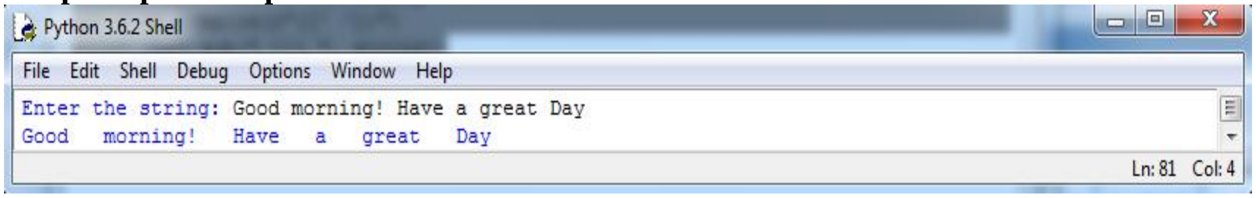
Algorithm

1. Read a string from the user.
2. Import the regular expression module, *re*
3. Use *compiler()* method from the *re* module to insert extra spaces before and after the string
4. Exit.

Program/Source Code

```
import re
string=input("Enter the string: ")
pat = re.compile(r"([' '])")
print (pat.sub(" \\1 ", string))
```

Sample Input / Output



```
Enter the string: Good morning! Have a great Day
Good   morning!   Have   a   great   Day
```

3.35 Write a Python program to accept password as a string and check if it is acceptable

Problem statement

The program to accept password as a string and check if it is acceptable.

Validation Rules:

- *At least 1 letter between [a-z] and 1 letter between [A-Z].*
- *At least 1 number between [0-9].*
- *At least 1 character from [$#@].*
- *Minimum length 6 characters.*
- *Maximum length 16 characters.*

Algorithm

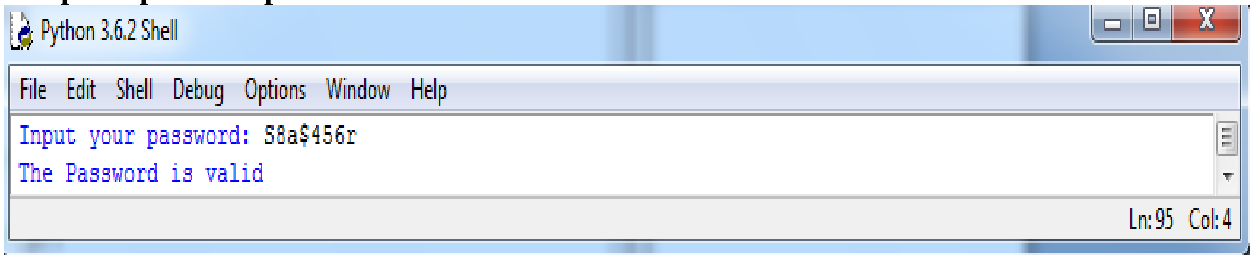
1. Read a string from the user.
2. Check the validity of the string whether it satisfies all the validation rules listed above
3. If it is valid string, then display “The password is valid”
4. Otherwise, display “The password is invalid”
5. Exit.

Program/Source Code

```
import re
p= input("Input your password: ")
x = True
```

```
while x:
    if (len(p)<6 or len(p)>12):
        break
    elif not re.search("[a-z]",p):
        break
    elif not re.search("[0-9]",p):
        break
    elif not re.search("[A-Z]",p):
        break
    elif not re.search("[$#@]",p):
        break
    elif re.search("\s",p):
        break
    else:
        print("The Password is valid")
        x=False
        break
if x:
    print("The Password is Invalid")
```

Sample Input / Output



```
Input your password: S8a$456r
The Password is valid
```

Write a python program to capitalize first and last word in a given sentence

Problem statement

The program reads a sentence and capitalizes first and last word in a given sentence.

Algorithm

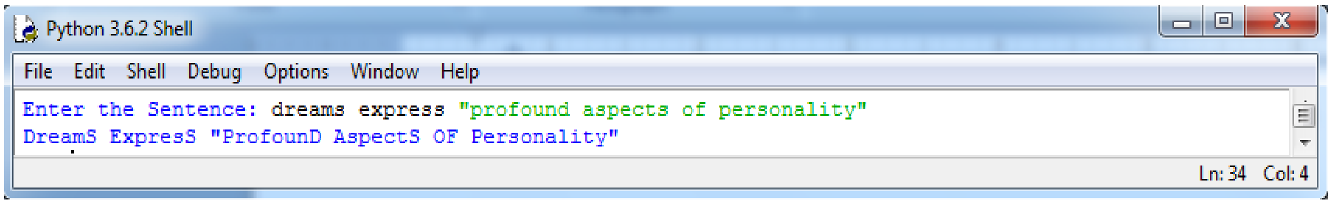
1. Read a string from the user.
2. Capitalize each first word in the sentence using the built-in *title*() method.
3. Identify the last word in the sentence and change its case to uppercase.
4. Print the resultant sentence.
5. Exit.

Program/Source Code

```
def capitalize(s):
     s, result = s.title(), ""
     for word in s.split():
            result += word[:-1] + word[-1].upper() + " "
        return result[:-1]      #To remove the last trailing space.

string=input("Enter the Sentence: ")
print (capitalize(string))
```

Sample Input / Output



```
Enter the Sentence: dreams express "profound aspects of personality"
DreamS ExpresS "ProfounD AspectS OF Personality"
```