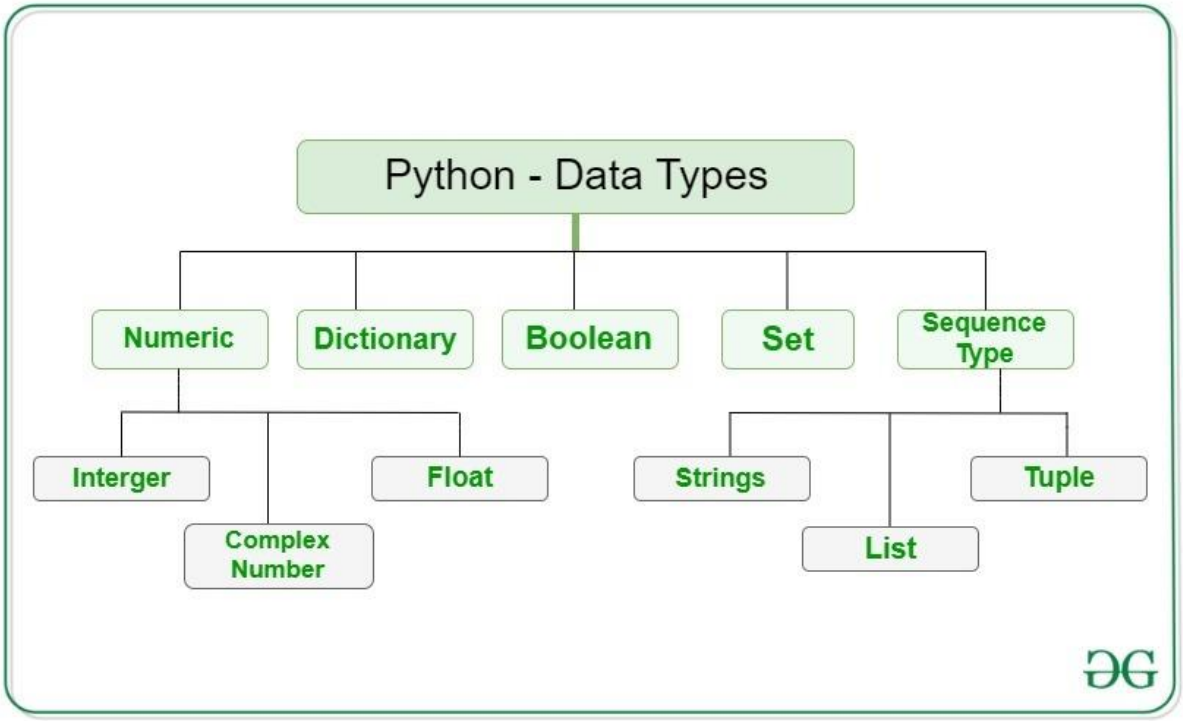
C	PYTHON
An Imperative programming model is basically followed by C.	An object-oriented programming model is basically followed by Python.
Variables are declared in C.	Python has no declaration.
C doesn’t have native OOP.	Python has OOP which is a part of language.
Pointers are available in C language.	No pointers functionality is available in Python.
C is a compiled language.	Python is an interpreted language.
There is a limited number of built-in functions available in C.	There is a large library of built-in functions in Python.
Implementation of data structures requires its functions to be explicitly implemented.	It is easy to implement data structures in Python with built-in insert, append functions.
C is compiled direct to machine code which is executed directly by the CPU	Python is firstly compiled to a byte-code and then it is interpreted by a large C program.
Declaring of variable type in C is necessary condition.	There is no need to declare a type of variable in Python.
C does not have complex data structures.	Python has some complex data structures.
C is statically typed.	Python is dynamically typed.

C	PYTHON
Syntax of C is harder than python because of which programmers prefer to use python instead of C	It is easy to learn, write and read Python programs than C.
C programs are saved with .c extension.	Python programs are saved by .py extension.
An assignment is allowed in a line.	Assignment gives an error in line. For example, a=5 gives an error in python.
In C language testing and debugging is harder.	In Python, testing and debugging is not harder than C.
C is complex than Python.	Python is much easier than C.
The basic if statement in c is represented as: if ()	The basic if statement in Python is represented as: if:

The basic if-else statement in Python is represented as:
if ( )
else
	The basic if-else statement is represented as:
	if :
	else:

C language is fast.	Python programming language is slow

2) Explain built-in data-type of python.



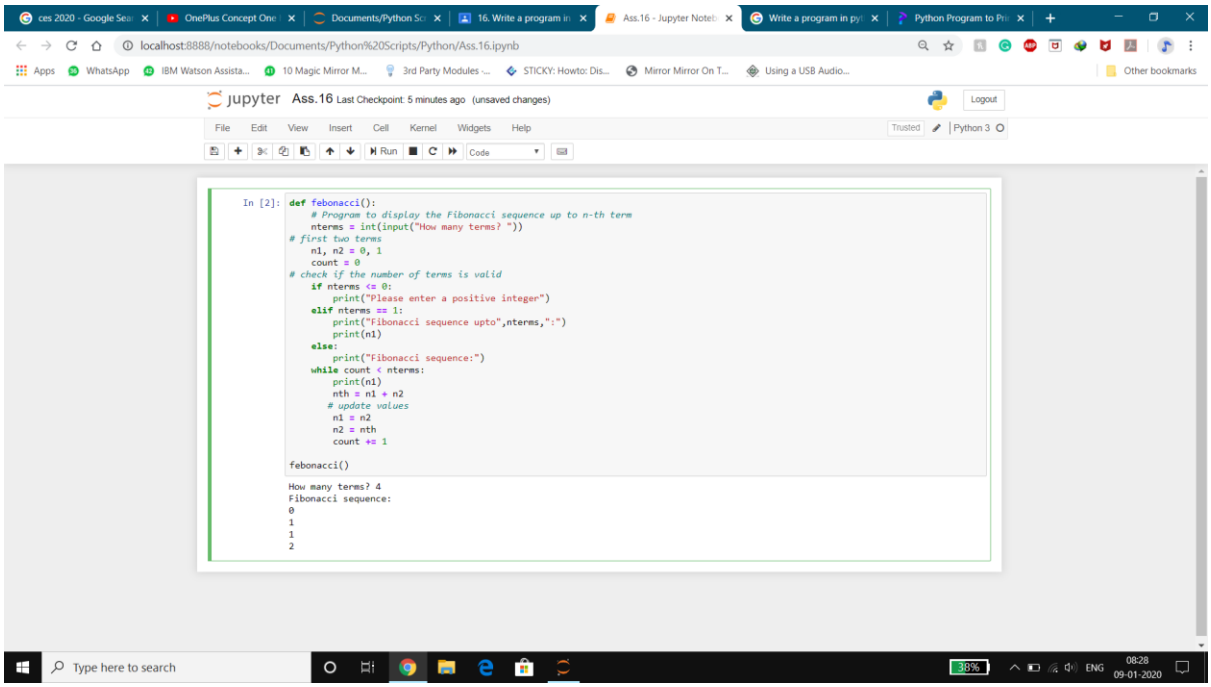
3) Write a Python program that counts the number of occurrences of the character in the given string. Provide two implementations: recursive and iterative.

In [*]:
```
#Write a iterative Python program that counts the number of occurrences of the character in the given string.
def Counter():
    inp=input('Enter String:')
    ch=input('Enter Which Char. you want to count:')
    count = 0

    for i in inp:
        if i == ch:
            count = count + 1

print ("Count of ",ch, "in" ,inp  ,"is : ", str(count))

Counter()
```

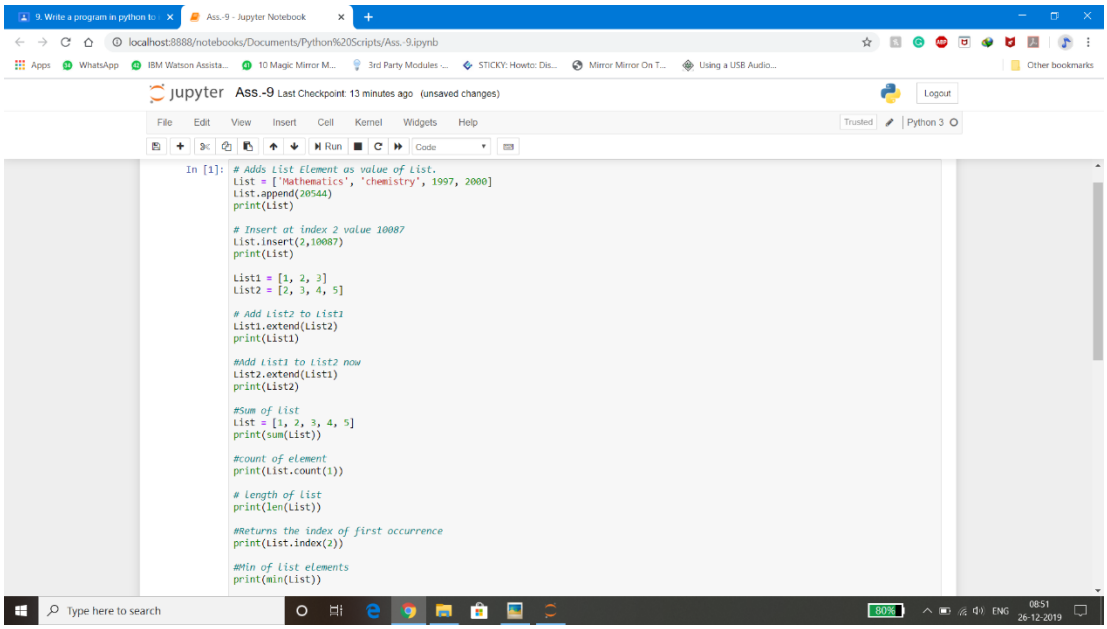
In [*]:
```
#Write a recursive Python program that counts the number of occurrences of the character in the given string.

def countSubstrig(str1, str2):

    n1 = len(str1)
    n2 = len(str2)

    # Base Case
    if (n1 == 0 or n1 < n2):
        return 0

    # Recursive Case
    # Checking if the first
    # substring matches
    if (str1[0 : n2] == str2):
        return countSubstrig(str1[n2 - 1:], str2) + 1

    # Otherwise, return the count
    # from the remaining index
    return countSubstrig(str1[n2 - 1:],str2)

# Driver Code
str1 = input('Enter String:')
str2 = input('Enter Which Char. you want to count:')
print(countSubstrig(str1, str2))
```

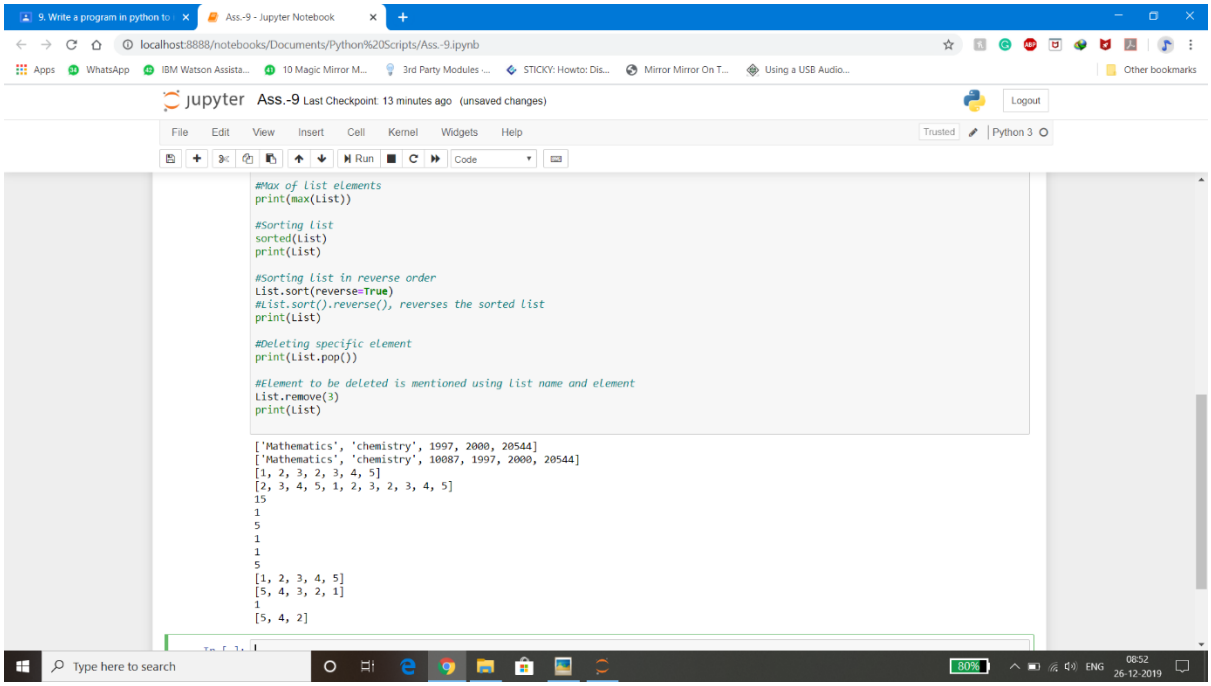
4. How to comment specific line(s) in Python program?
Same as 26

5. Write a Python program to print Fibonacci series up to n terms.



6) What is list in Python? Demonstrate use of any three methods of list.





7) What is the use of islower() and isupper() method?

Syntax :

```
string.islower()
Parameters:
islower() does not take any parameters
Returns :
1.True- If all characters in the string are lower.
2.False- If the string contains 1 or more non-lowercase characters.
```

```
string.isupper()
Parameters:
isupper() does not take any parameters
Returns :
1.True- If all characters in the string are uppercase.
2.False- If the string contains 1 or more non-uppercase characters.
```

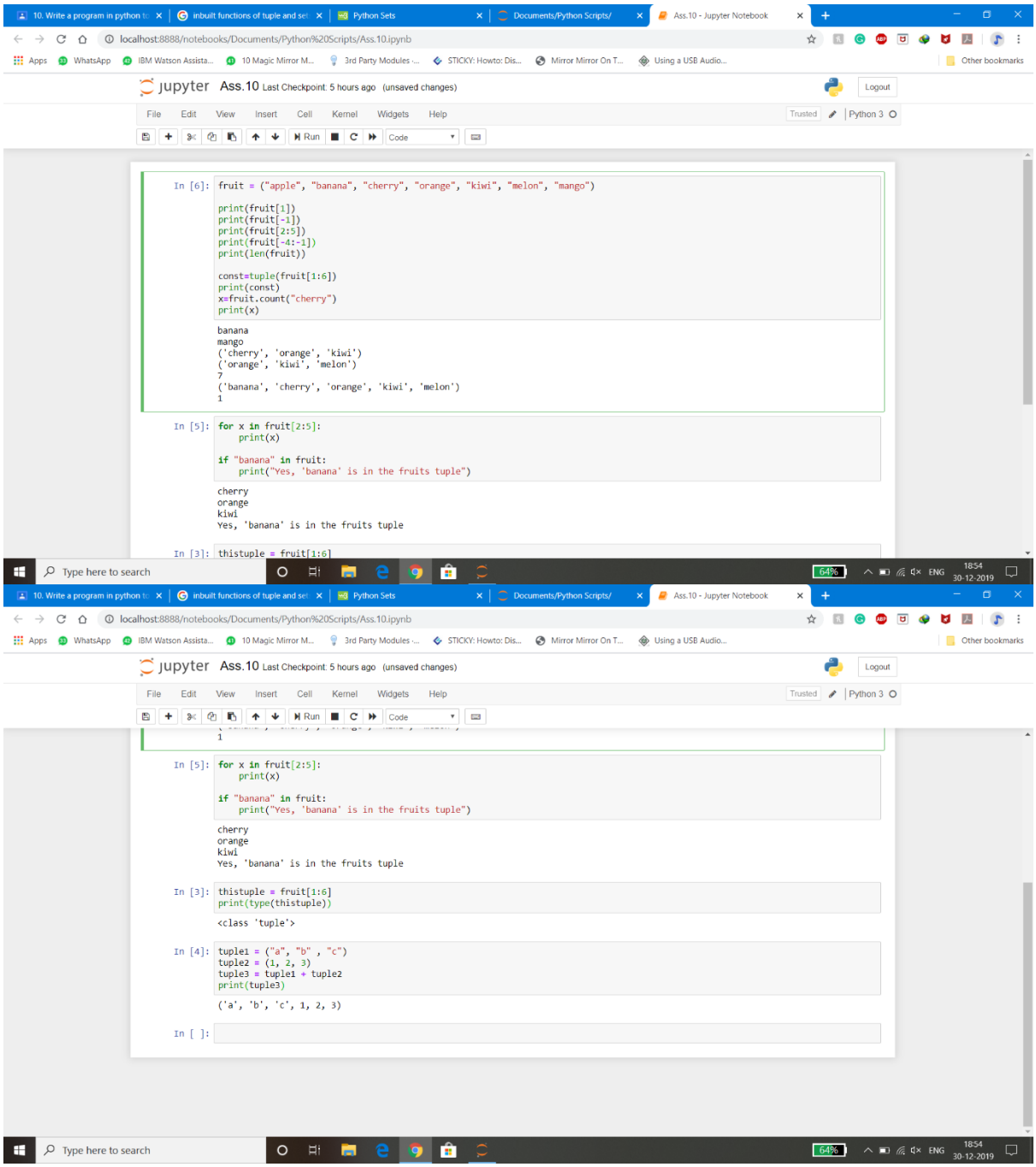
8) Give the syntax and significance of raw_input() and input() methods.

Same as 21

9) Write a Python program to check whether the given no is Armstrong or not using user defined function.

```
In [ ]: # Python program to check if the number is an Armstrong number or not
        # take input from the user
        num = int(input("Enter a number: "))
        # initialize sum
        sum = 0
        # find the sum of the cube of each digit
        temp = num
        while temp > 0:
            digit = temp % 10
            sum += digit ** 3
            temp //= 10
        # display the result
        if num == sum:
            print(num,"is an Armstrong number")
        else:
            print(num,"is not an Armstrong number")
```

10) Is tuple mutable? Demonstrate any two methods of tuple



11.Give the output of following Python code:

```
str1 = ‘This is Python’
print "Slice of String : ", str1[1 : 4 : 1]
print "Slice of String : ", str1[0 : -1 : 2]
s = 'This is Puthon'
print(s[1:4:1])
print(s[0:-1:2])
```

12. What is dictionary in Python? Explain with an example.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 2018
}
print(thisdict)
```

13. What is function in Python? Explain with an example

```
def my_function():
  print("Hello")
my_function()
```

14.Give syntax of the methods which can be used to take input from the user in python program.

```
print('Enter your name:')
x = input()
print('Hello, ' + x)
```

15. Give the syntax and significance of string functions: title() and strip().

```
string = " hello world "
print(string.strip(' world'))
txt = "Welcome to my world"
x = txt.title()
print(x)
```

16.How to comment specific line(s) in Python program?

```
def comments():
   #hello world
   print(comments)
```

17. How append() and extend() are different with reference to list in Python?

```
my_list = ['hello', 'world']
my_list.append('hello')
print(my_list)

my_list = ['hello', 'world']
another_list = [6, 0, 4, 1]
my_list.extend(another_list)
print(my_list)
```

18.What is the difference between = = and is operator in Python?

```
list1 = []
list2 = []
list3=list1

if (list1 == list2):
  print("True")
else:
  print("False")
if (list1 is list2):
  print("True")
else:
  print("False")

if (list1 is list3):
  print("True")
else:
  print("False")
```

19. Give the output of following Python code:

```
myStr = 'PPSU University’
print myStr [15 : : 1]
print myStr [-10 : -1 : 2]

mystr="ppsu university"
print(mystr[15::1])
print(mystr[-10:-1:2])
```

20. Explain List in Python:

thislist = ["apple", "banana"]

print(thislist)

21. raw_input & input :

Raw_input syntax:-

In python v2.x:

```
name=raw_input('Enter your name : ')
print ("Hi %s, Let us be friends!" % name);
```

In python v3.x:

```
name=input('Enter your name : ')
print ("Hi %s, Let us be friends!" % name);
```

output

Enter your name : nixCraft

Hi nixCraft, Let us be friends!

22. Explain Tuples in Python with example

A Tuple is a collection of Python objects separated by commas. In someways a tuple is similar to a list in terms of indexing, nested objects and repetition but a tuple is immutable unlike lists which are mutable.

Creating Tuples

```
# An empty tuple
empty_tuple = ('python')
print (empty_tuple)
```

Output

Python

23. Write a Python program to generate the fibonacci series using recursion.

Function for nth Fibonacci number

```
def Fibonacci(n)
    if n<0:
        print("Incorrect input")
    # First Fibonacci number is 0
    elif n==1:
        return 0
    # Second Fibonacci number is 1
    elif n==2:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)
# Driver Program
print(Fibonacci(9))
```

output:

21

24. Is String a mutable data type? Also explain the string operations length, indexing and slicing in detail with an appropriate example.

Some of these objects like lists and dictionaries are **mutable** , meaning you can change their content without changing their identity. Other objects like integers, floats, **strings** and tuples are objects that can not be changed. **Strings** are **immutable in Python**, which means you cannot change an existing **string**.

Indexing:

```
sample_str = 'Python String'
print (sample_str[0])      # return 1st character
# output: P
```

Slicing:

```
sample_str = 'Python String'
print (sample_str[3:5])          #return a range of character
# ho
```

String length:

Length of below string is 5

string = "geeks"
print(len(string))
output:5

25. What do you mean by immutable data type? Explain immutable data types with their operations and functions.

- Some of the **mutable** data types in Python are **list, dictionary, set** and **user-defined classes**.
- On the other hand, some of the **immutable** data types are **int, float, decimal, bool, string, tuple, and range**.

It's time for some examples. Let's start by comparing the **tuple** (**immutable**) and **list** (**mutable**) data types. We can define a list using **square brackets []** like this: numbers = [1, 2, 3]. To define a tuple, we just need to replace the brackets with **parentheses ()** like this: numbers = (1, 2, 3). From both data types, we can access elements by index and we can iterate over them. The main difference is that a tuple cannot be changed once it's defined.

Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	

26. Explain different types of comments in python.

Python provides three kinds of comments including block comment, inline comment and documentation string

Single line comment:

this is a single line comment

Multiline comment:

"""

This is python

You are using a python

Hello

""

27. Explain Indexing and Slicing operation for string manipulation with example in python.

Indexing:

```
sample_str = 'Python String'
print (sample_str[0])      # return 1st character
# output: P
```

Slicing:

```
sample_str = 'Python String'
print (sample_str[3:5])          #return a range of character
# ho
```

28. Explain Tuples, Lists and Dictionaries with example and give comment on mutability for each of them.

List:(mutable)

```
L = [1, "a" , "string" , 1+2]

print L

L.append(6)

print L

L.pop()

print L

print L[1]
```

output:

```
[1, 'a', 'string', 3]

[1, 'a', 'string', 3, 6]

[1, 'a', 'string', 3]

a
```

tuples:(immutable)

```
tup = (1, "a", "string", 1+2)
print tup
print tup[1]
```

output:

(1, 'a', 'string', 3)

a

Dictionaries:(mutable)

```
# Creating a Dictionary

# with Integer Keys

Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}

print("\nDictionary with the use of Integer Keys: ")

print(Dict)

Dict[1]='world'

print(Dict)
```

output:

Dictionary with the use of Integer Keys:

1: 'Geeks', 2: 'For', 3: 'Geeks'

1: 'world', 2: 'For', 3: 'Geeks'

29. Explain Control Flow Structure with an example.

Continue

It returns the control to the beginning of the loop.

```
# Prints all letters except 'e' and 's'
for letter in 'geeksforgeeks':
    if letter == 'e' or letter == 's':
         continue
    print 'Current Letter :', letter
    var = 10
```

output:

Current Letter : g

Current Letter : k

Current Letter : f

Current Letter : o

Current Letter : r

Current Letter : g

Current Letter : k

Break

It brings control out of the loop

```
for letter in 'geeksforgeeks':

    # break the loop as soon it sees 'e'
    # or 's'
    if letter == 'e' or letter == 's':
         break

print 'Current Letter :', letter
```

output:

Current Letter : e

Pass

We use pass statement to write empty loops. Pass is also used for empty control statement, function and classes.

```
# An empty loop
for letter in 'geeksforgeeks':
    pass
print 'Last Letter :', letter
```

output:

Last Letter : s

30. How many operators are there in Python. Explain any one of them.

We have multiple operators in Python, and each operator is subdivided into other operators. Let's list them down and know about each operator in detail.

- Arithmetic operators
- Comparison operators
- Assignment operators
- Logical operators
- Bitwise operators
- Membership operators
- Special operators
    - Identity operators
    - Membership operators

Arithmetic operators:

+	addition operator
—	subtraction operator
*	multiplication operator
/	division operator
//	integer divide operator
%	modulus (remainder) operator

```
x = 10
y = 20
# output: x - y = - 10(subtraction)
print ('x - y =', x - y )
```

Python Assignment Operators:

- = Assign Value
- += Add AND
- -= Subtract AND
- *= Multiply AND
- /= Divide AND
- %= Modulus AND
- **= Exponent AND
- //= Floor Division

Python logical Operators:

- And - Logical AND
- Or - Logical OR
- Not - Logical NOT

Python Bitwise operators:

- & Binary AND
- | Binary OR
- ^ Binary XOR
- ~ Binary Ones Complement
- << Binary Left Shift
- >> Binary Right Shif

31. Take one tuple and then exchange one element of that tuple and get the final output.

Tuple = (7,6,3,9)
Exchange 2nd element of given tuple.
Output will be Tuple = (7,4,3,9).