Python Strings

- Find length of string
- Accessing individual characters/slicing
- Difference between string and list.
- Convert to lowercase / uppercase / titlecase / swapcase
- Verify if the strings consists of alphabets/digits/upper/lower
- Split string
- Find substring

String

- A string is a sequence of characters.
- We can create string by enclosing characters in quotes.
- Python treats single quotes the same as double quotes.
- Python uses Unicode format to represent characters.

```
name = 'Mohammed Sikander'

organization = "CRANES SOFTWARE INTERNATIONAL LTD"

print(name)
print(organization)
```

Reading and Printing String

- Reading and Printing a string

```
print('Enter your name ')

name = input()

print('Hi ' , name , 'Lets learn about Strings in python')
```

Accessing Individual characters

- We can access individual characters of string using indexing.

```
name = 'Mohammed Sikander'


print('First character is ' , name[0])
print('Last character is ' , name[-1])
print('Fourth character is ' , name[3])
```

- If we try to access index out of the range or use decimal number, we will get errors.

- Since there is no separate “character” type, indexing a string produces strings of length 1

```
name = "SIKANDER"

print(name, type(name) )

print(name[0], type(name[0]))
```

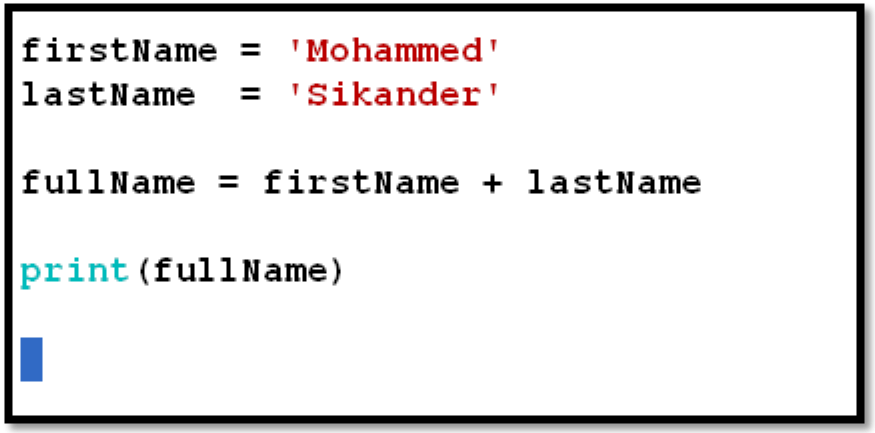
Slicing with Strings

- We can access substrings using slicing

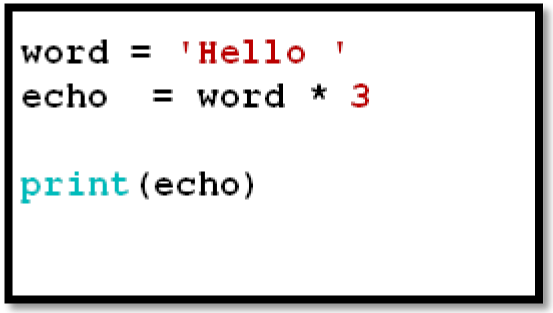
```
name = 'Mohammed Sikander'


print('First 5 characters are : ' , name[:5])
print('String excluding first 5 characters : ' , name[5:])
print('3rd to 8th character : ' , name[3:8])
```

String Concatenation

- Joining of two or more strings into a single one is called concatenation.
- The + operator does this in Python.
- The * operator can be used to repeat the string for a given number of times.



```
firstName = 'Mohammed'
lastName  = 'Sikander'

fullName = firstName + lastName

print(fullName)
```



```
word = 'Hello '
echo   = word * 3

print(echo)
```

Iterating Through String

- Using for loop we can iterate through a string.

```
word = 'CRANES'

for c in word:
        print(c)
```

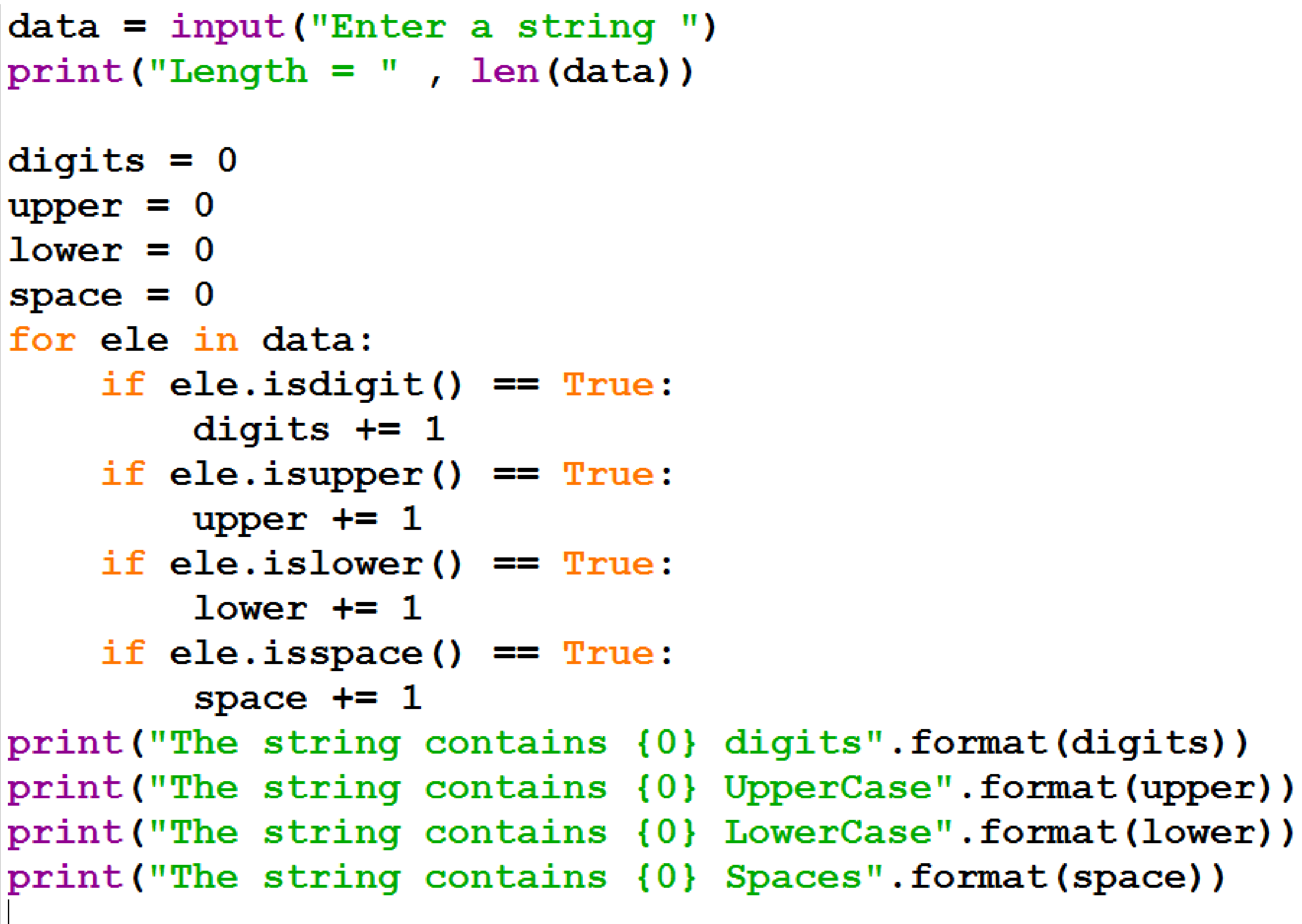
```
word = 'CRANES VARSITY'

count = 0
for c in word:
        if c == 'A':
                count = count + 1

print('Character A occured ' , count , 'number of times')
```

Determine Length of String

```
print('Enter your string ')

data = input()

print('Length = '  , len(data))
```

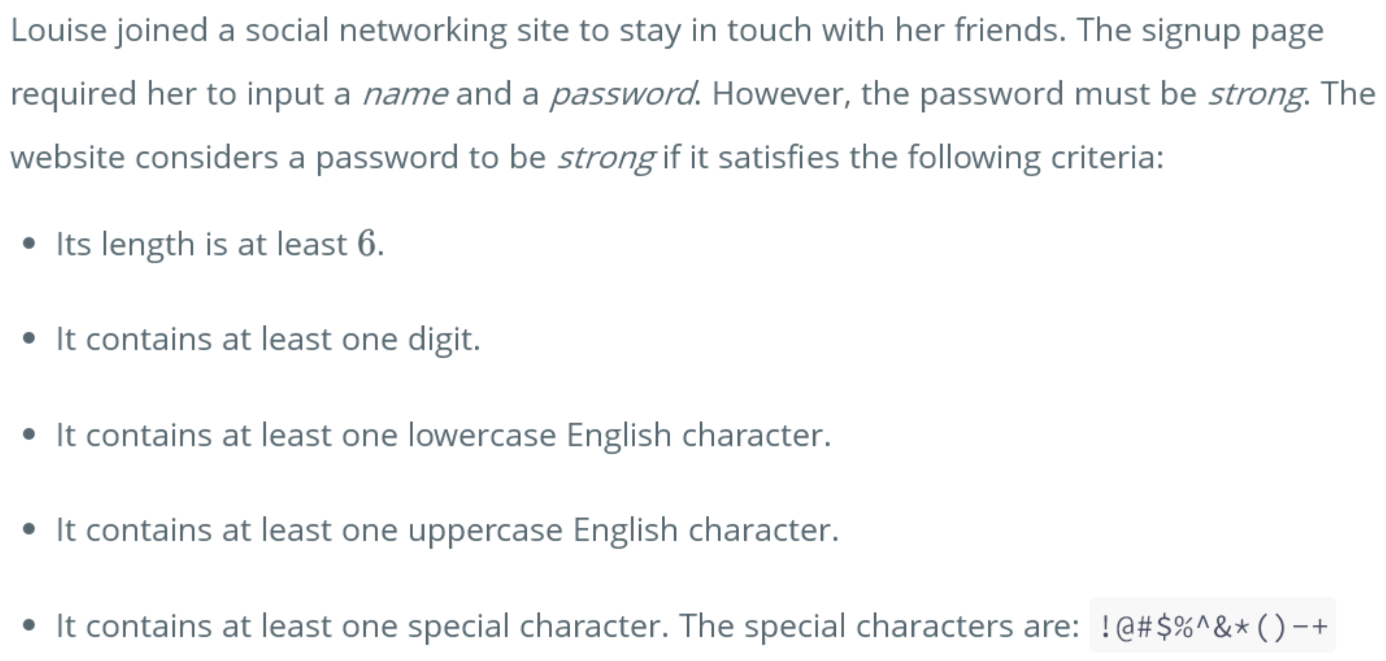
String Methods

str.isalnum()
str.isalpha()
str.isdigit()
str.islower()
str.isspace()
str.isupper()
str.istitle()



```python
data = input("Enter a string ")
print("Length = " , len(data))

digits = 0
upper = 0
lower = 0
space = 0
for ele in data:
    if ele.isdigit() == True:
        digits += 1
    if ele.isupper() == True:
        upper += 1
    if ele.islower() == True:
        lower += 1
    if ele.isspace() == True:
        space += 1
print("The string contains {0} digits".format(digits))
print("The string contains {0} UpperCase".format(upper))
print("The string contains {0} LowerCase".format(lower))
print("The string contains {0} Spaces".format(space))
```

Strong Password



Louise joined a social networking site to stay in touch with her friends. The signup page required her to input a *name* and a *password*. However, the password must be *strong*. The website considers a password to be *strong* if it satisfies the following criteria:

- Its length is at least 6.
- It contains at least one digit.
- It contains at least one lowercase English character.
- It contains at least one uppercase English character.
- It contains at least one special character. The special characters are: `!@#$%^&*()-+`

- Write code to check if the given password is strong or not.

String Methods

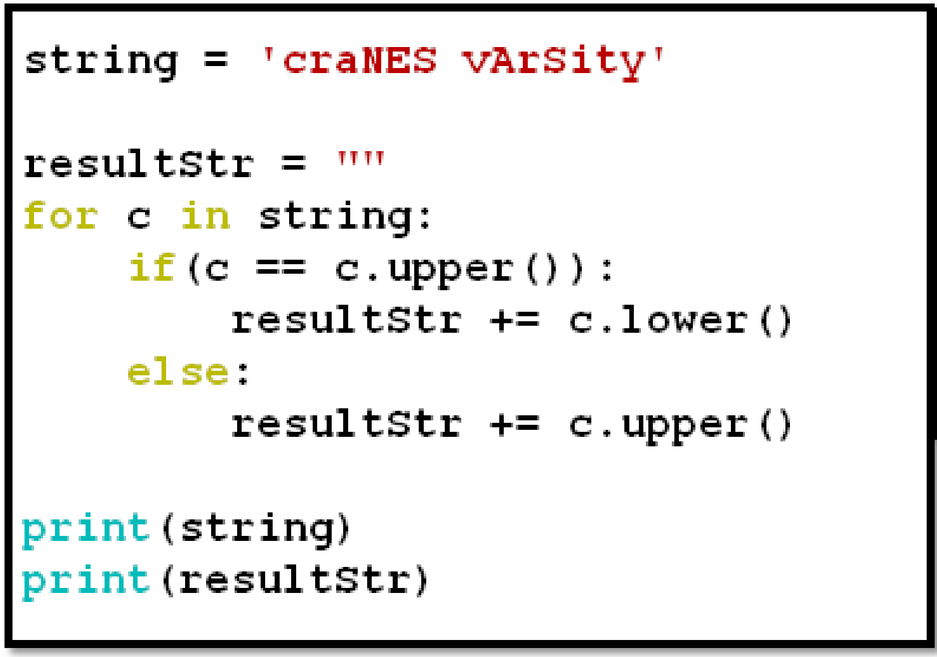
str.lower()	Convert all characters to lowercase
str.upper()	Convert all characters to uppercase
str.swapcase()	Convert uppercase to lowercase and vice versa
str.title()	First char of each word is changed to uppercase and others to lowercase
str.capitalize()	First char of the string is changed to uppercase others to lowercase

Python String Methods

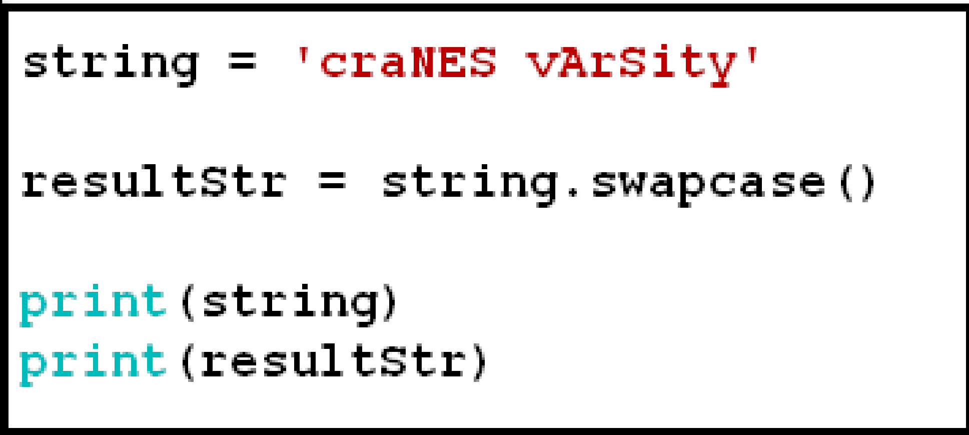
```
string = 'Cranes Varsity'

allCaps = string.upper()

allLower = string.lower()

print(string)

print(allCaps)

print(allLower)
```

▪

- You are given a string and your task is to *swap cases*. In other words, convert all lowercase letters to uppercase letters and vice versa.



```python
string = 'craNES vArSity'

resultStr = ""
for c in string:
    if(c == c.upper()):
        resultStr += c.lower()
    else:
        resultStr += c.upper()

print(string)
print(resultStr)
```



```python
string = 'craNES vArSity'

resultStr = string.swapcase()

print(string)
print(resultStr)
```

- Count the number of vowels in a string.

```
email = "sikander1248@gmail.com"

vowels = "aeiou"
vCount = 0
for c in email:
    if c in vowels:
        vCount += 1
print("Number of vowels = " , vCount)
```

String Methods

str.count(*sub*)	Return the number of non-overlapping occurrences of substring *sub* .
str.find(*sub*)	Return the lowest index in the string where substring *sub* is found. **Return -1 if *sub* is not found**.
str.index(*sub*)	Like find(), but raise ValueError when the substring is not found.
str.split(*sep* = *None*)	Return a list of the words in the string, using *sep* as the delimiter string.
S.join(list)	Return a string which is the concatenation of the strings in the list. The separator between elements is S.

```
mainStr = "abababa"
subStr = "aba"

res = mainStr.count(subStr)

print(subStr ,'occurred', res,'times')
```

Count number of overlapping substrings

```
mainStr = "abababa"
subStr = "aba"

try:
    count = 0
    index = 0
    while True:
        mainStr = mainStr[index:]
        print(mainStr)
        index = mainStr.index(subStr)
        index = index + 1
        count = count + 1
except:
      print(subStr ,'occurred', count,'times')
```

```
mainStr = "abababa"
subStr = "aba"

count = 0
index = 0
while True:
    mainStr = mainStr[index:]
    print(mainStr)
    index = mainStr.find(subStr)
    if index == -1:
        break
    index = index + 1
    count = count + 1

print(subStr ,'occurred', count,'times')
```

Split method

- Return a list of the words in String

```
data = "cranes varsity bangalore"

print(data)
print(type(data))
print(len(data))

words = data.split()
print(words)
print(type(words))
print(len(words))
```

Split - Seperator

- Default separator is any space(space,newline,tab)
- To specify any other separator, specify it explicitly.

```
data = "sindhu,niranjani,yoganand,ravi"

print(data)
print(type(data))
print(len(data))

words = data.split(',')
print(words)
print(type(words))
print(len(words))
```

Join Method

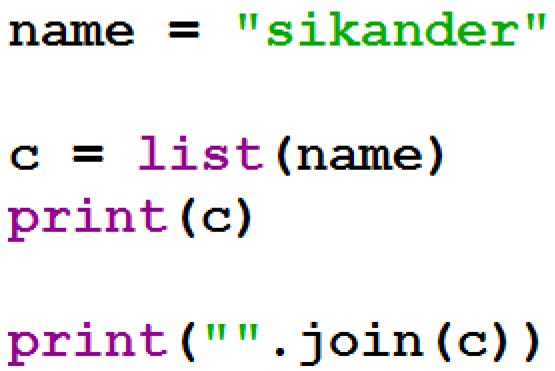
- Used to efficiently construct strings from multiple fragments.

- str.join(*iterable*)

- Return a string which is the concatenation of the strings in *iterable*.
- The separator between elements is the string providing this method.

- A TypeError will be raised if there are any non-string values in *iterable*.

- vowels = ['a', 'e', 'i', 'o', 'u']
- vowels
- ['a', 'e', 'i', 'o', 'u']
- s = "-".join(vowels)
- s
- 'a-e-i-o-u'



```
name = "sikander"

c = list(name)
print(c)

print("".join(c))
```

- nums = [12,23,21,43,56]
- nums
- [12, 23, 21, 43, 56]
- s = "".join(nums)
- TypeError: sequence item 0: expected str instance, int found

- s = "".join(map(str , nums))
- s
- '1223214356'

Replace Method

- Replace method

```
state = "KARNATAKA"

print( state.replace('A','') )
#KRNTK

print(state)
#KARNATAKA
```

Translate

```
a = "abcdefghijklmnopqrstuvwxyz"
b = "efghijklmnopqrstuvwxyzabcd"

encryptTransTable = str.maketrans(a, b)

data = input("Enter the Data ")
print("Original Data :", data)

encryptedData = data.translate(encryptTransTable)
print("Encrypted Data:", encryptedData)

decryptTransTable = str.maketrans(b, a)
decryptedData = encryptedData.translate(decryptTransTable)
print("Decrypted Data:",decryptedData)
```

Anagram

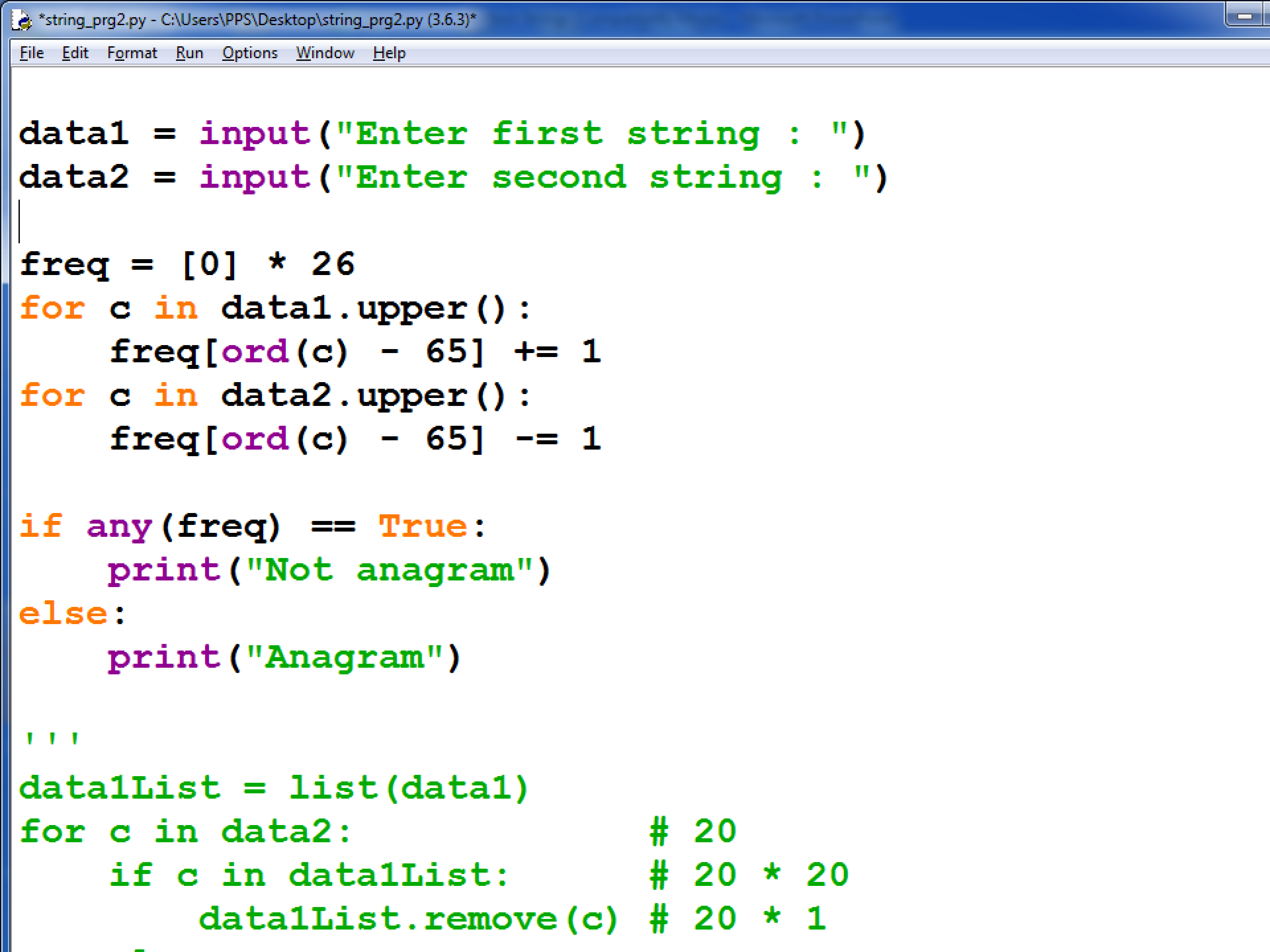
- An **anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.
- Anagram Example:

LISTEN	**SILENT**
BINARY	BRAINY
SCHOOL MASTER	**THE CLASSROOM**
CAR	ARC

Anagram

- Given two strings, verify if they are anagram

```
s1 = input("Enter first string : ")
s2 = input("Enter Second String : ")

if sorted(s1) == sorted(s2):
    print("Anagram")
else:
    print("Not an Anagram")
```


string_prg2.py - C:\Users\PPS\Desktop\string_prg2.py (3.6.3)
File Edit Format Run Options Window Help

```
data1 = input("Enter first string : ")
data2 = input("Enter second string : ")

freq = [0] * 26
for c in data1.upper():
    freq[ord(c) - 65] += 1
for c in data2.upper():
    freq[ord(c) - 65] -= 1

if any(freq) == True:
    print("Not anagram")
else:
    print("Anagram")

'''
data1List = list(data1)
for c in data2:              # 20
    if c in data1List:       # 20 * 20
        data1List.remove(c)  # 20 * 1
```

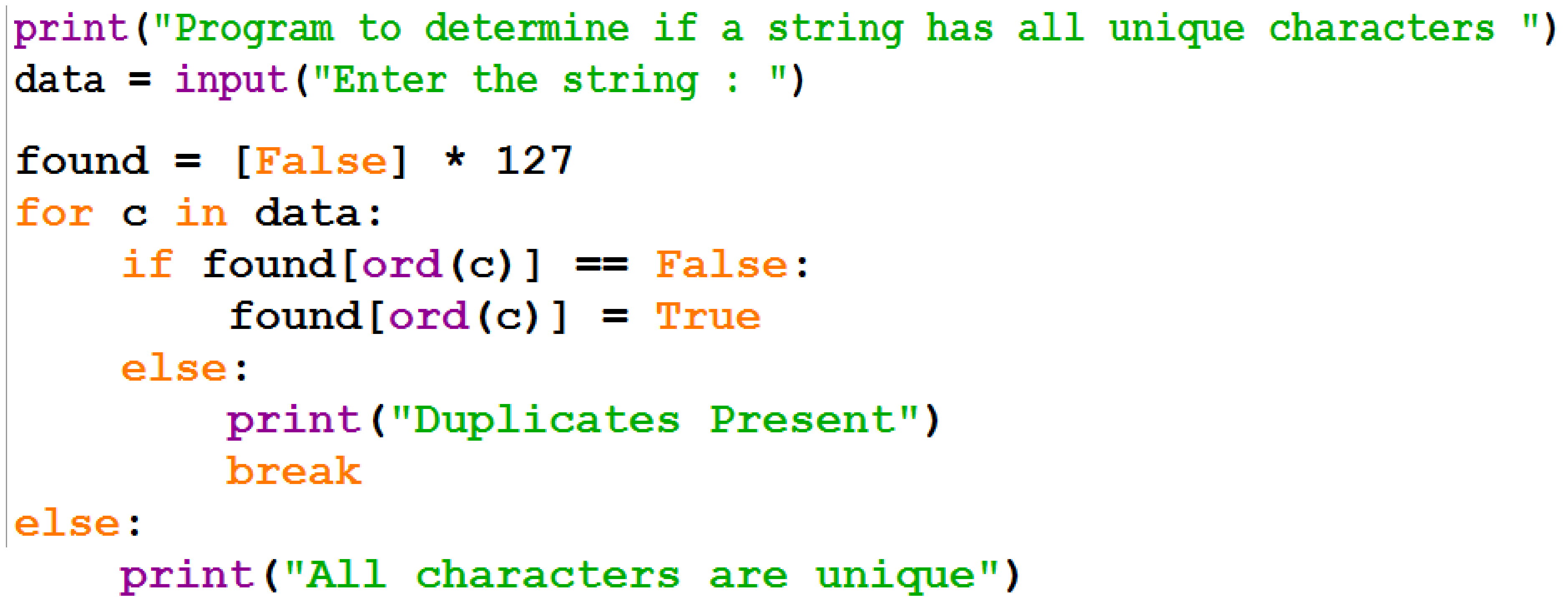
Pangram

- Pangram is a sentence containing every letter of the alphabet.
- Given a sentence, determine whether it is pangram, ignore case.

- Pangram Examples:
- The quick brown fox jumps over the lazy dog
- Two driven jocks help fax my big quiz.
- Pack my box with five dozen liquor jugs.
- The five boxing wizards jump quickly.
- Bright vixens jump; dozy fowl quack.

```
print("Verify if a given string is pangram ")
data = input("Enter the string : ")

data = data.upper()

freq = [False] * 26

for c in data:
    if c.isalpha():
        freq[ord(c) - 65] = True

if all(freq) == True:
    print("Pangram")
else:
    print("Not a Pangram")
```

- Write a program to verify if the given string has all unique characters



```python
print("Program to determine if a string has all unique characters ")
data = input("Enter the string : ")

found = [False] * 127
for c in data:
    if found[ord(c)] == False:
        found[ord(c)] = True
    else:
        print("Duplicates Present")
        break
else:
    print("All characters are unique")
```

- Write a program to remove all duplicate characters from a string.

```python
print("Program to remove all duplicate characters ")
data = input("Enter the string :  ")

unique = ""

for c in data:
    if c not in unique:
        unique += c

print(unique)
```

Reverse a String

```
i = len(data) - 1
reverse = ""
while i >= 0:
    reverse = reverse + data[i]
    i -= 1
print(reverse)
```

```
reverse = data[::-1]
print(reverse)
```

```
name = "SIKANDER"

charList = list(name)
charList.reverse()
print(charList)
reverseString = "".join(charList)
print(reverseString)
```