PYTHON PROGRAMMING

1	a) Demonstrate about Basics of Python Programming.	
	b) Demonstrate about fundamental Data types in Python Programming. (i.e., int, float, complex, bool and string types)	
	c) Demonstrate the working of following functions in Python. i) id( ) ii) type( ) iii) range( )	
	d) Write a Python program to demonstrate various base conversion functions.	
	e) Write a Python program to demonstrate various type conversion functions.	
2	a) Demonstrate the following Operators in Python with suitable examples. i) Arithmetic Operators ii) Relational Operators iii) Assignment Operator iv) Logical Operators v) Bit wise Operators vi) Ternary Operator vii) Membership Operators viii) Identity Operators	
3	a) Write Python programs to demonstrate the following: i) input( ) ii) print( ) iii) 'sep' attribute iv) 'end' attribute v) replacement Operator ({ })	
	b) Demonstrate the following Conditional statements in Python with suitable examples. i) if statement ii) if else statement iii) if – elif – else statement	

	c) Demonstrate the following Iterative statements in Python with suitable examples. i) while loop ii) for loop	
	d) Demonstrate the following control transfer statements in Python with suitable examples. i) break ii) continue iii) pass	
4	Write Python programs to print the following Patterns:	
	i) A A B A B C A B C D A B C D E	
	ii) * * * * * * * * * * * * * * *	
	iii) E E E E E E E E E D D D D D D D C C C C C B B B A	
	iv) 4 4 3 4 3 2 4 3 2 1 4 3 2 1 0 4 3 2 1 4 3 2 4 3 4	

	v) 4 3 4 2 3 4 1 2 3 4 0 1 2 3 4 1 2 3 4 2 3 4 3 4 4	
	vi) *	
	vii) ** ** **** **** ****** ****** ******** ******** ********** **********	
	viii) E D E C D E B C D E A B C D E B C D E C D E D E E	
5	a) Write a Python program to demonstrate various ways of accessing the string. i) By using Indexing (Both Positive and Negative) ii) By using Slice Operator	
	b) Demonstrate the following functions/methods which operates on strings in Python with suitable examples: i) len( ) ii) strip( ) iii) rstrip( ) iv) lstrip( ) v) find( ) vi) rfind( ) vii) index( ) viii) rindex()	

	ix) count( ) x) replace( ) xi) split( ) xii) join( ) xiii) upper( ) xiv) lower( ) xv) swapcase( ) xvi) title( ) xvii) capitalize( ) xviii) startswith() xix) endswith()	
7	a) Demonstrate the different ways of creating list objects with suitable example programs.	
	b) Demonstrate the following functions/methods which operates on lists in Python with suitable examples: i) list( ) ii) len( ) iii) count( ) iv) index ( ) v) append( ) vi) insert( ) vii) extend() viii) remove( ) ix) pop( ) x) reverse( ) xi) sort( ) xii) copy( ) xiii) clear( )	
	c) Demonstrate the following with suitable example programs: i) List slicing ii) List Comprehensions	
8	a) Demonstrate the different ways of creating tuple objects with suitable example programs.	
	b) Demonstrate the following functions/methods which operates on tuples in Python with suitable examples: i) len( ) ii) count( ) iii) index( ) iv) sorted( ) v) min ( ) vi)max( ) vii) cmp( ) viii) reversed( )	
9	a) Demonstrate the different ways of creating set objects with suitable example programs.	
	b) Demonstrate the following functions/methods which operates on sets in Python with suitable examples: i) add( ) ii) update( ) iii) copy( ) iv) pop( ) v) remove( ) vi)discard( ) vii) clear( ) viii) union() ix) intersection( ) x) difference( )	
10	a) Demonstrate the different ways of creating dictionary objects with suitable example programs.	
	b) Demonstrate the following functions/methods which operates on dictionary in Python with suitable examples: i) dict( ) ii) len( ) iii) clear( ) iv) get( ) v) pop( ) vi)popitem( ) vii) keys( ) viii) values() ix) items( ) x) copy( ) xi) update( )	
11	a) Demonstrate the following kinds of Parameters used while writing functions in Python. i) Positional Parameters ii) Default Parameters iii)Keyword Parameters iv) Variable length Parameters	
	b) Write a Python program to return multiple values at a time using a return statement.	
	c) Write a Python program to demonstrate Local and Global variables.	
	d) Demonstrate lambda functions in Python with suitable example programs.	

INTRODUCTION TO PYTHON PROGRAMMING

- Python is a high-level, general-purpose, interpreted, interactive and object-oriented programming language.
- It was created by Guido van Rossum during 1985- 1990.
- Like Perl, Python source code is also available under the GNU General Public License (GPL).

i. Why the name python?

Guido van Rossum was interested on watching a comedy show, which is telecasting on the BBC channel from 1969 to 1974 **The complete Monty Python's FlyingCircus**.

Guido Van Rossum thought he needed a name that was short,unique,andslightly mysterious for his programming language,so he decided to call the language **Python.**

ii. Applications of Python Programming

- Web Development
- Game Development
- Scientific and Numeric applications
- Artificial Intelligence and Machine Learning based applications
- Data Science related applications
- Desktop GUI applications
- Software Development
- Enterprise-level/Business Applications
- Education programs and training courses
- Web Scraping Applications
- Image Processing and Graphic Design Applications
- Data Analysis

iii. Features of Python programming

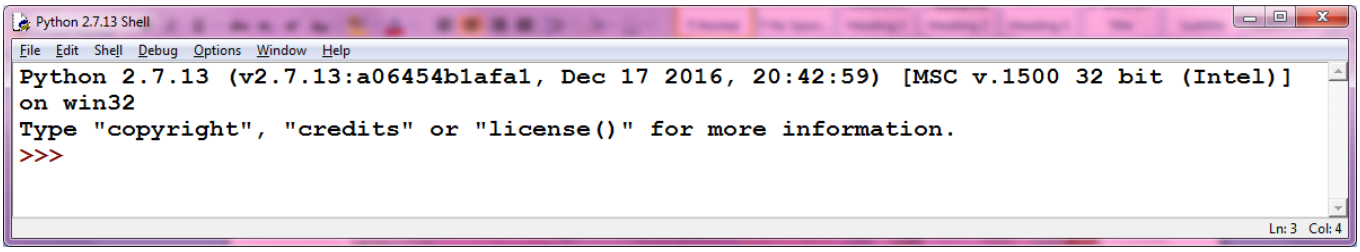
- Simple and Easy to Learn
- Freeware and Open Source
- Dynamically typed
- Object Oriented Programming and Procedure Oriented Programming
- Extensive Library
- Embedded
- Extensible
- Interpreted
- Portability
- Platform Independent

Experiment 1 :

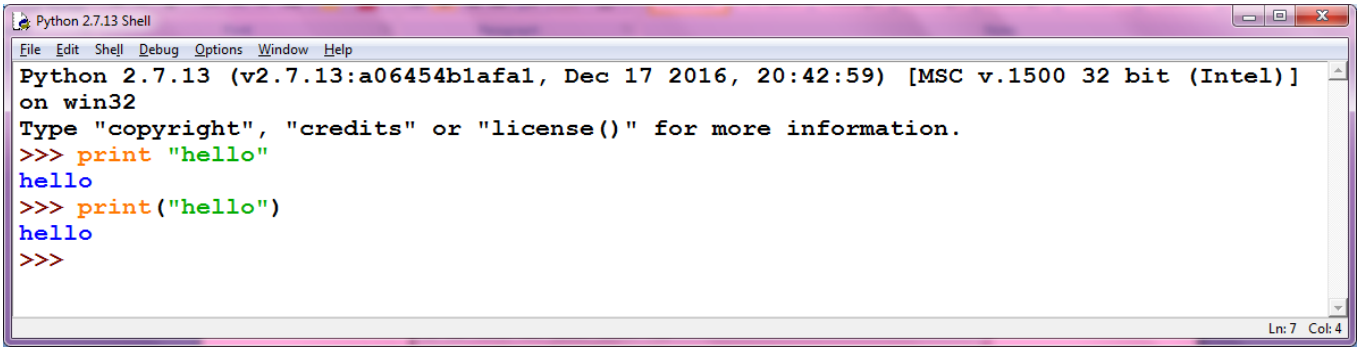
a) Demonstrate about Basics of Python Programming.

i. Running Python Interpreter:

- Python comes with an interactive interpreter.
- When you type **python** in your shell or command prompt, the python interpreter becomes active with a **>>>** (REPL) and waits for your commands.



- Now you can type any valid python expression at the prompt.
- Python reads the typed expression, evaluates it and prints the result.



ii. Running Python Scripts in IDLE (Integrated Development and Learning Environment):

- IDLE is the standard Python development environment.
- It's name is an acronym of **"Integrated DeveLopment Environment"**.
- It works well on both Unix and Windows platforms.
- It has a Python shell window, which gives you access to the Python interactive mode.
- It also has a file editor that lets you create and edit existing Python source files.

- Goto File menu click on New File (CTRL+N) and write the code and save add.py

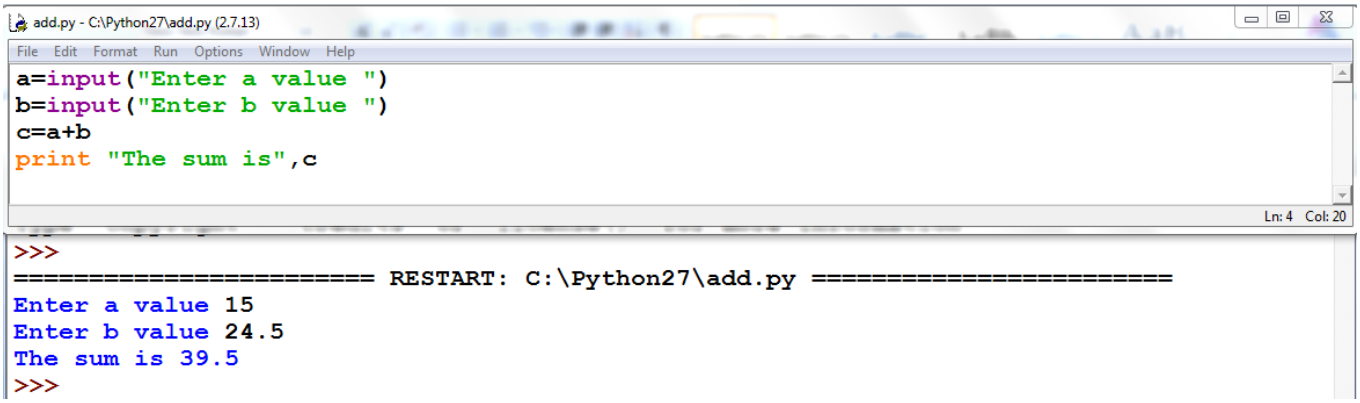
```
a=input("Enter a value ")

b=input("Enter b value ")

c=a+b

print "The sum is",c
```

- Then run the program by pressing **F5** or **Run ==> Run Module**.



iii. Running Python scripts in Command Prompt:

- Before going to run **python27** folder in the command prompt.
- Open your text editor, type the following text and save it as **hello.py.**

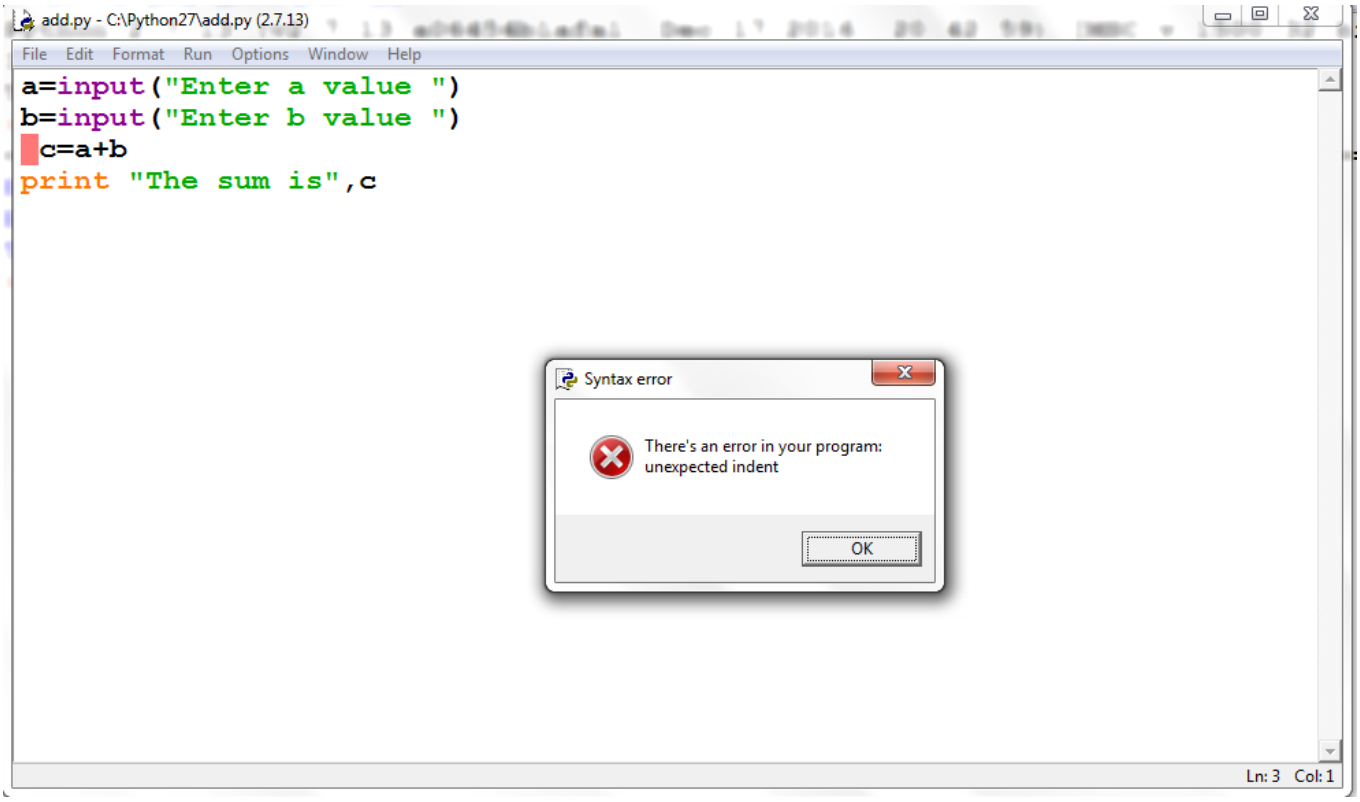
```
print "hello"
```

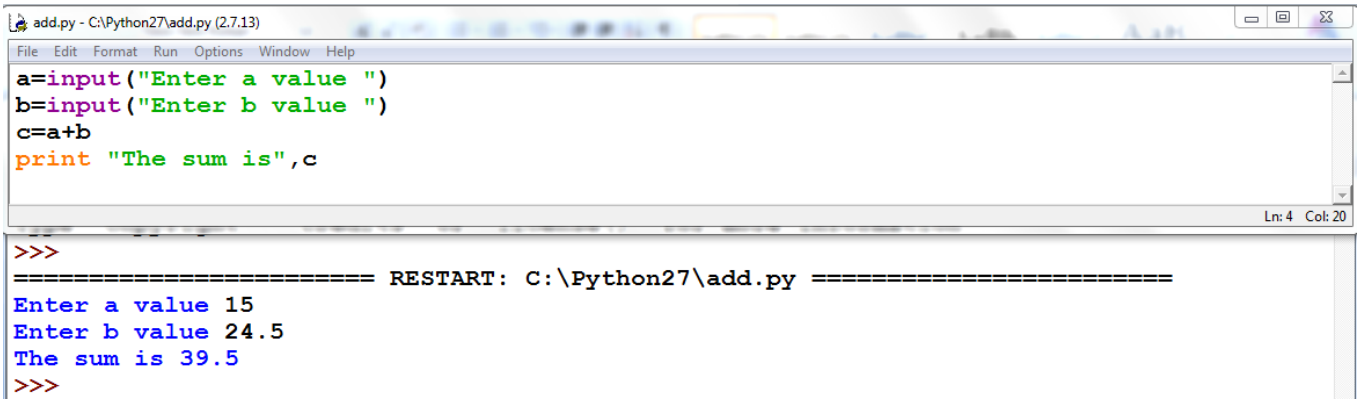
- In the command prompt, and run this program by calling python hello.py.
- Make sure you change to the directory where you saved the file before doing it.

Write a Python program to purposefully raise Indentation Error and correct it.

Indentation:

- Code blocks are identified by indentation rather than using symbols like curly braces.
- Indentation clearly identifies which block of code a statement belongs to. Of course, code blocks can consist of single statements, too.
- When one is new to Python, indentation may come as a surprise. Humans generally prefer to avoid change, so perhaps after many years of coding with brace delimitation, the first impression of using pure indentation may not be completely positive.
- However, recall that two of Python's features are that it is simplistic in nature and easy to read.
- Python does not support braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation.
- All the continuous lines indented with same number of spaces would form a block. **Python strictly follow indentation rules to indicate the blocks.**





b) Demonstrate about Fundamental Data types in Python Programming.

Description:

- Every value in Python has a datatype.
- Since everything is an object in Python programming, data types are actually classes and variables are instances (objects) of these classes.
- There are various data types in Python. Some of the important types are listed below.

1. Python Numbers

- Integers, floating point numbers and complex numbers fall under Python numbers category.
- They are defined as int, float and complex classes in Python.
- We can use the **type()** function to know which class a variable or a value belongs to.
- Similarly, the **isinstance()** function is used to check if an object belongs to a particular class.

Example Programs:

In [21]:

```
a = 5
print(a, "is of type", type(a))

a = 2.0
print(a, "is of type", type(a))

a = 1+2j
print(a, "is complex number?", isinstance(1+2j,complex))
```

```
5 is of type <class 'int'>
2.0 is of type <class 'float'>
(1+2j) is complex number? True
```

In [1]:

```
a=34
print(type(a))
b=456792357968700
print(b)
print(type(b))
```

```
<class 'int'>
456792357968700
<class 'int'>
```

Observation: int datatype is used to represent integral values. In python3 long int values can also be represented by using int type only.

In [2]:

```
f=3.413
print(f)
print(type(f))
f1=1.2e4
print(f1)
```

```
3.413
<class 'float'>
12000.0
```

Observation: float datatype is used to represent floating point values. We can represent float values scientifically by 'e'or'E'.

In [3]:

```
c=3+4j
print(type(c))
print(c.real)
print(c.imag)
```

```
<class 'complex'>
3.0
4.0
```

Observation: In complex datatype we have some inbuilt attributes to retrieve real and imaginary parts.

In [4]:

```
b=True
print(type(b))
a=5
b=8
c=a<b
print(c)
```

```
<class 'bool'>
True
```

Observation: This datatype is used to represent boolean values. The only allowed values for this datatype are True and False.

Points to Ponder

- Integers can be of any length, it is only limited by the memory available.
- A floating-point number is accurate up to 15 decimal places. Integer and floating points are separated by decimal points.
- Complex numbers are written in the form, **x + yj**, where x is the real part and y is the imaginary part. Here are some examples.

In [23]:

```
a = 1234567890123456789
print(a)

b = 0.1234567890123456789
print(b)                          # float variable 'b' truncated

c = 1+2j
print(c)
```

```
1234567890123456789
0.12345678901234568
(1+2j)
```

2. Python Strings

- String is sequence of Unicode characters.
- We can use single quotes or double quotes to represent strings.
- Multi-line strings can be denoted using triple quotes, ''' or """.

In [24]:

```
s = "This is a string"
print(s)
s = '''A multiline
string'''
print(s)
```

```
This is a string
A multiline
string
```

In [5]:

```
s1='python'
print(s1)
s2="record"
print(s2)
```

```
python
record
```

In [6]:

```
s="python is
a freeware"
print(s)
```

```
  File "<ipython-input-6-bccaede3693b>", line 1
    s="python is
               ^
SyntaxError: EOL while scanning string literal
```

Observation: Multiline string literals are represented by triple quotes.They cannot be represented by using single or double quotes.

c) Demonstrate the working of ‘id’ and ‘type’ functions.

1. id() function:

- The id() function returns identity (unique integer) of an object.

The syntax of id() is:

```
id(object)
```

- As we can see the function accepts a single parameter and is used to return the identity of an object.
- This identity has to be unique and constant for this object during the lifetime.
- Two objects with non-overlapping lifetimes may have the same id() value.
- If we relate this to C, then they are actually the memory address, here in Python it is the unique id.
- The output is the identity of the object passed.
- This is random but when running in the same program, it generates unique and same identity.

Examples:

In [1]:

```
id(1025)
```

Out[1]:

```
2497999538928
```

In [2]:

```
id(1025)    # Here, the Output varies with different runs
```

Out[2]:

```
2497999539408
```

In [3]:

```
id("RGM")
```

Out[3]:

```
2497997578032
```

In [4]:

```
s = "RGM"
print(id(s))
```

```
2497999585904
```

In [6]:

```
s2 = "RGM"
print(id(s2))
```

```
2497999585904
```

In [7]:

```
print(id(s)==id(s2))
```

```
True
```

In [8]:

```
# Use in Lists
list1 = ["Python", "Java", "C++"]
print(id(list1[0]))
print(id(list1[2]))
```

```
2497932653936
2497999588144
```

In [10]:

```
# This returns false
print(id(list1[0])==id(list1[2]))
```

```
False
```

Return Value from id():

- The id() function returns identity of the object.
- This is an integer which is unique for the given object and remains constant during its lifetime.

In [11]:

```
print('id of 5 =',id(5))
a = 5
print('id of a =',id(a))
b = a
print('id of b =',id(b))
c = 5.0
print('id of c =',id(c))
```

```
id of 5 = 140737105207824
id of a = 140737105207824
id of b = 140737105207824
id of c = 2497999538800
```

2. The 'type()' function:

- Python have a built-in method called as type which generally come in handy while figuring out the type of variable used in the program in the runtime.
- The type function returns the datatype of any arbitrary object.

In [21]:

```
print(type(5))
print(type(5.0))
print(type('5'))
print(type("5"))
print(type([]))
print(type(()))
print(type({}))
print(type({a,}))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'str'>
<class 'list'>
<class 'tuple'>
<class 'dict'>
<class 'set'>
```

- type takes anything and returns its data type. Integers, strings, lists, dictionaries, tuples, functions, classes, modules, even types are acceptable.
- type can take a variable and return its datatype.

3. range():

- range() function is used to generate sequence of numbers.

In [1]:

```
r = range(10)
print(type(r))
print(r)
```

```
<class 'range'>
range(0, 10)
```

In [2]:

```
r = range(10)
print(type(r))
print(r)
for i in r:
    print(i)
```

```
<class 'range'>
range(0, 10)
0
1
2
3
4
5
6
7
8
9
```

In [3]:

```
r = range(10)
print(type(r))
print(r)
for i in r:
    print(i,end = ' ')
```

```
<class 'range'>
range(0, 10)
0 1 2 3 4 5 6 7 8 9
```

In [4]:

```
r = range(10)
print(type(r))
print(r)
for i in r:
    print(i,end = '\t')
```

```
<class 'range'>
range(0, 10)
0	1	2	3	4	5	6	7	8	9
```

In [6]:

```
r = range(10,-25)
print(type(r))
print(r)
for i in r:
    print(i,end = ' ')
```

```
<class 'range'>
range(10, -25)
```

In [7]:

```
r = range(-25,10)
print(type(r))
print(r)
for i in r:
    print(i,end = ' ')
```

```
<class 'range'>
range(-25, 10)
-25 -24 -23 -22 -21 -20 -19 -18 -17 -16 -15 -14 -13 -12 -11 -10 -9 -8 -7 -
6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9
```

In [8]:

```
r = range(1,21,1)
for i in r:
    print(i,end = ' ')
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

In [9]:

```
r = range(1,21,2)
for i in r:
    print(i,end = ' ')
```

```
1 3 5 7 9 11 13 15 17 19
```

In [10]:

```
r = range(1,21,3)
for i in r:
    print(i,end = ' ')
```

```
1 4 7 10 13 16 19
```

In [11]:

```
r = range(1,21,4)
for i in r:
    print(i,end = ' ')
```

```
1 5 9 13 17
```

In [12]:

```
r = range(1,21,-5)
for i in r:
    print(i,end = ' ')
```

In [13]:

```
r = range(21,1,-5)
for i in r:
    print(i,end = ' ')
```

```
21 16 11 6
```

In [14]:

```
r = range(21,0,-5)
for i in r:
    print(i,end = ' ')
```

```
21 16 11 6 1
```

In [15]:

```
r = range(10,20)
print(r[0])
print(r[-1])
r1 = r[1:5]
print(r1)
for i in r1:
    print(i)
r[1] = 3445
```

```
10
19
range(11, 15)
11
12
13
14

---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-15-0c82f2e5b728> in <module>
      6 for i in r1:
      7     print(i)
----> 8 r[1] = 3445

TypeError: 'range' object does not support item assignment
```

d) Python Programs to demonstrate various Base Conversion functions.

In [16]:

```
print(bin(12))
print(bin(0XA11))
print(bin(0o2345))
print(bin(54.67))
```

```
0b1100
0b101000010001
0b10011100101

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-16-6aaa1bd5a457> in <module>
      2 print(bin(0XA11))
      3 print(bin(0o2345))
----> 4 print(bin(54.67))

TypeError: 'float' object cannot be interpreted as an integer
```

Observation: bin() is used to convert from any base to binary except float values.

In [17]:

```
print(oct(456))
print(oct(0b1101))
print(oct(0xAB123))
print(oct(56.5))
```

```
0o710
0o15
0o2530443

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-17-3d6cec98201e> in <module>
      2 print(oct(0b1101))
      3 print(oct(0xAB123))
----> 4 print(oct(56.5))

TypeError: 'float' object cannot be interpreted as an integer
```

Observation: oct() is used to convert from any base to octal except float values.

In [18]:

```
print(hex(675))
print(hex(0b1101))
print(hex(0o456))
print(hex(45.65))
```

```
0x2a3
0xd
0x12e

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-18-d5dc2a4af973> in <module>
      2 print(hex(0b1101))
      3 print(hex(0o456))
----> 4 print(hex(45.65))

TypeError: 'float' object cannot be interpreted as an integer
```

Observation: hex() is used to convert from any base to octal except float values.

e) Python Programs to demonstrate of various Type Conversion Functions.

- Converting the value from one type to another type is called as Type casting or Type Coersion.

In [19]:

```
print(int(45.56))
print(int(True))
print(int(False))
print(int(3+5j))
```

```
45
1
0

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-857e366a281f> in <module>
      2 print(int(True))
      3 print(int(False))
----> 4 print(int(3+5j))

TypeError: can't convert complex to int
```

Observation: We can use int() to convert from other types to int except complex.We get typeError when we try to convert complex to int.

In [20]:

```
print(int('10'))
print(int('10.5'))
print(int('ten'))
```

```
10

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-20-b088da98807d> in <module>
      1 print(int('10'))
----> 2 print(int('10.5'))
      3 print(int('ten'))

ValueError: invalid literal for int() with base 10: '10.5'
```

Observation: To convert string to int type, compulsary string should contain integer values and specified with base-10.

In [21]:

```
print(float('10'))
print(float('10.5'))
print(float('ten'))
```

```
10.0
10.5

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-21-3b0ac2ae8b09> in <module>
      1 print(float('10'))
      2 print(float('10.5'))
----> 3 print(float('ten'))

ValueError: could not convert string to float: 'ten'
```

Observation: To convert string to float is not possible when the string contains charecters.

In [22]:

```
print(float(3+4j))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-22-dd9e89f1504b> in <module>
----> 1 print(float(3+4j))

TypeError: can't convert complex to float
```

Observation: It is not possible to convert complex to float.

In [23]:

```
print(complex(2))
print(complex(5.0))
print(complex(True))
print(complex('10'))
print(complex('ram'))
```

```
(2+0j)
(5+0j)
(1+0j)
(10+0j)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-23-f42d0e340fec> in <module>
      3 print(complex(True))
      4 print(complex('10'))
----> 5 print(complex('ram'))

ValueError: complex() arg is a malformed string
```

Observation: It is possible to convert any type to complex except string contain characters.

In [24]:

```
print(bool(0))
print(bool(1))
print(bool(4))
print(bool(2.0))
print(bool(2+3j))
print(bool(True))
print(bool(False))
print(bool(-1))
```

```
False
True
True
True
True
True
False
True
```

Observation: To convert any type to bool is possiple.it always give either True or False.

In [25]:

```
print(str(12))
print(str(2.0))
print(str(2+8j))
print(str(True))
print(str(-122))
```

```
12
2.0
(2+8j)
True
-122
```

Observation: It is possible to convert from any type to string type.

Experiment 2: Demonstration of various operators used in Python with suitable example programs.

Operator is a symbol which can perform specified operation on operands.

Types of Operators used in Python:

1. Arithmetic operators.
2. Relational or Comparision operators.
3. Equality operators.
4. Logical operators.
5. Bitwise operators.
6. Shift operators.
7. Assignment operators.
8. Ternary operator. (or) Conditional operator.
9. Special operators:

    a. Identity operator.

    b. Membership operator.

10. Operator precedence.

1. Arithmetic Operators:

The arithmetic operations are the basic mathematical operators which are used in our daily life. Mainly it consists of seven operators.

i. Addition operator --> '+'

ii. Subtraction operator --> '-'

iii. Multiplication operator --> '*'

iv. Normal Division operator --> '/'

v. Modulo Division operator --> '%'

vi. Floor Division operator --> '//'

vii. Exponential operator (or) power operator --> '**'

i. Addition Operator :

- Generally addition operator is used to perform the addition operation on two operands.
- But in python we can use addition operator to perform the concatenation of strings, lists and so on, but operands must of same datatype.

In [1]:

```
x = 2
y = 3
print("ADDITION RESULT : ", x + y)
```

```
ADDITION RESULT :  5
```

In [2]:

```
x = 2
y = 3.3
print("ADDITION RESULT : ", x + y) # both float and int type are accept
```

```
ADDITION RESULT :  5.3
```

In [3]:

```
x = 2.7
y = 3.3
print("ADDITION RESULT : ", x + y) # both float type are accept
```

```
ADDITION RESULT :  6.0
```

In [4]:

```
x = "2"
y = 3
print("ADDITION RESULT : ", x + y) #str type and int can't be added.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-d873d6fd7998> in <module>
      1 x = "2"
      2 y = 3
----> 3 print("ADDITION RESULT : ", x + y) #str type and int can't be added.

TypeError: can only concatenate str (not "int") to str
```

In [5]:

```
x = "2"
y = "3"
print("ADDITION RESULT : ", x + y) # concatenation will take place
```

```
ADDITION RESULT :  23
```

In [6]:

```
x = "2"
y = 4.8
print("ADDITION RESULT : ", x + y) # float type and str typr can't be added.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-32bf23d43c09> in <module>
      1 x = "2"
      2 y = 4.8
----> 3 print("ADDITION RESULT : ", x + y) # float type and str typr can't be added.

TypeError: can only concatenate str (not "float") to str
```

In [7]:

```
x = 2
y = bool(4.8)
print("ADDITION RESULT : ", x + y) #here bool(4.8) returns True i.e, 1
```

```
ADDITION RESULT :  3
```

In [8]:

```
x = "2"
y = bool(4.8)
print("ADDITION RESULT : ", x + y) #bool type cant be concatenated.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-aa2b47f2b5f5> in <module>
      1 x = "2"
      2 y = bool(4.8)
----> 3 print("ADDITION RESULT : ", x + y) #bool type cant be concatenated.

TypeError: can only concatenate str (not "bool") to str
```

In [9]:

```
x = "2"
y = str(bool(4.8))
print("ADDITION RESULT : ", x + y)
#bool returns 1 generally but we converted into str then it gives True
```

```
ADDITION RESULT :  2True
```

In [10]:

```
x = "2"
y = str(complex(4.8))       #Here both strings so concatenation will take place
print("ADDITION RESULT : ", x + y)
```

```
ADDITION RESULT :  2(4.8+0j)
```

In [11]:

```
x = 2
y = complex(4.8)
print("ADDITION RESULT : ", x + y)
# here both are int type so addtion will take place
```

```
ADDITION RESULT :  (6.8+0j)
```

ii. Subtraction Operator :

- Generally subtraction operator is used to perform the subtraction operation on two operands.

In [12]:

```
a = 30
b = 10
print("Subtraction result : ",a-b)
```

```
Subtraction result :  20
```

In [13]:

```
a = 30
b = "10"
print("Subtraction result : ",a-b)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-f0fd62944ccb> in <module>
      1 a = 30
      2 b = "10"
----> 3 print("Subtraction result : ",a-b)

TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

In [14]:

```
a = "30"
b = "10"
print("Subtraction result : ",a-b)
# can not perform subtraction on str type operands.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-0bebbed27be9> in <module>
      1 a = "30"
      2 b = "10"
----> 3 print("Subtraction result : ",a-b)
      4 # can not perform subtraction on str type operands.

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

In [15]:

```
a = 20
b = 10.00
print("Subtraction result : ",a-b)
```

```
Subtraction result :  10.0
```

In [16]:

```
a = 20
b = bool(10)
print("Subtraction result : ",a-b)
```

```
Subtraction result :  19
```

iii. Multiplication operator :

- Generally multiplication operator is used to perform the multiplication operation on two operends
- But in python we can use multiplication operator to perform the repetition of strings, lists and so on, but operands must belongs to same datatype.

In [17]:

```
num1 = 23
num2 = 35
print("MULTIPLICATION RESULT : ",num1 * num2)
```

```
MULTIPLICATION RESULT :  805
```

In [18]:

```
num1 = 23
num2 = 35.0
print("MULTIPLICATION RESULT : ",num1 * num2)
```

```
MULTIPLICATION RESULT :  805.0
```

In [19]:

```
num1 = "23"
num2 = 5
print("MULTIPLICATION RESULT : ",num1 * num2) # 23 string will prints 5 times
```

```
MULTIPLICATION RESULT :  2323232323
```

In [20]:

```
num1 = "23"
num2 = "5"
print("MULTIPLICATION RESULT : ",num1 * num2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-e4135d9e3a29> in <module>
      1 num1 = "23"
      2 num2 = "5"
----> 3 print("MULTIPLICATION RESULT : ",num1 * num2)

TypeError: can't multiply sequence by non-int of type 'str'
```

In [21]:

```
l = "(1,2,3,4)"
print(float(l * 5))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-21-18109e54b2f8> in <module>
      1 l = "(1,2,3,4)"
----> 2 print(float(l * 5))

ValueError: could not convert string to float: '(1,2,3,4)(1,2,3,4)(1,2,3,4)(1,2,3,4)(1,2,3,4)'
```

In [22]:

```
l = "123"
print(float(l * 4))
#initially it will prints string 5 times and converts it into float
```

```
123123123123.0
```

In [23]:

```
l = "123"
b = 2.3
print("MULTIPLICATION RESULT : ", l * b)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-23-e235870edcad> in <module>
      1 l = "123"
      2 b = 2.3
----> 3 print("MULTIPLICATION RESULT : ", l * b)

TypeError: can't multiply sequence by non-int of type 'float'
```

In [24]:

```
l = "123"
b = bool(2.3)
print("MULTIPLICATION RESULT : ", l * b)
```

```
MULTIPLICATION RESULT :  123
```

In [25]:

```
l =[1,2,3]
m = [2,4,5]
print(l * m) # multiplication of two list data types is not possible
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-25-309b92e03dcb> in <module>
      1 l =[1,2,3]
      2 m = [2,4,5]
----> 3 print(l * m) # multiplication of two list data types is not possible

TypeError: can't multiply sequence by non-int of type 'list'
```

In [26]:

```
l = (5,6,7)
m = (1,2,3)
print(l* m) # multiplication of two tuple data types is not possible
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-91a31577591d> in <module>
      1 l = (5,6,7)
      2 m = (1,2,3)
----> 3 print(l* m) # multiplication of two tuple data types is not possible

TypeError: can't multiply sequence by non-int of type 'tuple'
```

In [27]:

```
l = bool(1)
m = bool(4657)
print(l * m) # as bool returns 1 it prints only one time
```

```
1
```

In [28]:

```
l = bool()
m = bool(123456789)
print(l*m) # As bool doesn't contain any value it consider as zero.
```

```
0
```

In [29]:

```
l = str(bool([1,2,3]))
m = 99
print(l*m)
```

```
TrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTr
ueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrue
TrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTr
ueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrue
TrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTrueTr
ueTrueTrueTrueTrueTrueTrue
```

In [30]:

```
l = bool([1,2,3])
m = 99
print(l*m)
```

```
99
```

iv. Division Operator :

- Generally division operator is used to perform the division operation on two operands.
- It returns the result in float type.

In [31]:

```
a = 3
b = 45
print("Division result : ", a/ b) # returns float value
```

```
Division result :  0.06666666666666667
```

In [32]:

```
a = 3
b = "45"
print("Division result : ", b / a)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call las
t)
<ipython-input-32-1b2bbedeebd4> in <module>
      1 a = 3
      2 b = "45"
----> 3 print("Division result : ", b / a)

TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

In [33]:

```
a = 3
b = 45.0000
print("Division result : ", b / a)
```

```
Division result :  15.0
```

In [34]:

```
a = 3
b = bool(0.0000)
print("Division result : ", a / b)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-34-854e10cbf4f9> in <module>
      1 a = 3
      2 b = bool(0.0000)
----> 3 print("Division result : ", a / b)

ZeroDivisionError: division by zero
```

In [35]:

```
a = 3
b = complex((90))
print("Division result : ", a / b)
```

```
Division result :  (0.03333333333333333+0j)
```

In [36]:

```
a = [1,2,3]
b = [7,8,9]
print("Division result : ", a / b)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-36-8289b4627a90> in <module>
      1 a = [1,2,3]
      2 b = [7,8,9]
----> 3 print("Division result : ", a / b)

TypeError: unsupported operand type(s) for /: 'list' and 'list'
```

In [37]:

```
a = (1,2,3)
b = (1,2,3)
print("Division result : ", a / b)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-37-f02db8ba9671> in <module>
      1 a = (1,2,3)
      2 b = (1,2,3)
----> 3 print("Division result : ", a / b)

TypeError: unsupported operand type(s) for /: 'tuple' and 'tuple'
```

In [38]:

```
a = {1,2,3}
b = {1,2,3}
print("Division result : ", a / b)
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-38-cd4ea53f676a> in <module>
      1 a = {1,2,3}
      2 b = {1,2,3}
----> 3 print("Division result : ", a / b)

TypeError: unsupported operand type(s) for /: 'set' and 'set'
```

In [39]:

```
l = bool()
m = bool(9)
print(l / m)
```

```
0.0
```

v. Modulo Division:

- It returns reminder.

In [40]:

```
a = 3
b = 4
print(a%b)
print(b%a)
```

```
3
1
```

vi.Floor Division:

Suppose 10.3 is there, what is the floor value of 10.3?

- Answer is 10

What is the ceil value of 10.3?

- Answer is 11

In [41]:

```
print(10/2)
```

```
5.0
```

In [42]:

```
print(10/3)
```

```
3.3333333333333335
```

- If you want to get integer value as result of division operation, you need to make use of floor division(//) operator.
- floor division(//) operator meant for integral arithmetic operations as well as floating point arithmetic operations.
- The result of floor division(//) operator can be always floor value of either integer value or float value based on your arguments.
- If both arguments are 'int' type, then the result is 'int' type.
- If atleast one of the argument is float type, then the result is also float type.

In [43]:

```
print(10//2)
```

```
5
```

In [44]:

```
print(10/3)
```

```
3.3333333333333335
```

In [45]:

```
print(10.0/3)
```

```
3.3333333333333335
```

In [46]:

```
print(10.0//3)
```

```
3.0
```

In [47]:

```
print(10//3)
```

```
3
```

In [48]:

```
print(10.0//3.0)
```

```
3.0
```

In [49]:

```
print(20/2)
print(20.5/2)
print(20//2)
print(20.5//2)
print(30//2)
print(30.0//2)
```

```
10.0
10.25
10
10.0
15
15.0
```

vii. Power Operator or Exponential Operaor :

In [50]:

```
print(10**2) # meaning of this is 10 to the power 2
print(3**3)
```

```
100
27
```

2. Relational Operators (or) Comparison Operators:

Following are the relational operators usedin Python:

1. Less than (<)
2. Greater than (>)
3. Leass than or Equal to (<=)
4. Greater than or Equal to (>=)

i) We can apply relational operators for number types:

In [51]:

```
a = 10
b = 20
print('a < b is', a<b)
print('a <= b is', a<=b)
print('a > b is', a>b)
print('a >= b is', a>=b)
```

```
a < b is True
a <= b is True
a > b is False
a >= b is False
```

ii) We can apply relational operators for 'str' type also, here comparison is performed based on ASCII or Unicode values.

How to know the Unicode or ASCII value of any character?

- By using **ord()** function, we can get the ASCII value of any character.

In [52]:

```
print(ord('a'))
print(ord('A'))
```

```
97
65
```

- If you know the ASCII value and to find the corresponding character, you need to use the **chr()** function.

In [53]:

```
print(chr(97))
print(chr(65))
```

```
a
A
```

In [54]:

```
s1 = 'karthi' # ASCII value of 'a' is 97
s2 = 'sahasra' # ASCII value of 'b' is 98
print(s1<s2)
print(s1<=s2)
print(s1>s2)
print(s1>=s2)
```

```
True
True
False
False
```

In [55]:

```
s1 = 'karthi'
s2 = 'karthi'
print(s1<s2)
print(s1<=s2)
print(s1>s2)
print(s1>=s2)
```

```
False
True
False
True
```

In [56]:

```
s1 = 'karthi'
s2 = 'Karthi'
print(s1<s2)
print(s1<=s2)
print(s1>s2)
print(s1>=s2)
```

```
False
False
True
True
```

iii) We can apply relational operators even for boolean types also.

In [57]:

```
print(True > False)
print(True >= False) # True ==> 1
print(True < False) # False ==> 0
print(True <= False)
```

```
True
True
False
False
```

In [58]:

```
print(10 > 'karthi')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-58-e2ae37134b58> in <module>
----> 1 print(10 > 'karthi')

TypeError: '>' not supported between instances of 'int' and 'str'
```

In [60]:

```
a = 10
b = 20
if a>b:
    print('a is greater than b')
else:
    print('a is not greater than b')
```

```
a is not greater than b
```

iv) Chaining of relational operatrs:

- Chaining of relational operators is possible.
- In the chaining, if all comparisons returns True then only result is True.
- If atleast one comparison returns False then the result is False.

In [61]:

```
print(10<20)          # ==>True
print(10<20<30)       # ==>True
print(10<20<30<40)    # ==>True
print(10<20<30<40>50) # ==>False
```

```
True
True
True
False
```

3. Equality Operators:

Equality operators are used to check whether the given two values are equal or not. The following are the equality operators used in Python.

1. Equal to (==)
2. Not Equal to (!=)

In [62]:

```
print(10==20)
print(10!=20)
```

```
False
True
```

In [63]:

```
print(1==True)
print(10==10.0)
print('karthi'=='karthi')
```

```
True
True
True
```

We can apply these operators for any type, even for incompatible types also.

In [64]:

```
print(10=='karthi')
```

```
False
```

In [65]:

```
print(10=='10')
```

```
False
```

Note:

- Chaining concept is applicable for equality operators.
- If atleast one comparison returns False then the result is False. otherwise the result is True.

In [66]:

```
print(10==20==30==40)
print(10==10==10==10)
```

```
False
True
```

4. Logical operators:

Following are the various logical operators used in Python.

1. and
2. or
3. not

You can apply these operators for boolean types and non-boolean types, but the behavior is different.

For boolean types:

and ==> If both arguments are True then only result is True

or ==> If atleast one arugemnt is True then result is True

not ==> complement

i) 'and' Operator for boolean type:

- If both arguments are True then only result is True

In [67]:

```
print(True and True)
print(True and False)
print(False and True)
print(False and False)
```

```
True
False
False
False
```

ii) 'or' Operator for boolean type:

- If both arguments are True then only result is True.

In [68]:

```
print(True or True)
print(True or False)
print(False or True)
print(False or False)
```

```
True
True
True
False
```

iii) 'not' Operator for boolean type:

- Complement (or) Reverse

In [69]:

```
print(not True)
print(not False)
```

```
False
True
```

Eg:

Now we will try to develop a small authentication application with this knowledge.

- we will read user name and password from the keyboard.
- if the user name is karthi and password is sahasra, then that user is valid user otherwise invalid user.

In [70]:

```
userName = input('Enter User Name : ')
password = input('Enter Password : ')
if userName == 'karthi' and password == 'sahasra':
    print('valid User')
else:
    print('invalid user')
```

```
Enter User Name : karthi
Enter Password : sahasra
valid User
```

In [71]:

```
userName = input('Enter User Name : ')
password = input('Enter Password : ')
if userName == 'karthi' and password == 'sahasra':
    print('valid User')
else:
    print('invalid user')
```

```
Enter User Name : ECE
Enter Password : CSE
invalid user
```

For non-boolean types behaviour:

Note:

- 0 means False
- non-zero means True
- empty strings, list,tuple, set,dict is always treated as False

i) X and Y:

Here, X and Y are non boolean types and the result may be either X or Y but not boolean type (i.e., The result is always non boolean type only).

- if 'X' is evaluates to false then the result is 'X'.
- If 'X' is evaluates to true then the result is 'Y'.

In [72]:

```
print(10 and 20)
print(0 and 20)
print('karthi' and 'sahasra')
print('' and 'karthi') # first argument is empty string
print(' ' and 'karthi')
# first argument contains space character, so it is not empty
print('karthi' and '') # second argument is empty string
print('karthi' and ' ')
# second argument contains space character, so it is not empty
```

```
20
0
sahasra

karthi
```

ii) X or Y

Here, X and Y are non boolean types and the result may be either X or Y but not boolean type (i.e., The result is always non boolean type only).

- if 'X' is evaluates to true then the result is 'X'.
- If 'X' is evaluates to false then the result is 'Y'.

In [1]:

```
print(10 or 20)
print(0 or 20)
print('karthi' or 'sahasra')
print('' or 'karthi') # first argument is empty string
print(' ' or 'karthi')
# first argument contains space character, so it is not empty
print('karthi' or '') # second argument is empty string
print('karthi' or ' ')
```

```
10
20
karthi
karthi

karthi
karthi
```

iii) not X:

Even you apply not operator for non boolean type, the result is always boolean type only.

- If X is evalutates to False then result is True otherwise False.

In [2]:

```
print(not 'karthi')
print(not '')
print(not 0)
print(not 10)
```

```
False
True
True
False
```

5. Bitwise Operators:

- We can apply these operators bit by bit.
- These operators are applicable only for int and boolean types. By mistake if we are trying to apply for any other type then we will get Error.

Following are the various bitwise operators used in Python:

1. Bitwise and (&)
2. Bitwise or (|)
3. Bitwise ex-or (^)
4. Bitwise complement (~)
5. Bitwise leftshift Operator (<<)
6. Bitwise rightshift Operator(>>)

1. Bitwise and (&):

In [3]:

```
print(10.5 & 20.6)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-d0942894908d> in <module>
----> 1 print(10.5 & 20.6)

TypeError: unsupported operand type(s) for &: 'float' and 'float'
```

In [4]:

```
print('karthi' | 'karthi')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-482742ac27fc> in <module>
----> 1 print('karthi' | 'karthi')

TypeError: unsupported operand type(s) for |: 'str' and 'str'
```

In [5]:

```
print(bin(10))
print(bin(20))
print(10 & 20) # Valid
print(10.0 & 20.0) # In valid
```

```
0b1010
0b10100
0

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-2449e0efc92e> in <module>
      2 print(bin(20))
      3 print(10 & 20) # Valid
----> 4 print(10.0 & 20.0) # In valid

TypeError: unsupported operand type(s) for &: 'float' and 'float'
```

In [6]:

```
print(True & False)
```

```
False
```

2. Bitwise or (|):

In [7]:

```
print(True | False)
```

```
True
```

3. Bitwise ex-or (^):

In [9]:

```
print(2^4)
```

```
6
```

Behavior of Bitwise Operators:

- & ==> If both bits are 1 then only result is 1 otherwise result is 0
- | ==> If atleast one bit is 1 then result is 1 otherwise result is 0
- ^ ==>If bits are different then only result is 1 otherwise result is 0
- ~ ==> bitwise complement operator, i.e 1 means 0 and 0 means 1
- << ==> Bitwise Left shift Operataor
- Bitwise Right Shift Operator ==> >>

In [8]:

```
print(4 & 5) # 100 & 101
print(4 | 5) # 100 | 101
print(4 ^ 5) # 100 ^ 101
```

```
4
5
1
```

Bitwise Complement Operator (~):

- We have to apply complement for total bits.

In [10]:

```
print(~4) # 4 ==> 100
```

```
-5
```

Here, we have to apply complement for total bits, not for three bits (in case of 4). In Python minimum 28 bits required to represent an integer.

Note:

- The most significant bit acts as sign bit. 0 value represents +ve number where as 1 represents -ve value.
- Positive numbers will be repesented directly in the memory where as Negative numbers will be represented indirectly in 2's complement form.

How you can find two's complement of a number?

- To find Two's complement of a number, first you need to find One's complement of that number and add 1 to it.
- One's complement ==> Interchange of 0's and 1's

In [11]:

```
print(~5)
```

```
-6
```

In [12]:

```
print(~-4) # negative values are stored in the memory in 2's complement form.
```

```
3
```

6. Shift Operators:

Following are the various shift operators used in Python:

1. Left Shift Operator (<<)
2. Right Shift Operator (<<)

1. Left Shift Operator (<<):

- After shifting the bits from left side, empty cells to be filled with zero.

In [13]:

```
print(10<<2)
```

```
40
```

2. Right Shift Operator (<<):

- After shifting the empty cells we have to fill with sign bit.(0 for +ve and 1 for -ve).

In [14]:

```
print(10>>2)
```

```
2
```

We can apply bitwise operators for boolean types also.

In [15]:

```
print(True & False)
print(True | False)
print(True ^ False)
print(~True)
print(~False)
print(True<<2)
print(True>>2)
```

```
False
True
True
-2
-1
4
0
```

7. Assignment Operators:

- We can use assignment operator to assign value to the variable.

In [ ]:

```
x = 2
```

We can combine asignment operator with some other operator to form **compound assignment operator**.

Eg :

x+=10 ====> x = x+10

In [16]:

```
x = 10
x += 20 # x = x + 20
print(x)
```

```
30
```

The following is the list of all possible compound assignment operators in Python:

+=

-=

*=

/=

%=

//=

**=

&=

|=

^=

<<= and >>=

In [17]:

```
x = 10 # 1010
x &= 5 # 0101
print(x)
```

```
0
```

In [18]:

```
x = 10
x **= 2 # x = x**2
print(x)
```

```
100
```

In Python increment/decrement operators concept is not there.

Let us see the following code

In [19]:

```
x = 10
print(++x)
print(++++x)
# Here, + and - are sign bits, not increment and decrement operators
print(-x)
print(--x)
print(+++++++++++-x)
print(-------------+x)
```

```
10
10
-10
10
-10
-10
```

8. Ternary Operator (or) Conditional Operator

1. If the operator operates on only one operand, we will call such operator as unary operator. For eg:, ~a.
2. If the operator operates on Two operands, we will call such operator as binary operator. For eg:, a + b.
3. If the operator operates on Three operands, we will call such operator as Ternary operator.

Syntax:

```
x = firstValue if condition else secondValue
```

- If condition is True then firstValue will be considered else secondValue will be considered.

In [1]:

```
a,b=23,43 # a =23 b = 43
c = 50 if a>b else 100
print(c)
```

```
100
```

Read two integer numbers from the keyboard and print minimum value using ternary operator.

In [2]:

```
x =int(input("Enter First Number:"))
y =int(input("Enter Second Number:"))
min=x if x<y else y
print("Minimum Value:",min)
```

```
Enter First Number:255
Enter Second Number:22
Minimum Value: 22
```

In [3]:

```
x =int(input("Enter First Number:"))
y =int(input("Enter Second Number:"))
min=x if x<y else y
print("Minimum Value:",min)
```

```
Enter First Number:22
Enter Second Number:255
Minimum Value: 22
```

Program for finding minimum of 3 numbers using nesting of ternary operators.

In [4]:

```
a=int(input("Enter First Number:"))
b=int(input("Enter Second Number:"))
c=int(input("Enter Third Number:"))
min= a if a<b and a<c else b if b<c else c
print("Minimum Value:",min)
```

```
Enter First Number:101
Enter Second Number:201
Enter Third Number:301
Minimum Value: 101
```

In [5]:

```
a=int(input("Enter First Number:"))
b=int(input("Enter Second Number:"))
c=int(input("Enter Third Number:"))
min= a if a<b and a<c else b if b<c else c
print("Minimum Value:",min)
```

```
Enter First Number:-10
Enter Second Number:-20
Enter Third Number:-30
Minimum Value: -30
```

Python Program for finding maximum of 3 numbers.

In [6]:

```
a=int(input("Enter First Number:"))
b=int(input("Enter Second Number:"))
c=int(input("Enter Third Number:"))
max=a if a>b and a>c else b if b>c else c
print("Maximum Value:",max)
```

```
Enter First Number:33
Enter Second Number:22
Enter Third Number:44
Maximum Value: 44
```

Assume that there are two numbers, x and y, whose values to be read from the keyboard, and print the following outputs based on the values of x and y.

- case 1: If both are equal, then the output is : Both numbers are equal
- case 2: If first number is smaller than second one, then the output is: First Number is Less than Second Number
- case 3: If the firts number is greater than second number, then the output is : First Number Greater than Second Number

In [7]:

```
a=int(input("Enter First Number:"))
b=int(input("Enter Second Number:"))
print("Both numbers are equal" if a==b
      else "First Number is Less than Second Number"
if a<b else "First Number Greater than Second Number")
```

```
Enter First Number:10
Enter Second Number:10
Both numbers are equal
```

In [8]:

```
a=int(input("Enter First Number:"))
b=int(input("Enter Second Number:"))
print("Both numbers are equal" if a==b
      else "First Number is Less than Second Number"
if a<b else "First Number Greater than Second Number")
```

```
Enter First Number:10
Enter Second Number:20
First Number is Less than Second Number
```

In [9]:

```
a=int(input("Enter First Number:"))
b=int(input("Enter Second Number:"))
print("Both numbers are equal" if a==b
      else "First Number is Less than Second Number"
if a<b else "First Number Greater than Second Number")
```

```
Enter First Number:20
Enter Second Number:10
First Number Greater than Second Number
```

9. Special Operators:

There are two types of special operators are there in Python:

1. Identity Operators
2. Membership Operators

1. Identity Operators:

We can use identity operators for address comparison. There are two identity operators used in Python:

i) is

ii) is not

- r1 is r2 returns True if both r1 and r2 are pointing to the same object.
- r1 is not r2 returns True if both r1 and r2 are not pointing to the same object.

In [1]:

```
a=10
b=10
print(a is b)
x=True
y=True
print( x is y)
```

```
True
True
```

In [2]:

```
a="karthi"
b="karthi"
print(id(a))
print(id(b))
print(a is b)
```

```
1509178647664
1509178647664
True
```

In [3]:

```
list1=["one","two","three"]
list2=["one","two","three"]
print(id(list1))
print(id(list2))
print(list1 is list2)
print(list1 is not list2) # reference comaprison (is & is not)
print(list1 == list2) # content comparison (==)
```

```
1509178190144
1509178196736
False
True
True
```

Note:

- We can use is operator for address comparison where as == operator for content comparison.

2. Membership Operators

We can use Membership operators to check whether the given object present in the given collection.(It may be String,List,Set,Tuple or Dict)

There are two types of membership operators used in Python:

i) in

ii) not in

- in returns True if the given object present in the specified Collection.
- not in retruns True if the given object not present in the specified Collection.

In [4]:

```
x="hello learning Python is very easy!!!"
print('h' in x)
print('d' in x)
print('d' not in x)
print('python' in x) # case sensitivity
print('Python' in x)
```

```
True
False
True
False
True
```

In [5]:

```
list1=["sunny","bunny","chinny","pinny"]
print("sunny" in list1)
print("tunny" in list1)
print("tunny" not in list1)
```

```
True
False
True
```

10. Operator Precedence:

- If multiple operators present then which operator will be evaluated first is decided by operator precedence.

In [6]:

```
print(3+10*2)
print((3+10)*2)
```

```
23
26
```

The following list describes operator precedence in Python:

() ==> Parenthesis

** ==> exponential operator

~,- ==> Bitwise complement operator,unary minus operator

*,/,%,// ==> multiplication,division,modulo,floor division

+,- ==> addition,subtraction

<<,>> ==> Left and Right Shift

& ==> bitwise And

^ ==> Bitwise X-OR

| ==> Bitwise OR

<,<=,>,>=,==, != ==> Relational or Comparison operators

=,+=,-=,*=... ==> Assignment operators

is , is not ==> Identity Operators

in , not in ==> Membership operators

not ==> Logical not

and ==> Logical and

or ==> Logical or

In [7]:

```
a=30
b=20
c=10
d=5
print((a+b)*c/d)
# division operoator in Python always going to provide float value as result
print((a+b)*(c/d))
print(a+(b*c)/d)
```

```
100.0
100.0
70.0
```

In [8]:

```
print(3/2*4+3+(10/5)**3-2)
print(3/2*4+3+2.0**3-2)
print(3/2*4+3+8.0-2)
print(1.5*4+3+8.0-2)
print(6.0+3+8.0-2)
```

```
15.0
15.0
15.0
15.0
15.0
```

Experiment 3:

a) Write Python programs to demonstrate the following:

```
i) input( )          ii) print( )      iii) ‘sep’ attribute
iv) ‘end’ attribute v) replacement Operator ({ })
```

i. input():

- This function always reads the data from the keyboard in the form of String Format. We have to convert that string type to our required type by using the corresponding type casting methods.

In [1]:

```
type(input("Enter value:"))
```

```
Enter value:10
```

Out[1]:

```
str
```

In [2]:

```
type(input("Enter value:"))
```

```
Enter value:22.7
```

Out[2]:

```
str
```

In [3]:

```
type(input("Enter value:"))
```

```
Enter value:True
```

Out[3]:

```
str
```

In [4]:

```
type(input("Enter value:"))
```

```
Enter value:'SMVDU'
```

Out[4]:

```
str
```

Note:

Why input() in Python 3 gave the priority for string type as return type?

Reason: The most commonly used type in any programming language is str type , that's why they gave the priority for str type as default return type of input() function.

Demo Program 1: Read input data from the Keyboard

In [5]:

```
x=input("Enter First Number:")
y=input("Enter Second Number:")
i = int(x)
j = int(y)
print("The Sum:",i+j)
```

```
Enter First Number:1000
Enter Second Number:2000
The Sum: 3000
```

Above code in simplified form:

In [6]:

```
x=int(input("Enter First Number:"))
y=int(input("Enter Second Number:"))
print("The Sum:",x+y)
```

```
Enter First Number:1000
Enter Second Number:2000
The Sum: 3000
```

We can write the above code in single line also.

In [7]:

```
print("The Sum:",int(input("Enter First Number:"))
      +int(input("Enter Second Number:")))
```

```
Enter First Number:1000
Enter Second Number:2000
The Sum: 3000
```

Demo Program 2: Write a program to read Employee data from the keyboard and print that data.

In [8]:

```
eno=int(input("Enter Employee No:"))
ename=input("Enter Employee Name:")
esal=float(input("Enter Employee Salary:"))
eaddr=input("Enter Employee Address:")
married=bool(input("Employee Married ?[True|False]:"))
print("Please Confirm your provided Information")
print("Employee No :",eno)
print("Employee Name :",ename)
print("Employee Salary :",esal)
print("Employee Address :",eaddr)
print("Employee Married ? :",married)
```

```
Enter Employee No:11111
Enter Employee Name:Karthikeya
Enter Employee Salary:100000
Enter Employee Address:Nandyal
Employee Married ?[True|False]:T
Please Confirm your provided Information
Employee No : 11111
Employee Name : Karthikeya
Employee Salary : 100000.0
Employee Address : Nandyal
Employee Married ? : True
```

In [9]:

```
eno=int(input("Enter Employee No:"))
ename=input("Enter Employee Name:")
esal=float(input("Enter Employee Salary:"))
eaddr=input("Enter Employee Address:")
married=bool(input("Employee Married ?[True|False]:"))
print("Please Confirm your provided Information")
print("Employee No :",eno)
print("Employee Name :",ename)
print("Employee Salary :",esal)
print("Employee Address :",eaddr)
print("Employee Married ? :",married)
```

```
Enter Employee No:11111
Enter Employee Name:Karthikeya
Enter Employee Salary:100000
Enter Employee Address:Nandyal
Employee Married ?[True|False]:False
Please Confirm your provided Information
Employee No : 11111
Employee Name : Karthikeya
Employee Salary : 100000.0
Employee Address : Nandyal
Employee Married ? : True
```

In [10]:

```
eno=int(input("Enter Employee No:"))
ename=input("Enter Employee Name:")
esal=float(input("Enter Employee Salary:"))
eaddr=input("Enter Employee Address:")
married=bool(input("Employee Married ?[True|False]:"))
print("Please Confirm your provided Information")
print("Employee No :",eno)
print("Employee Name :",ename)
print("Employee Salary :",esal)
print("Employee Address :",eaddr)
print("Employee Married ? :",married)
```

```
Enter Employee No:11111
Enter Employee Name:Karthikeya
Enter Employee Salary:100000
Enter Employee Address:Nandyal
Employee Married ?[True|False]:
Please Confirm your provided Information
Employee No : 11111
Employee Name : Karthikeya
Employee Salary : 100000.0
Employee Address : Nandyal
Employee Married ? : False
```

When you are not providing any value to the married (Just press Enter), then only it considers empty string and gives the False value. In the above example, to read the boolean data, we need to follow the above process.

But it is not our logic requirement. If you want to convert string to Boolean type, instead of using bool() function we need to use **eval()** function.

In [11]:

```
eno=int(input("Enter Employee No:"))
ename=input("Enter Employee Name:")
esal=float(input("Enter Employee Salary:"))
eaddr=input("Enter Employee Address:")
married=eval(input("Employee Married ?[True|False]:"))
print("Please Confirm your provided Information")
print("Employee No :",eno)
print("Employee Name :",ename)
print("Employee Salary :",esal)
print("Employee Address :",eaddr)
print("Employee Married ? :",married)
```

```
Enter Employee No:11111
Enter Employee Name:Karthikeya
Enter Employee Salary:100000
Enter Employee Address:Nandyal
Employee Married ?[True|False]:False
Please Confirm your provided Information
Employee No : 11111
Employee Name : Karthikeya
Employee Salary : 100000.0
Employee Address : Nandyal
Employee Married ? : False
```

eval() Function:

- eval() Function is a single function which is the replacement of all the typecasting functions in Python.

In [12]:

```
x = (input('Enter Something : '))
print(type(x))
```

```
Enter Something : 10
<class 'str'>
```

In [13]:

```
x = (input('Enter Something :'))
print(type(x))
```

```
Enter Something :33.3
<class 'str'>
```

In [15]:

```
x = eval((input('Enter Something : ')))
print(type(x))
```

```
Enter Something : 'Nandyal'
<class 'str'>
```

In [16]:

```
x = eval((input('Enter Something : ')))
print(type(x))
```

```
Enter Something : 10
<class 'int'>
```

In [17]:

```
x = eval((input('Enter Something : ')))
print(type(x))
```

```
Enter Something : 33.3
<class 'float'>
```

In [18]:

```
x = eval((input('Enter Something : ')))
print(type(x))
```

```
Enter Something : 'Nandyal'
<class 'str'>
```

In [19]:

```
x = eval((input('Enter Something : ')))
print(type(x))
```

```
Enter Something : [1,2,3]
<class 'list'>
```

In [20]:

```
x = eval((input('Enter Something : ')))
print(type(x))
```

```
Enter Something : (1,2,3)
<class 'tuple'>
```

In [21]:

```
x = eval((input('Enter Something : ')))
print(type(x))
```

```
Enter Something : (10)
<class 'int'>
```

In [22]:

```
x = eval((input('Enter Something : ')))
print(type(x))
```

```
Enter Something : (1,)
<class 'tuple'>
```

ii.print():

- We can use print() function to display output to the console for end user sake.
- Multiple forms are there related to print() function.

Form-1: print() without any argument

- Just it prints new line character (i.e.,\n)

In [1]:

```
print('karthi')
print() # prints new line character
print('sahasra')
```

```
karthi

sahasra
```

see the difference in below code:

In [2]:

```
print('karthi')
#print() # prints new line character
print('sahasra')
```

```
karthi
sahasra
```

Form-2: print() function to print of string argument

In [3]:

```
print("Hello World")
```

```
Hello World
```

We can use escape characters also.

In [4]:

```
print("Hello \n World")
print("Hello\tWorld")
```

```
Hello
 World
Hello	World
```

- We can use repetetion operator (*) in the string.

In [6]:

```
print(10*"RGM")
print("RGM"*10)
```

```
RGMRGMRGMRGMRGMRGMRGMRGMRGMRGMRGM
RGMRGMRGMRGMRGMRGMRGMRGMRGMRGMRGM
```

- We can use + operator also.

In [7]:

```
print("RGM"+"xyz")
```

```
RGMxyz
```

Note:

- If both arguments are string type then + operator acts as concatenation operator.
- If one argument is string type and second is any other type like int then we will get Error
- If both arguments are number type then + operator acts as arithmetic addition operator.

Form-3: print() with variable number of arguments:

In [8]:

```
a,b,c=10,20,30
print("The Values are :",a,b,c)
# here, we are passing 4 arguments to the print function.
```

```
The Values are : 10 20 30
```

iii. 'sep' attribute:

Form-4: print() with 'sep' attribute:

- By default output values are seperated by space. If we want we can specify seperator by using "sep" attribute.
- 'sep' means seperator.

In [9]:

```
a,b,c=10,20,30
print(a,b,c) # 10 20 30
print(a,b,c,sep=',') # 10,20,30
print(a,b,c,sep=':') # 10:20:30
print(a,b,c,sep='-') # 10-20-30
```

```
10 20 30
10,20,30
10:20:30
10-20-30
```

iv. 'end' attribute:

Form-5: print() with 'end' attribute:

In [10]:

```
print("Hello")
print("Karthi")
print("Sahasra")
```

```
Hello
Karthi
Sahasra
```

- If we want output in the same line with space, we need to use end attribute.
- default value of 'end' attribute is newline character. (That means, if there is no end attribute, automatically newline character will be printed).

In [11]:

```
print("Hello",end=' ')
print("Karthi",end=' ') # if end is space character
print("Sahasra")
```

```
Hello Karthi Sahasra
```

In [12]:

```
print("Hello",end='')
print("Karthi",end='') # if end is nothing
print("Sahasra")
```

```
HelloKarthiSahasra
```

In [13]:

```
print('hello',end = '::')
print('karthi',end = '****')
print('sahasra')
```

```
hello::karthi****sahasra
```

Eg: Program to demonstrate both 'sep' and 'end' attributes.

In [14]:

```
print(10,20,30,sep = ':', end = '***')
print(40,50,60,sep = ':') # default value of 'end' attribute is '\n'
print(70,80,sep = '**',end = '$$')
print(90,100)
```

```
10:20:30***40:50:60
70**80$$90 100
```

Eg : Consider the following case,

In [15]:

```
print('karthi' + 'sahasra') # Concatanation
print('karthi','sahasra') # ',' means space is the seperator
print(10,20,30)
```

```
karthisahasra
karthi sahasra
10 20 30
```

Form-6: print(object) statement:

- We can pass any object (like list,tuple,set etc)as argument to the print() statement.

In [16]:

```
l=[10,20,30,40]
t=(10,20,30,40)
print(l)
print(t)
```

```
[10, 20, 30, 40]
(10, 20, 30, 40)
```

Form-7: print(String,variable list):

- We can use print() statement with String and any number of arguments.

In [17]:

```
s="Karthi"
a=6
s1="java"
s2="Python"
print("Hello",s,"Your Age is",a)
print("You are learning",s1,"and",s2)
```

```
Hello Karthi Your Age is 6
You are learning java and Python
```

Form-8: print(formatted string):

%i ====>int

%d ====>int

%f =====>float

%s ======>String type

Syntax:

```
print("formatted string" %(variable list))
```

In [18]:

```
a=10
b=20
c=30
print("a value is %i" %a)
print("b value is %d and c value is %d" %(b,c))
```

```
a value is 10
b value is 20 and c value is 30
```

In [19]:

```
s="Karthi"
list=[10,20,30,40]
print("Hello %s ...The List of Items are %s" %(s,list))
```

```
Hello Karthi ...The List of Items are [10, 20, 30, 40]
```

In [20]:

```
price = 70.56789
print('Price value = {}'.format(price))
print('Price value = %f'%price)
print('Price value = %.2f'%price)
```

```
Price value = 70.56789
Price value = 70.567890
Price value = 70.57
```

v. Replacement Operator ({}):

Form-9: print() with replacement operator { }

In [21]:

```
name = "Karthi"
salary = 100000
sister = "Sahasra"
print("Hello {} your salary is {} and Your Sister {} is waiting"
      .format(name,salary,sister))
print("Hello {0} your salary is {1} and Your Sister {2} is waiting"
      .format(name,salary,sister))
print("Hello {1} your salary is {2} and Your Sister {0} is waiting"
      .format(name,salary,sister))
print("Hello {2} your salary is {0} and Your Sister {1} is waiting"
      .format(salary,sister,name))
print("Hello {x} your salary is {y} and Your Sister {z} is waiting"
      .format(x=name,y=salary,z=sister))
```

```
Hello Karthi your salary is 100000 and Your Sister Sahasra is waiting
Hello Karthi your salary is 100000 and Your Sister Sahasra is waiting
Hello 100000 your salary is Sahasra and Your Sister Karthi is waiting
Hello Karthi your salary is 100000 and Your Sister Sahasra is waiting
Hello Karthi your salary is 100000 and Your Sister Sahasra is waiting
```

In [22]:

```
a,b,c,d = 10,20,30,40 # print a=10,b=20,c=30,d=40
print('a = {},b = {},c = {},d = {}'.format(a,b,c,d))
```

```
a = 10,b = 20,c = 30,d = 40
```

b) Demonstrate the following Conditional statements in Python with suitable examples.

```
i) if statement            ii) if  else statement
iii) if – elif – else statement
```

i) if statement:

Syntax:

if condition:

```
    statement 1

    statement 2

    statement 3
```

statement

In [23]:

```
if 10<20:
    print('10 is less than 20')
print('End of Program')
```

```
10 is less than 20
End of Program
```

In [24]:

```
if 10<20:
print('10 is less than 20')
print('End of Program')
```

```
  File "<ipython-input-24-f2d3b9a6180e>", line 2
    print('10 is less than 20')
    ^
IndentationError: expected an indented block
```

In [25]:

```
name=input("Enter Name:")
if name=="Karthi":
    print("Hello Karthi Good Morning")
print("How are you!!!")
```

```
Enter Name:Karthi
Hello Karthi Good Morning
How are you!!!
```

In [26]:

```
name=input("Enter Name:")
if name=="Karthi":
    print("Hello Karthi Good Morning")
print("How are you!!!")
```

```
Enter Name:Sourav
How are you!!!
```

ii) if else statement:

Syntax:

if condition:

```
    Action 1
```

else:

```
    Action 2
```

- if condition is true then Action-1 will be executed otherwise Action-2 will be executed.

In [27]:

```
name = input('Enter Name : ')
if name == 'Karthi':
    print('Hello Karthi! Good Morning')
else:
    print('Hello Guest! Good Morning')
print('How are you?')
```

```
Enter Name : Karthi
Hello Karthi! Good Morning
How are you?
```

In [29]:

```
name = input('Enter Name : ')
if name == 'Karthi':
    print('Hello Karthi! Good Morning')
else:
    print('Hello Guest! Good Morning')
print('How are you?')
```

```
Enter Name : sourav
Hello Guest! Good Morning
How are you?
```

iii) if – elif – else statement :

Syntax:

```
if condition1:
    Action-1
elif condition2:
    Action-2
elif condition3:
    Action-3
elif condition4:
    Action-4
...
else:
    Default Action
```

Based condition the corresponding action will be executed.

In [30]:

```
brand=input("Enter Your Favourite Brand:")
if brand=="RC":
    print("It is childrens brand")
elif brand=="KF":
    print("It is not that much kick")
elif brand=="FO":
    print("Buy one get Free One")
else :
    print("Other Brands are not recommended")
```

```
Enter Your Favourite Brand:RC
It is childrens brand
```

In [31]:

```
brand=input("Enter Your Favourite Brand:")
if brand=="RC":
    print("It is childrens brand")
elif brand=="KF":
    print("It is not that much kick")
elif brand=="FO":
    print("Buy one get Free One")
else :
    print("Other Brands are not recommended")
```

```
Enter Your Favourite Brand:FO
Buy one get Free One
```

In [32]:

```
brand=input("Enter Your Favourite Brand:")
if brand=="RC":
    print("It is childrens brand")
elif brand=="KF":
    print("It is not that much kick")
elif brand=="FO":
    print("Buy one get Free One")
else :
    print("Other Brands are not recommended")
```

```
Enter Your Favourite Brand:ABC
Other Brands are not recommended
```

Points to Ponder:

1. else part is always optional.
2. There is no switch statement in Python.

c) Demonstrate the following Iterative statements in Python with suitable examples.

```
i) while loop          ii) for loop
```

i) while loop:

- If we want to execute a group of statements iteratively until some condition false,then we should go for while loop.

Syntax:

while condition:

    body

Eg 1: Write a Python program to print numbers from 1 to 10 by using while loop.

In [13]:

```
x=1
while x <=10:
    print(x,end=' ')
    x=x+1
```

```
1 2 3 4 5 6 7 8 9 10
```

Eg 2: Write a Python program to display the sum of first 'n' numbers.

In [14]:

```
n=int(input("Enter number:"))
sum=0
i=1
while i<=n:
    sum=sum+i
    i=i+1
print("The sum of first",n,"numbers is :",sum)
```

```
Enter number:10
The sum of first 10 numbers is : 55
```

Eg 3: write a program to prompt user to enter some name until entering RGM.

In [15]:

```
name=""
while name!="RGM":
    name=input("Enter Name:")
print("Thanks for confirmation")
```

```
Enter Name:SREC
Enter Name:GPR
Enter Name:KSRM
Enter Name:AITS
Enter Name:RGM
Thanks for confirmation
```

ii) for loop:

- If we want to execute some action for every element present in some sequence (it may be string or collection) then we should go for for loop.

Syntax:

for x in sequence:

```
    body
```

- Where, 'sequence' can be string or any collection.
- Body will be executed for every element present in the sequence.

Eg 1: Write a Python Program to print characters present in the given string.

In [1]:

```
s="Sahasra"
for x in s :
    print(x)
```

```
S
a
h
a
s
r
a
```

In [3]:

```
s="Sahasra"
for x in s :
    print(x,end='\t')
```

```
S	a	h	a	s	r	a
```

Eg 2: To print characters present in string index wise.

In [4]:

```
s=input("Enter some String: ")
i=0
for x in s :
    print("The character present at ",i,"index is :",x)
    i=i+1
```

```
Enter some String: Karthikeya
The character present at  0 index is : K
The character present at  1 index is : a
The character present at  2 index is : r
The character present at  3 index is : t
The character present at  4 index is : h
The character present at  5 index is : i
The character present at  6 index is : k
The character present at  7 index is : e
The character present at  8 index is : y
The character present at  9 index is : a
```

Eg 3: Write a Python program to print Hello 10 times.

In [5]:

```
s = 'Hello'
for i in range(1,11):
    print(s)
```

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

In [7]:

```
s = 'Hello'
for i in range(10):
    print(s)
```

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

Eg 4: Write a Python program to display numbers from 0 to 10.

In [8]:

```
for i in range(0,11):
     print(i,end=' ')
```

```
0 1 2 3 4 5 6 7 8 9 10
```

Eg 5: Write a Python program to display odd numbers from 0 to 20.

In [9]:

```
for i in range(21):
    if(i%2!=0):
        print(i,end=' ')
```

```
1 3 5 7 9 11 13 15 17 19
```

Eg 6: Write a Python Program to display numbers from 10 to 1 in descending order.

In [10]:

```
for i in range(10,0,-1):
    print(i,end=' ')
```

```
10 9 8 7 6 5 4 3 2 1
```

Eg 7: Write a Python program to print sum of numbers presenst inside list.

In [11]:

```
list=eval(input("Enter List:"))
sum=0;
for x in list:
    sum=sum+x;
print("The Sum=",sum)
```

```
Enter List:10,20,30,40
The Sum= 100
```

In [12]:

```
list=eval(input("Enter List:"))
sum=0;
for x in list:
    sum=sum+ x;
print("The Sum=",sum)
```

```
Enter List:[10,20,30,40]
The Sum= 100
```

Nested Loops:

- Sometimes we can take a loop inside another loop,which are also known as nested loops.

Eg 1:

In [16]:

```
for i in range(3):
    for j in range(2):
        print('Hello')
```

```
Hello
Hello
Hello
Hello
Hello
Hello
```

Eg 2:

In [17]:

```
for i in range(4):
    for j in range(4):
        print('i = {} j = {}'.format(i,j))
```

```
i = 0 j = 0
i = 0 j = 1
i = 0 j = 2
i = 0 j = 3
i = 1 j = 0
i = 1 j = 1
i = 1 j = 2
i = 1 j = 3
i = 2 j = 0
i = 2 j = 1
i = 2 j = 2
i = 2 j = 3
i = 3 j = 0
i = 3 j = 1
i = 3 j = 2
i = 3 j = 3
```

Write Python Programs to display the below patterns.

Pattern-1:

```
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
```

In [21]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print("* "*n)
```

```
Enter the number of rows: 10
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
```

Pattern-2:

```
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9
10 10 10 10 10 10
```

In [23]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+1):
        print(i,end=" ")
    print()
```

```
Enter the number of rows: 7
1 1 1 1 1 1 1
2 2 2 2 2 2 2
3 3 3 3 3 3 3
4 4 4 4 4 4 4
5 5 5 5 5 5 5
6 6 6 6 6 6 6
7 7 7 7 7 7 7
```

Pattern-3:

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

In [24]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+1):
        print(j,end=" ")
    print()
```

```
Enter the number of rows: 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

Pattern-4:

A A A A A A A A A A
B B B B B B B B B B
C C C C C C C C C C
D D D D D D D D D D
E E E E E E E E E E
F F F F F F F F F F
G G G G G G G G G G
H H H H H H H H H H
I I I I I I I I I
J J J J J J J J J J

In [25]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+1):
        print(chr(64+i),end=" ")
    print()
```

```
Enter the number of rows: 10
A A A A A A A A A A
B B B B B B B B B B
C C C C C C C C C C
D D D D D D D D D D
E E E E E E E E E E
F F F F F F F F F F
G G G G G G G G G G
H H H H H H H H H H
I I I I I I I I I I
J J J J J J J J J J
```

Pattern-5:

```
A B C D E F G H I J
A B C D E F G H I J
A B C D E F G H I J
A B C D E F G H I J
A B C D E F G H I J
A B C D E F G H I J
A B C D E F G H I J
A B C D E F G H I J
A B C D E F G H I J
A B C D E F G H I J
```

In [26]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+1):
        print(chr(64+j),end=" ")
    print()
```

```
Enter the number of rows: 7
A B C D E F G
A B C D E F G
A B C D E F G
A B C D E F G
A B C D E F G
A B C D E F G
A B C D E F G
```

Pattern-6:

```
10 10 10 10 10 10 10 10 10 10
9 9 9 9 9 9 9 9 9 9
8 8 8 8 8 8 8 8 8 8
7 7 7 7 7 7 7 7 7 7
6 6 6 6 6 6 6 6 6 6
5 5 5 5 5 5 5 5 5 5
4 4 4 4 4 4 4 4 4 4
3 3 3 3 3 3 3 3 3 3
2 2 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 1 1 1
```

n=int(input("Enter the number of rows: ")) for i in range(1,n+1): for j in range(1,n+1): print(n+1-i,end=" ")
print()

Pattern-6:

10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1

In [28]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+1):
        print(n+1-j,end=" ")
    print()
```

```
Enter the number of rows: 10
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
```

Pattern-7:

J J J J J J J J J J
I I I I I I I I I
H H H H H H H H H H
G G G G G G G G G G
F F F F F F F F F F
E E E E E E E E E E
D D D D D D D D D D
C C C C C C C C C C
B B B B B B B B B B
A A A A A A A A A A

In [29]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+1):
        print(chr(65+n-i),end=" ")
    print()
```

```
Enter the number of rows: 10
J J J J J J J J J J
I I I I I I I I I I
H H H H H H H H H H
G G G G G G G G G G
F F F F F F F F F F
E E E E E E E E E E
D D D D D D D D D D
C C C C C C C C C C
B B B B B B B B B B
A A A A A A A A A A
```

Pattern-8:

J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A

In [30]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+1):
        print(chr(65+n-j),end=" ")
    print()
```

```
Enter the number of rows: 10
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
J I H G F E D C B A
```

Pattern-9:

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
```

In [31]:

```
n=int(input("Enter the number of rows:"))
for i in range(1,n+1):
    for j in range(1,i+1):
        print("*",end=" ")
    print()
```

```
Enter the number of rows:5
*
* *
* * *
* * * *
* * * * *
```

Alternative Way:

In [32]:

```
n=int(input("Enter the number of rows:"))
for i in range(1,n+1):
    print("* "*i)
```

```
Enter the number of rows:5
*
* *
* * *
* * * *
* * * * *
```

Pattern-10:

1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
7 7 7 7 7 7 7
8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9
10 10 10 10 10 10 10 10 10 10

In [34]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,i+1):
        print(i,end=" ")
    print()
```

```
Enter the number of rows: 9
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
7 7 7 7 7 7 7
8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9
```

In [36]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,i+1):
        print(i,end="\t")
    print()
```

```
Enter the number of rows: 10
1
2       2
3       3       3
4       4       4       4
5       5       5       5       5
6       6       6       6       6       6
7       7       7       7       7       7       7
8       8       8       8       8       8       8       8
9       9       9       9       9       9       9       9       9
10      10      10      10      10      10      10      10      10      10
```

Pattern-11:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
```

In [37]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,i+1):
        print(j,end=" ")
    print()
```

```
Enter the number of rows: 10
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
```

Pattern-12:

```
A
B B
C C C
D D D D
E E E E E
F F F F F F
G G G G G G G
H H H H H H H H
I I I I I I I I I
J J J J J J J J J J
```

In [38]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,i+1):
        print(chr(64+i),end=" ")
    print()
```

```
Enter the number of rows: 10
A
B B
C C C
D D D D
E E E E E
F F F F F F
G G G G G G G
H H H H H H H H
I I I I I I I I I
J J J J J J J J J J
```

Pattern-13:

```
A
A B
A B C
A B C D
A B C D E
A B C D E F
A B C D E F G
A B C D E F G H
A B C D E F G H I
A B C D E F G H I J
```

In [39]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,i+1):
        print(chr(64+j),end=" ")
    print()
```

```
Enter the number of rows: 10
A
A B
A B C
A B C D
A B C D E
A B C D E F
A B C D E F G
A B C D E F G H
A B C D E F G H I
A B C D E F G H I J
```

Pattern-14:

```
* * * * * * * * * *
* * * * * * * * *
* * * * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

In [40]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+2-i):
        print("*",end=" ")
    print()
```

```
Enter the number of rows: 10
* * * * * * * * * *
* * * * * * * * *
* * * * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

Pattern-15:

```
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3
4 4 4 4 4 4 4
5 5 5 5 5 5
6 6 6 6 6
7 7 7 7
8 8 8
9 9
10
```

In [41]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+2-i):
        print(i,end=" ")
    print()
```

```
Enter the number of rows: 10
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3
4 4 4 4 4 4 4
5 5 5 5 5 5
6 6 6 6 6
7 7 7 7
8 8 8
9 9
10
```

Pattern-16:

```
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

In [42]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+2-i):
        print(j,end=" ")
    print()
```

```
Enter the number of rows: 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Pattern-17:

```
A A A A A A A A A A
B B B B B B B B B
C C C C C C C C
D D D D D D D
E E E E E E
F F F F F
G G G G
H H H
I I
J
```

In [43]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+2-i):
        print(chr(64+i),end=" ")
    print()
```

```
Enter the number of rows: 10
A A A A A A A A A A
B B B B B B B B B
C C C C C C C C
D D D D D D D
E E E E E E
F F F F F
G G G G
H H H
I I
J
```

Pattern-18:

```
A B C D E F G H I J
A B C D E F G H I
A B C D E F G H
A B C D E F G
A B C D E F
A B C D E
A B C D
```

In [44]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+2-i):
        print(chr(64+j),end=" ")
    print()
```

```
Enter the number of rows: 7
A B C D E F G
A B C D E F
A B C D E
A B C D
A B C
A B
A
```

Pattern-19:

10 10 10 10 10 10 10 10 10 10
9 9 9 9 9 9 9 9 9
8 8 8 8 8 8 8 8
7 7 7 7 7 7 7
6 6 6 6 6 6
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1

In [45]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+2-i):
        print(n+1-i,end=" ")
    print()
```

```
Enter the number of rows: 9
9 9 9 9 9 9 9 9 9
8 8 8 8 8 8 8 8
7 7 7 7 7 7 7
6 6 6 6 6 6
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1
```

Pattern-20:

10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2
10 9 8 7 6 5 4 3
10 9 8 7 6 5 4
10 9 8 7 6 5
10 9 8 7 6
10 9 8 7
10 9 8
10 9
10

In [46]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+2-i):
        print(n+1-j,end=" ")
    print()
```

```
Enter the number of rows: 10
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2
10 9 8 7 6 5 4 3
10 9 8 7 6 5 4
10 9 8 7 6 5
10 9 8 7 6
10 9 8 7
10 9 8
10 9
10
```

Pattern-21:

J J J J J J J J J J
I I I I I I I I I
H H H H H H H H
G G G G G G G
F F F F F F
E E E E E
D D D D
C C C
B B
A

In [1]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+2-i):
        print(chr(65+n-i),end=" ")
    print()
```

```
Enter the number of rows: 10
J J J J J J J J J J
I I I I I I I I I
H H H H H H H H
G G G G G G G
F F F F F F
E E E E E
D D D D
C C C
B B
A
```

Pattern-22:

J I H G F E D C B A
J I H G F E D C B
J I H G F E D C
J I H G F E D
J I H G F E
J I H G F
J I H G
J I H
J I
J

In [2]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    for j in range(1,n+2-i):
        print(chr(65+n-j),end=" ")
    print()
```

```
Enter the number of rows: 10
J I H G F E D C B A
J I H G F E D C B
J I H G F E D C
J I H G F E D
J I H G F E
J I H G F
J I H G
J I H
J I
J
```

Pattern-23:

```
*
**
***
****
*****
******
*******
```

In [5]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),"*"*i,end=" ")
    print()
```

```
Enter the number of rows: 5
     *
    **
   ***
  ****
 *****
```

In [6]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),"* "*i,end=" ")
    print()
```

```
Enter the number of rows: 5
     *
    * *
   * * *
  * * * *
 * * * * *
```

Alternative Way:

In [7]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(1,i+1):
        print("*",end=" ")
    print()
```

```
Enter the number of rows: 5
    *
   * *
  * * *
 * * * *
* * * * *
```

Pattern-25:

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

6 6 6 6 6 6

7 7 7 7 7 7 7

8 8 8 8 8 8 8 8

In [8]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),(str(i)+" ")*i)
    print()
```

```
Enter the number of rows: 8
        1 

       2 2 

      3 3 3 

     4 4 4 4 

    5 5 5 5 5 

   6 6 6 6 6 6 

  7 7 7 7 7 7 7 

 8 8 8 8 8 8 8 8 
```

Another Pattern:

In [9]:

```
n = int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(' '*(n-i),((str(i))*i))
    print()
```

```
Enter the number of rows: 5
     1

    22

   333

  4444

 55555
```

Pattern-26:

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10

In [10]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(1,i+1):
        print(j,end=" ")
    print()
```

```
Enter the number of rows: 10
         1
        1 2
       1 2 3
      1 2 3 4
     1 2 3 4 5
    1 2 3 4 5 6
   1 2 3 4 5 6 7
  1 2 3 4 5 6 7 8
 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
```

Pattern-27:

A

B B

C C C

D D D D

E E E E E

F F F F F F

G G G G G G G

H H H H H H H H

In [11]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),(chr(64+i)+" ")*i)
    print()
```

```
Enter the number of rows: 8
        A

       B B

      C C C

     D D D D

    E E E E E

   F F F F F F

  G G G G G G G

 H H H H H H H H
```

Pattern-30:

```
5 5 5 5 5
  4 4 4 4
    3 3 3
      2 2
        1
```

In [14]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(i-1),str(i)*(n+1-i))
```

```
Enter the number of rows: 5
 11111
  2222
   333
    44
     5
```

In [15]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(i-1),str(n+1-i)*(n+1-i))
```

```
Enter the number of rows: 5
 55555
  4444
   333
    22
     1
```

In [12]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(i-1),(str(n+1-i)+" ")*(n+1-i))
```

```
Enter the number of rows: 5
 5 5 5 5 5
  4 4 4 4
   3 3 3
    2 2
     1
```

Pattern-31:

```
1 2 3 4 5
  1 2 3 4
    1 2 3
      1 2
        1
```

In [16]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(i-1),end="")
    for j in range(1,n+2-i):
        print(j,end=" ")
    print()
```

```
Enter the number of rows: 5
1 2 3 4 5
 1 2 3 4
  1 2 3
   1 2
    1
```

In [17]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(i-1),end=' ')
    for j in range(1,n+2-i):
        print(j,end="")
    print()
```

```
Enter the number of rows: 5
 12345
  1234
   123
    12
     1
```

Pattern-32:

```
E E E E E
  D D D D
    C C C
      B B
        A
```

In [18]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(i-1),(str(chr(65+n-i))+" ")*(n+1-i))
```

```
Enter the number of rows: 5
 E E E E E
  D D D D
   C C C
    B B
     A
```

In [19]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(i-1),(str(chr(65+n-i)))*(n+1-i))
```

```
Enter the number of rows: 5
 EEEEE
  DDDD
   CCC
    BB
     A
```

Pattern-33:

```
A B C D E
 A B C D
  A B C
   A B
    A
```

In [20]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(i-1),end="")
    for j in range(65,66+n-i):
        print(chr(j),end=" ")
    print()
```

```
Enter the number of rows: 5
A B C D E
 A B C D
  A B C
   A B
    A
```

In [21]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(i-1),end="")
    for j in range(65,66+n-i):
        print(chr(j),end="")
    print()
```

```
Enter the number of rows: 5
ABCDE
 ABCD
  ABC
   AB
    A
```

Pattern-35:

```
    1
   2 2 2
  3 3 3 3 3
 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5
```

In [23]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),str(i)*(2*i-1))
```

```
Enter the number of rows: 5
     1
    222
   33333
  4444444
 555555555
```

Pattern-36:

```
    A
   B B B
  C C C C C
 D D D D D D D
E E E E E E E E E
```

In [24]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),(str(chr(64+i)))*(2*i-1))
```

```
Enter the number of rows: 5
     A
    BBB
   CCCCC
  DDDDDDD
 EEEEEEEEE
```

Pattern-37:

```
    A
  C C C
 E E E E E
G G G G G G G
I I I I I I I I I
```

In [25]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),(str(chr(64+2*i-1)))*(2*i-1))
```

```
Enter the number of rows: 5
     A
    CCC
   EEEEE
  GGGGGGG
 IIIIIIIII
```

In [26]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),(str(chr(64+3*i-1)))*(2*i-1))
```

```
Enter the number of rows: 5
     B
    EEE
   HHHHH
  KKKKKKK
 NNNNNNNNN
```

Pattern-38:

```
    1
   1 2 3
  1 2 3 4 5
 1 2 3 4 5 6 7
1 2 3 4 5 6 7 8 9
```

In [28]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(1,2*i):
        print(j,end="")
    print()
```

```
Enter the number of rows: 5
    1
   123
  12345
 1234567
123456789
```

Pattern-39:

```
    1
   3 2 1
  5 4 3 2 1
 7 6 5 4 3 2 1
9 8 7 6 5 4 3 2 1
```

In [29]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(2*i-1,0,-1):
        print(j,end="")
    print()
```

```
Enter the number of rows: 5
    1
   321
  54321
 7654321
987654321
```

Few more similar patterns:

In [31]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(2*i,1,-1):
        print(j,end="")
    print()
```

```
Enter the number of rows: 5
    2
   432
  65432
 8765432
1098765432
```

In [32]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(2*i,0,-1):
        print(j,end="")
    print()
```

```
Enter the number of rows: 5
    21
   4321
  654321
 87654321
10987654321
```

Pattern-40:

```
    A
   A B C
  A B C D E
 A B C D E F G
A B C D E F G H I
```

In [33]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(65,65+2*i-1):
        print(chr(j),end="")
    print()
```

```
Enter the number of rows: 5
    A
   ABC
  ABCDE
 ABCDEFG
ABCDEFGHI
```

Pattern-41:

```
    A
   C B A
  E D C B A
 G F E D C B A
I H G F E D C B A
```

In [34]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(65+2*i-2,64,-1):
        print(chr(j),end="")
    print()
```

```
Enter the number of rows: 5
    A
   CBA
  EDCBA
 GFEDCBA
IHGFEDCBA
```

Pattern-42:

```
    0
   1 0 1
  2 1 0 1 2
 3 2 1 0 1 2 3
4 3 2 1 0 1 2 3 4
```

In [37]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(1,i):
        print(i-j,end="")
    for k in range(0,i):
        print(k,end="")
    print()
```

```
Enter the number of rows: 5
    0
   101
  21012
 3210123
432101234
```

Pattern-43:

```
    A
   B A B
  C B A B C
 D C B A B C D
E D C B A B C D E
```

In [38]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(1,i):
        print(chr(i-j+65),end="")
    for k in range(0,i):
        print(chr(k+65),end="")
    print()
```

```
Enter the number of rows: 5
    A
   BAB
  CBABC
 DCBABCD
EDCBABCDE
```

Pattern-44:

```
    1
   1 2 1
  1 2 3 2 1
 1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
```

In [39]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(1,i+1):
        print(j,end="")
    for k in range(i-1,0,-1):
        print(k,end="")
    print()
```

```
Enter the number of rows: 5
    1
   121
  12321
 1234321
123454321
```

Pattern-45:

```
    A
   A B A
  A B C A B
 A B C D A B C
A B C D E A B C D
```

In [41]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(1,i+1):
        print(chr(64+j),end="")
    for k in range(1,i):
        print(chr(64+k),end="")
    print()
```

```
Enter the number of rows: 5
    A
   ABA
  ABCAB
 ABCDABC
ABCDEABCD
```

Pattern 46:

```
    5
   5 4
  5 4 3
 5 4 3 2
5 4 3 2 1
```

In [42]:

```
n=int(input("Enter a number:"))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(1,i+1):
        print(n+1-j,end="")
    print()
```

```
Enter a number:5
    5
   54
  543
 5432
54321
```

In [44]:

```
n=int(input("Enter a number:"))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(1,i+1):
        print(n-j+1,end=" ")
    print()
```

```
Enter a number:5
    5
   5 4
  5 4 3
 5 4 3 2
5 4 3 2 1
```

Pattern 47:

```
* * * * * * * * *
 * * * * * * *
  * * * * *
   * * *
    *
```

In [45]:

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(i-1),end="")
    for j in range(1,num+2-i):
        print("*",end=" ")
    for k in range(1,num+1-i):
        print("*",end=" ")
    print()
```

```
Enter a number:5
* * * * * * * * *
 * * * * * * *
  * * * * *
   * * *
    *
```

In [47]:

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(i-1),end="")
    for j in range(1,num+2-i):
        print("*",end="")
    for k in range(1,num+1-i):
        print("*",end="")
    print()
```

```
Enter a number:5
*********
 *******
  *****
   ***
    *
```

Alternative Way:

In [48]:

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(i-1),end="")
    for j in range(1,num+2-i):
        print("*",end=" ")
    print()
```

```
Enter a number:5
* * * * *
 * * * *
  * * *
   * *
    *
```

Pattern 48:

```
5 5 5 5 5 5 5 5 5
 4 4 4 4 4 4 4
  3 3 3 3 3
   2 2 2
    1
```

In [49]:

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(i-1),end="")
    for j in range(0,num+1-i):
        print(num+1-i,end="")
    for k in range(1,num+1-i):
        print(num+1-i,end="")
    print()
```

```
Enter a number:5
555555555
 4444444
  33333
   222
    1
```

Simi;ar type Patterns:

In [50]:

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(i-1),end="")
    for j in range(0,num+2-i):
        print(num+1-i,end="")
    for k in range(1,num-i+1):
        print(num+1-i,end="")
    print()
```

```
Enter a number:5
5555555555
 44444444
  333333
   2222
    11
```

Pattern 49:

```
9 9 9 9 9 9 9 9 9
 7 7 7 7 7 7 7
  5 5 5 5 5
   3 3 3
    1
```

In [51]:

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(i-1),end="")
    for j in range(0,num+1-i):
        print(2*num+1-2*i,end="")
    for k in range(1,num+1-i):
        print(2*num+1-2*i,end="")
    print()
```

```
Enter a number:5
999999999
 7777777
  55555
   333
    1
```

Pattern 50:

```
1 2 3 4 5 6 7
 1 2 3 4 5
  1 2 3
   1
```

In [52]:

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(i-1),end="")
    for j in range(1,num+2-i):
        print(j,end="")
    for k in range(2,num+2-i):
        print(num+k-i,end="")
    print()
```

```
Enter a number:5
123456789
 1234567
  12345
   123
    1
```

d) Demonstrate the following control transfer statements in Python with suitable examples.

```
i) break          ii) continue          iii) pass
```

i) break:

We can use break statement inside loops to break loop execution based on some condition.

In [3]:

```
for i in range(10):
    if i==7:
        print("processing is enough..plz break")
        break
    print(i)
```

```
0
1
2
3
4
5
6
processing is enough..plz break
```

In [4]:

```
cart=[10,20,600,60,70]
for item in cart:
    if item>500:
        print("To place this order insurence must be required")
        break
    print(item)
```

```
10
20
To place this order insurence must be required
```

ii) continue:

We can use continue statement to skip current iteration and continue next iteration.

Eg 1: Write a Python program to print odd numbers in the range 0 to 9.

In [5]:

```
for i in range(10):
    if i%2==0:
        continue
    print(i)
```

```
1
3
5
7
9
```

Eg 2:

In [6]:

```
cart=[10,20,500,700,50,60]
for item in cart:
    if item >= 500:
        print("We cannot process this item :",item)
        continue
    print(item)
```

```
10
20
We cannot process this item : 500
We cannot process this item : 700
50
60
```

Eg 3:

In [7]:

```
numbers=[10,20,0,5,0,30]
for n in numbers:
    if n==0:
        print("Hey how we can divide with zero..just skipping")
        continue
    print("100/{} = {}".format(n,100/n))
```

```
100/10 = 10.0
100/20 = 5.0
Hey how we can divide with zero..just skipping
100/5 = 20.0
Hey how we can divide with zero..just skipping
100/30 = 3.3333333333333335
```

Loops with else block:

Inside loop execution,if break statement not executed ,then only else part will be executed. else means loop without break.

In [9]:

```
cart=[10,20,30,40,50]
for item in cart:
    if item>=500:
        print("We cannot process this order")
        break
    print(item)
else:
    print("Congrats ...all items processed successfully")
```

```
10
20
30
40
50
Congrats ...all items processed successfully
```

In [11]:

```
cart=[10,20,600,30,40,50]
for item in cart:
    if item>=500:
        print("We cannot process this order")
        break
    print(item)
else:
    print("Congrats ...all items processed successfully")
```

```
10
20
We cannot process this order
```

iii) pass:

pass is a keyword in Python.
In our programming syntactically if block is required which won't do anything then we can define that empty block with pass keyword.
pass statement ==>

```
1. It is an empty statement

2. It is null statement

3. It won't do anything
```

In [12]:

```
if True: # It is invalid
```

```
  File "<ipython-input-12-2d7926e5e65a>", line 1
    if True: # It is invalid
                            ^
SyntaxError: unexpected EOF while parsing
```

In [13]:

```
if True:
    pass # It is valid
```

In [14]:

```
def m1(): # It is invalid
```

```
  File "<ipython-input-14-55805493e471>", line 1
    def m1(): # It is invalid
                             ^
SyntaxError: unexpected EOF while parsing
```

In [15]:

```
def m1():
    pass # It is valid
```

Use of pass:

Sometimes in the parent class we have to declare a function with empty body and child class responsible to provide proper implementation. Such type of empty body we can define by using pass keyword. (It is something like abstract method in java).

Example:

In [16]:

```
for i in range(100):
    if i%9==0:
        print(i)
else:
    pass
```

```
0
9
18
27
36
45
54
63
72
81
90
99
```

More example programs on all the above concepts:

Q1. Write a program to find biggest of given 2 numbers.

In [33]:

```
n1=int(input("Enter First Number:"))
n2=int(input("Enter Second Number:"))
if n1>n2:
    print("Biggest Number is:",n1)
else :
    print("Biggest Number is:",n2)
```

```
Enter First Number:10
Enter Second Number:20
Biggest Number is: 20
```

Q2. Write a program to find biggest of given 3 numbers.

In [34]:

```
n1=int(input("Enter First Number:"))
n2=int(input("Enter Second Number:"))
n3=int(input("Enter Third Number:"))
if n1>n2 and n1>n3:
    print("Biggest Number is:",n1)
elif n2>n3:
    print("Biggest Number is:",n2)
else :
    print("Biggest Number is:",n3)
```

```
Enter First Number:10
Enter Second Number:20
Enter Third Number:35
Biggest Number is: 35
```

Q3. Write a program to find smallest of given 2 numbers?

In [35]:

```
n1=int(input("Enter First Number:"))
n2=int(input("Enter Second Number:"))
if n1>n2:
    print("Smallest Number is:",n2)
else :
    print("Smallest Number is:",n1)
```

```
Enter First Number:35
Enter Second Number:44
Smallest Number is: 35
```

Q4. Write a program to find smallest of given 3 numbers?

In [36]:

```
n1=int(input("Enter First Number:"))
n2=int(input("Enter Second Number:"))
n3=int(input("Enter Third Number:"))
if n1<n2 and n1<n3:
    print("Smallest Number is:",n1)
elif n2<n3:
    print("Smallest Number is:",n2)
else :
    print("Smallest Number is:",n3)
```

```
Enter First Number:100
Enter Second Number:350
Enter Third Number:125
Smallest Number is: 100
```

Q5. Write a program to check whether the given number is even or odd?

In [37]:

```
n1=int(input("Enter First Number:"))
rem = n1 % 2
if rem == 0:
    print('Entered Number is an Even Number')
else:
    print('Entered Number is an Odd Number')
```

```
Enter First Number:34
Entered Number is an Even Number
```

In [39]:

```
n1=int(input("Enter First Number:"))
rem = n1 % 2
if rem == 0:
    print('Entered Number is an Even Number')
else:
    print('Entered Number is an Odd Number')
```

```
Enter First Number:33
Entered Number is an Odd Number
```

Q6. Write a program to check whether the given number is in between 1 and 100?

In [40]:

```
n=int(input("Enter Number:"))
if n>=1 and n<=100 :
    print("The number",n,"is in between 1 to 100")
else:
    print("The number",n,"is not in between 1 to 100")
```

```
Enter Number:45
The number 45 is in between 1 to 100
```

In [41]:

```
n=int(input("Enter Number:"))
if n>=1 and n<=100 :
    print("The number",n,"is in between 1 to 100")
else:
    print("The number",n,"is not in between 1 to 100")
```

```
Enter Number:123
The number 123 is not in between 1 to 100
```

Q7. Write a program to take a single digit number from the key board and print it's value in English word?

In [42]:

```
n=int(input("Enter a digit from o to 9:"))
if n==0 :
    print("ZERO")
elif n==1:
    print("ONE")
elif n==2:
    print("TWO")
elif n==3:
    print("THREE")
elif n==4:
    print("FOUR")
elif n==5:
    print("FIVE")
elif n==6:
    print("SIX")
elif n==7:
    print("SEVEN")
elif n==8:
    print("EIGHT")
elif n==9:
    print("NINE")
else:
    print("PLEASE ENTER A DIGIT FROM 0 TO 9")
```

```
Enter a digit from o to 9:8
EIGHT
```

In [43]:

```
n=int(input("Enter a digit from o to 9:"))
if n==0 :
    print("ZERO")
elif n==1:
    print("ONE")
elif n==2:
    print("TWO")
elif n==3:
    print("THREE")
elif n==4:
    print("FOUR")
elif n==5:
    print("FIVE")
elif n==6:
    print("SIX")
elif n==7:
    print("SEVEN")
elif n==8:
    print("EIGHT")
elif n==9:
    print("NINE")
else:
    print("PLEASE ENTER A DIGIT FROM 0 TO 9")
```

```
Enter a digit from o to 9:10
PLEASE ENTER A DIGIT FROM 0 TO 9
```

Another Way of writing program for the same requirement:

In [44]:

```
list1 = ['ZERO','ONE','TWO','THREE','FOUR','FIVE','SIX','SEVEN','EIGHT','NINE']
n =int(input('Enter a digit from 0 to 9 :'))
print(list1[n])
```

```
Enter a digit from 0 to 9 :7
SEVEN
```

In [45]:

```
list1 = ['ZERO','ONE','TWO','THREE','FOUR','FIVE','SIX','SEVEN','EIGHT','NINE']
n =int(input('Enter a digit from 0 to 9 :'))
print(list1[n])
```

```
Enter a digit from 0 to 9 :15

---------------------------------------------------------------------------
-
IndexError                                Traceback (most recent call las
t)
<ipython-input-45-bbd0655ae62d> in <module>
      1 list1 = ['ZERO','ONE','TWO','THREE','FOUR','FIVE','SIX','SEVEN','E
IGHT','NINE']
      2 n =int(input('Enter a digit from 0 to 9 :'))
----> 3 print(list1[n])

IndexError: list index out of range
```

How can you extend the above program from 0 to 99?

In [47]:

```
words_upto_19 = ['','ONE','TWO','THREE','FOUR','FIVE','SIX','SEVEN','EIGHT',
                 'NINE','TEN','ELEVEN','TWELVE','THIRTEEN','FOURTEEN','FIFTEEN',
                 'SIXTEEN','SEVENTEEN','EIGHTEEN','NINETEEN']
words_for_tens = ['','','TWENTY','THIRTY','FORTY','FIFTY','SIXTY','SEVENTY',
                  'EIGHTY','NINETY']
n = int(input('Enter a number from 0 to 99 : '))
output = ''
if n == 0:
    output = 'ZERO'
elif n <= 19:
    output = words_upto_19[n]
elif n<=99:
    output = words_for_tens[n//10]+' '+words_upto_19[n%10]
else:
    output = 'Pleae Enter a value fron 0 to 99 only'
print(output)
```

```
Enter a number from 0 to 99 : 0
ZERO
```

In [48]:

```
words_upto_19 = ['','ONE','TWO','THREE','FOUR','FIVE','SIX','SEVEN','EIGHT',
                 'NINE','TEN','ELEVEN','TWELVE','THIRTEEN','FOURTEEN','FIFTEEN',
                 'SIXTEEN','SEVENTEEN','EIGHTEEN','NINETEEN']
words_for_tens = ['','','TWENTY','THIRTY','FORTY','FIFTY','SIXTY','SEVENTY',
                  'EIGHTY','NINETY']
n = int(input('Enter a number from 0 to 99 : '))
output = ''
if n == 0:
    output = 'ZERO'
elif n <= 19:
    output = words_upto_19[n]
elif n<=99:
    output = words_for_tens[n//10]+' '+words_upto_19[n%10]
else:
    output = 'Pleae Enter a value fron 0 to 99 only'
print(output)
```

```
Enter a number from 0 to 99 : 9
NINE
```

In [49]:

```
words_upto_19 = ['','ONE','TWO','THREE','FOUR','FIVE','SIX','SEVEN','EIGHT',
                 'NINE','TEN','ELEVEN','TWELVE','THIRTEEN','FOURTEEN','FIFTEEN',
                 'SIXTEEN','SEVENTEEN','EIGHTEEN','NINETEEN']
words_for_tens = ['','','TWENTY','THIRTY','FORTY','FIFTY','SIXTY','SEVENTY',
                  'EIGHTY','NINETY']
n = int(input('Enter a number from 0 to 99 : '))
output = ''
if n == 0:
    output = 'ZERO'
elif n <= 19:
    output = words_upto_19[n]
elif n<=99:
    output = words_for_tens[n//10]+' '+words_upto_19[n%10]
else:
    output = 'Pleae Enter a value fron 0 to 99 only'
print(output)
```

```
Enter a number from 0 to 99 : 19
NINETEEN
```

In [50]:

```
words_upto_19 = ['','ONE','TWO','THREE','FOUR','FIVE','SIX','SEVEN','EIGHT',
                 'NINE','TEN','ELEVEN','TWELVE','THIRTEEN','FOURTEEN','FIFTEEN',
                 'SIXTEEN','SEVENTEEN','EIGHTEEN','NINETEEN']
words_for_tens = ['','','TWENTY','THIRTY','FORTY','FIFTY','SIXTY','SEVENTY',
                  'EIGHTY','NINETY']
n = int(input('Enter a number from 0 to 99 : '))
output = ''
if n == 0:
    output = 'ZERO'
elif n <= 19:
    output = words_upto_19[n]
elif n<=99:
    output = words_for_tens[n//10]+' '+words_upto_19[n%10]
else:
    output = 'Pleae Enter a value fron 0 to 99 only'
print(output)
```

```
Enter a number from 0 to 99 : 56
FIFTY SIX
```

Q8. Python program to find all prime numbers within a given range.

Theory:

Prime numbers:

A prime number is a natural number greater than 1 and having no positive divisor other than 1 and itself.

For example: 3, 7, 11 etc are prime numbers.

Composite number: Other natural numbers that are not prime numbers are called composite numbers.

For example: 4, 6, 9 etc. are composite numbers.

Here is source code of the Python Program to check if a number is a prime number.

In [24]:

```
r=int(input("Enter Range: "))
for a in range(2,r+1):
    k=0
    for i in range(2,a//2+1):
        if(a%i==0):
            k=k+1
    if(k<=0):
        print(a)
```

```
Enter Range: 20
2
3
5
7
11
13
17
19
```

In [25]:

```
r=int(input("Enter Range: "))
for a in range(2,r+1):
    k=0
    for i in range(2,a//2+1):
        if(a%i==0):
            k=k+1
    if(k<=0):
        print(a,end=' ')
```

```
Enter Range: 20
2 3 5 7 11 13 17 19
```

Q9. Python program to print ‘n' terms of Fibonacci series of numbers.

Theory:

A Fibonacci sequence is the integer sequence of 0, 1, 1, 2, 3, 5, 8....
The first two terms are 0 and 1. All other terms are obtained by adding the preceding two terms. This means to say the nth term is the sum of (n-1)th and (n-2)th term.

In [27]:

```
n = int(input("How many terms? "))
# first two terms
n1 = 0
n2 = 1
count = 0
# check if the number of terms is valid
if n <= 0:
    print("Please enter a positive integer")
elif n == 1:
    print("Fibonacci sequence upto",n,":")
    print(n1)
else:
    print("Fibonacci sequence upto",n,":")
    while count < n:
        print(n1,end=' ')
        next = n1 + n2
    # update values
        n1 = n2
        n2 = next
        count += 1
```

```
How many terms? 20
Fibonacci sequence upto 20 :
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
```

In [28]:

```
n = int(input("How many terms? "))
# first two terms
n1 = 0
n2 = 1
count = 0
# check if the number of terms is valid
if n <= 0:
    print("Please enter a positive integer")
elif n == 1:
    print("Fibonacci sequence upto",n,":")
    print(n1)
else:
    print("Fibonacci sequence upto",n,":")
    while count < n:
        print(n1,end=' ')
        next = n1 + n2
    # update values
        n1 = n2
        n2 = next
        count += 1
```

```
How many terms? 10
Fibonacci sequence upto 10 :
0 1 1 2 3 5 8 13 21 34
```

Q10. Write a Python program to compute distance between two points taking input from the user (Pythagorean Theorem).

In [30]:

```
import math;
x1=int(input("Enter x1--->"))
y1=int(input("Enter y1--->"))
x2=int(input("Enter x2--->"))
y2=int(input("Enter y2--->"))
d1 = (x2 - x1) * (x2 - x1);
d2 = (y2 - y1) * (y2 - y1);
res = math.sqrt(d1+d2)
print ("Distance between two points:",res);
```

```
Enter x1--->5
Enter y1--->10
Enter x2--->15
Enter y2--->20
Distance between two points: 14.142135623730951
```

Q11. Write a Python program to compute the GCD of two numbers.

In [32]:

```
def gcd(a,b):
    if(b==0):
        return a
    else:
        return gcd(b,a%b)
a=int(input("Enter first number:"))
b=int(input("Enter second number:"))
print (gcd(a,b))
```

```
Enter first number:5
Enter second number:15
5
```

Q12. Write a Python program to find the exponentiation of a number.

In [34]:

```
num =int(input("Enter a number:"))
exp =int(input("Enter a number:"))
res=num
for i in range(1,exp):
    res=num*res
print ("Exponent",res)
```

```
Enter a number:3
Enter a number:4
Exponent 81
```

Q13. A cashier has currency notes of denominations 10, 50 and 100. If the amount to be withdrawn is input through the keyboard in hundreds, write a Python program find the total number of currency notes of each denomination the cashier will have to give to the withdrawer.

In [4]:

```
Amount = int(input("Please Enter Amount for Withdraw :"))

print("\n\nRequired notes of 100 is : " , Amount // 100)    # Floor Division
print ("Required notes of 50 is : " , (Amount % 100) // 50)
print ("Required notes of 10 is : " , (((Amount % 100) % 50) // 10))
print ("Amount still remaining is : " , (((Amount % 100) % 50) % 10))
```

```
Please Enter Amount for Withdraw :1575


Required notes of 100 is :  15
Required notes of 50 is :  1
Required notes of 10 is :  2
Amount still remaining is :  5
```

In [5]:

```
Amount = int(input("Please Enter Amount for Withdraw :"))

print("\n\nRequired notes of 100 is : " , Amount // 100)    # Floor Division
print ("Required notes of 50 is : " , (Amount % 100) // 50)
print ("Required notes of 10 is : " , (((Amount % 100) % 50) // 10))
print ("Amount still remaining is : " , (((Amount % 100) % 50) % 10))
```

```
Please Enter Amount for Withdraw :0


Required notes of 100 is :  0
Required notes of 50 is :  0
Required notes of 10 is :  0
Amount still remaining is :  0
```

In [6]:

```
Amount = int(input("Please Enter Amount for Withdraw :"))

print("\n\nRequired notes of 100 is : " , Amount // 100)    # Floor Division
print ("Required notes of 50 is : " , (Amount % 100) // 50)
print ("Required notes of 10 is : " , (((Amount % 100) % 50) // 10))
print ("Amount still remaining is : " , (((Amount % 100) % 50) % 10))
```

```
Please Enter Amount for Withdraw :100456


Required notes of 100 is :  1004
Required notes of 50 is :  1
Required notes of 10 is :  0
Amount still remaining is :  6
```

Q14. Python Program to calculate overtime pay of 10 employees. Overtime is paid at the rate of Rs. 12.00 per hour for every hour worked above 40 hours. Assume that employees do not work for fractional part of an hour.

In [7]:

```
overtime_pay = 0
for i in range(10) :
    print("\nEnter the time employee worked in hr ")
    time_worked = int(input())
    if (time_worked>40):
        over_time = time_worked - 40
        overtime_pay = overtime_pay + (12 * over_time)
print("\nTotal Overtime Pay Of 10 Employees Is ", overtime_pay)
```

```
Enter the time employee worked in hr
45

Enter the time employee worked in hr
42

Enter the time employee worked in hr
43

Enter the time employee worked in hr
44

Enter the time employee worked in hr
50

Enter the time employee worked in hr
53

Enter the time employee worked in hr
42

Enter the time employee worked in hr
66

Enter the time employee worked in hr
44

Enter the time employee worked in hr
33

Total Overtime Pay Of 10 Employees Is  828
```

Q15. A library charges a fine for every book returned late. For first five days the fine is 50 paise, for 6 to 10 days fine is one rupee, and above 10 days fine is five rupees. If you return the book after 30 days your membership will be cancelled. Write a Python program to accept the number of days the member is late to return the book and display the fine or the appropriate message.

In [16]:

```
days = int(input("Enter Number of Days : "))

if((days>0) and (days<=5)):
    fine = 0.5 * days
elif((days>5) and (days<=10)):
    fine = 1 * days
elif((days>10)):
    fine = 5 * days
if(days > 30):
    print('Your Membership Cancelled !!!')
print('You have to pay Rs.',fine)
```

```
Enter Number of Days : 45
Your Membership Cancelled !!!
You have to pay Rs. 225
```

In [17]:

```
days = int(input("Enter Number of Days : "))

if((days>0) and (days<=5)):
    fine = 0.5 * days
elif((days>5) and (days<=10)):
    fine = 1 * days
elif((days>10)):
    fine = 5 * days
if(days > 30):
    print('Your Membership Cancelled !!!')
print('You have to pay Rs.',fine)
```

```
Enter Number of Days : 6
You have to pay Rs. 6
```

In [18]:

```
days = int(input("Enter Number of Days : "))

if((days>0) and (days<=5)):
    fine = 0.5 * days
elif((days>5) and (days<=10)):
    fine = 1 * days
elif((days>10)):
    fine = 5 * days
if(days > 30):
    print('Your Membership Cancelled !!!')
print('You have to pay Rs.',fine)
```

```
Enter Number of Days : 1
You have to pay Rs. 0.5
```

In [19]:

```
days = int(input("Enter Number of Days : "))

if((days>0) and (days<=5)):
    fine = 0.5 * days
elif((days>5) and (days<=10)):
    fine = 1 * days
elif((days>10)):
    fine = 5 * days
if(days > 30):
    print('Your Membership Cancelled !!!')
print('You have to pay Rs.',fine)
```

```
Enter Number of Days : 12
You have to pay Rs. 60
```

Q16. Two numbers are entered through keyboard, Write a Python program to find the value of one number raised to the power another.

In [20]:

```
import math
n1 = int(input("Please enter a number : "))
exp = int(input("Please enter exponent value : "))

power = math.pow(n1,exp)

print('The result of {} power {} = {}'.format(n1,exp,power))
```

```
Please enter a number : 3
Please enter exponent value : 4
The result of 3 power 4 = 81.0
```

Viva Questions:

Q 1. What is the difference between for loop and while loop in Python?

We can use loops to repeat code execution.
Repeat code for every item in sequence ==>for loop
Repeat code as long as condition is true ==>while loop

Q 2. How to exit from the loop?

by using break statement.

Q 3. How to skip some iterations inside loop?

by using continue statement.

Q4. When else part will be executed with respect to loops?

If loop executed without break.

GOOD LUCK

Experiment 4:

Write Python programs to print the following Patterns.

i)

```
A
A B
A B C
A B C D
A B C D E
```

In [1]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(n-i),end="")
    for j in range(1,i+1):
        print(chr(64+j),end=" ")
    print()
```

```
Enter the number of rows: 5
    A
   A B
  A B C
 A B C D
A B C D E
```

ii)

```
* * * * *
 * * * *
  * * *
   * *
    *
```

In [3]:

```
n=int(input("Enter the number of rows: "))
for i in range(1,n+1):
    print(" "*(i-1),"*"*(n+1-i))
```

```
Enter the number of rows: 5
 *****
  ****
   ***
    **
     *
```

iii)

```
E E E E E E E E E
 D D D D D D D
  C C C C C
   B B B
    A
```

In [6]:

```
num=int(input("Enter the number of rows:"))
for i in range(1,num+1):
    print(" "*(i-1),end="")
    for j in range(1,num+2-i):
        print(chr(65+num-i),end=" ")
    for k in range(2,num+2-i):
        print(chr(65+num-i),end=" ")
    print()
```

```
Enter the number of rows:5
E E E E E E E E E
 D D D D D D D
  C C C C C
   B B B
    A
```

iv)

```
4
4 3
4 3 2
4 3 2 1
4 3 2 1 0
4 3 2 1
4 3 2
4 3
4
```

In [7]:

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(num-i),end="")
    for j in range(1,i+1):
        print(num-j,end=" ")
    print()
for k in range(1,num):
    print(" "*k,end="")
    for l in range(1,num+1-k):
        print(num-l,end=" ")
    print()
```

```
Enter a number:5
    4
   4 3
  4 3 2
 4 3 2 1
4 3 2 1 0
 4 3 2 1
  4 3 2
   4 3
    4
```

v)

```
4
3 4
2 3 4
1 2 3 4
0 1 2 3 4
1 2 3 4
2 3 4
3 4
4
```

In [8]:

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    for j in range(1,i+1):
        print(num-i+j-1,end=" ")
    print()
for a in range(1,num+1):
    for k in range(0,num-a):
        print(k+a,end=" ")
    print()
```

```
Enter a number:5
4
3 4
2 3 4
1 2 3 4
0 1 2 3 4
1 2 3 4
2 3 4
3 4
4
```

vi)

```
*       *
* *     * *
* * *   * * *
* * * *  * * * *
* * * * * * * * * *
```

In [16]:

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(num-i),end="")
    for j in range(1,i+1):
        print("*",end=" ")
    print("  "*(num-i),end="")
    for k in range(1,i+1):
        print("*",end=" ")
    print()
```

```
Enter a number:5
    *         *
   * *       * *
  * * *     * * *
 * * * *   * * * *
* * * * * * * * * *
```

vii)

```
**
**
****
****
******
******
********
********
**********
**********
```

In [17]:

```
n=int(input("Enter a number"))
for i in range(1,2*n+1):
    if i%2==0:
        print("*"*i,end=" ")
    else:
        print("*"*(i+1),end=" ")
    print()
```

```
Enter a number5
**
**
****
****
******
******
********
********
**********
**********
```

viii)

```
E
D E
C D E
B C D E
A B C D E
B C D E
C D E
D E
E
```

In [18]:

```
num=int(input("Enter a number:"))
for i in range(1,num+1):
    print(" "*(num-i),end="")
    for j in range(0,i):
        print(chr(65+num+j-i),end=" ")
    print()
for k in range(1,num):
    print(" "*k,end="")
    for l in range(0,num-k):
        print(chr(65+k+l),end=" ")
    print()
```

```
Enter a number:5
    E
   D E
  C D E
 B C D E
A B C D E
 B C D E
  C D E
   D E
    E
```

Experiment 5:

a) Write a Python program to demonstrate various ways of accessing the string.

```
 i) By using Indexing (Both Positive and Negative)

ii) By using Slice Operator
```

What is String?

- Any sequence of characters within either single quotes or double quotes is considered as a String.

Syntax:

s='karthi'

s="karthi"

Note:

- In most of other languges like C, C++,Java, a single character with in single quotes is treated as char data type value. But in Python we are not having char data type.Hence it is treated as String only.

In [1]:

```
ch = 'a'
print(type(ch))
```

```
<class 'str'>
```

How to access characters of a String?

We can access characters of a string by using the following ways.

```
1. By using index

2. By using slice operator
```

1. By using index:

- Python supports both +ve and -ve index.
- +ve index means left to right(Forward direction).
- -ve index means right to left(Backward direction).

In [2]:

```
s = 'Karthi'
print(s[0])
print(s[5])
print(s[-1])
print(s[19])
```

```
K
i
i

---------------------------------------------------------------------------
-
IndexError                                Traceback (most recent call las
t)
<ipython-input-2-d39a6b459de8> in <module>
      3 print(s[5])
      4 print(s[-1])
----> 5 print(s[19])

IndexError: string index out of range
```

Eg: Q 1. Write a program to accept some string from the keyboard and display its characters by index wise(both positive and negative index).

In [6]:

```
s=input("Enter Some String:")
i=0
for x in s:
    print("The character present at positive index {} and at negative index {} is {}".f
ormat(i,len(s)-i,s[i]))
    i=i+1
```

```
Enter Some String:RGMxyz
The character present at positive index 0 and at negative index 6 is R
The character present at positive index 1 and at negative index 5 is G
The character present at positive index 2 and at negative index 4 is M
The character present at positive index 3 and at negative index 3 is x
The character present at positive index 4 and at negative index 2 is y
The character present at positive index 5 and at negative index 1 is z
```

2. Accessing characters by using slice operator:

- string slice means a part of the string (i.e, Sub string).

Syntax:

```
string_Name [beginindex:endindex:step]
```

Here,

i. beginindex: From where we have to consider slice(substring)

ii. endindex: We have to terminate the slice(substring) at endindex-1

iii. step: incremented / decremented value

Note:

- Slicing operator returns the sub string form beginindex to endindex - 1
- If we are not specifying begin index then it will consider from beginning of the string.
- If we are not specifying end index then it will consider up to end of the string.
- The default value for step is 1

In [7]:

```
s = 'abcdefghijk'
print(s[2:7])
```

```
cdefg
```

In [8]:

```
s = 'abcdefghijk'
print(s[:7])
```

```
abcdefg
```

In [9]:

```
s = 'abcdefghijk'
print(s[2:])
```

```
cdefghijk
```

In [10]:

```
s = 'abcdefghijk'
print(s[:])
```

```
abcdefghijk
```

In [11]:

```
s = 'abcdefghijk'
print(s[2:7:1])
```

```
cdefg
```

In [12]:

```
s = 'abcdefghijk'
print(s[2:7:2])
```

```
ceg
```

In [13]:

```
s = 'abcdefghijk'
print(s[2:7:3])
```

```
cf
```

In [14]:

```
s = 'abcdefghijk'
print(s[::1])
```

```
abcdefghijk
```

In [15]:

```
s = 'abcdefghijk'
print(s[::2])
```

```
acegik
```

In [16]:

```
s = 'abcdefghijk'
print(s[::3])
```

```
adgj
```

In [17]:

```
s="Learning Python is very very easy!!!"
s[1:7:1]
```

Out[17]:

```
'earnin'
```

In [18]:

```
s="Learning Python is very very easy!!!"
s[1:7]
```

Out[18]:

```
'earnin'
```

In [19]:

```
s="Learning Python is very very easy!!!"
s[1:7:2]
```

Out[19]:

```
'eri'
```

In [20]:

```
s="Learning Python is very very easy!!!"
s[:7]
```

Out[20]:

```
'Learnin'
```

In [21]:

```
s="Learning Python is very very easy!!!"
s[7:]
```

Out[21]:

```
'g Python is very very easy!!!'
```

In [22]:

```
s="Learning Python is very very easy!!!"
s[::]
```

Out[22]:

```
'Learning Python is very very easy!!!'
```

In [23]:

```
s="Learning Python is very very easy!!!"
s[:]
```

Out[23]:

```
'Learning Python is very very easy!!!'
```

In [24]:

```
s="Learning Python is very very easy!!!"
s[::-1]
```

Out[24]:

```
'!!!ysae yrev yrev si nohtyP gninraeL'
```

In [27]:

```
s = 'Learning Python'
print(s[::-1])
print(s[::-2])
print(s[::-3])
print(s[::-5])
print(s[::-10])
print(s[::-100])
print(s[3:5:-1])
print(s[3:5:1])
print(s[5:3:-1])
print(s[5:0:-1])
print(s[-2:-1:-1])
print(s[2:-1:-1])
print(s[2:0:1])
print(s[0:0:1])
```

```
nohtyP gninraeL
nhy nnaL
nt ia
nPn
nn
n

rn
in
inrae
```

Important Conclusions:

1. In the backward direction if the end value is -1, then the result is always empty string.
2. In the farward diretions if the end value is 0, then the result is always empty string.

In forward direction:

- default value for begin: 0
- default value for end: length of string
- default value for step: +1

In backward direction:

- default value for begin: -1
- default value for end: -(length of string + 1)

Note:

- Either forward or backward direction, we can take both +ve and -ve values for begin and end index.

b) Demonstrate the following functions/methods which operates on strings in Python with suitable examples:

```
i) len( )     ii) strip( )      iii) rstrip( )      iv) lstrip( )

v) find( )    vi) rfind( )       vii) index( )     viii) rindex()

ix) count( )    x) replace( )      xi) split( )     xii) join( )

xiii) upper( )  xiv) lower( )     xv) swapcase( )   xvi) title( )

xvii) capitalize( )    xviii) startswith()    xix) endswith()
```

i.len():

- We can use **len()** function to find the number of characters present in the string.

In [28]:

```
s='karthi'
print(len(s)) #6
```

6

Q1. Write a Python program to access each character of string in forward and backward direction by using while loop.

In [29]:

```
s="Learning Python is very easy !!!"
n=len(s)
i=0
print("Forward direction")
print()
while i<n:
    print(s[i],end=' ')
    i +=1
print('')
print('')
print("Backward direction")
print()
i=-1
while i>=-n:
    print(s[i],end=' ')
    i=i-1
```

```
Forward direction

L e a r n i n g   P y t h o n   i s   v e r y   e a s y   ! ! !

Backward direction

! ! !   y s a e   y r e v   s i   n o h t y P   g n i n r a e L
```

Alternative way [Using slice operator]:

In [30]:

```
s="Learning Python is very easy !!!"
print("Forward direction")
print('')
for i in s:
    print(i,end=' ')
print('')
print('')
print("Forward direction")
print('')
for i in s[::]:
    print(i,end=' ')
print('')
print('')
print('Backward Direction')
print('')
for i in s[::-1]:
    print(i,end=' ')
```

```
Forward direction

L e a r n i n g   P y t h o n   i s   v e r y   e a s y   ! ! !

Forward direction

L e a r n i n g   P y t h o n   i s   v e r y   e a s y   ! ! !

Backward Direction

! ! !   y s a e   y r e v   s i   n o h t y P   g n i n r a e L
```

Another Alternative:

In [31]:

```
s = input('Enter the string : ')
print('Data in Farward Direction')
print(s[::1])
print()
print('Data in Backward Direction')
print(s[::-1])
```

```
Enter the string : Python Learning is easy
Data in Farward Direction
Python Learning is easy

Data in Backward Direction
ysae si gninraeL nohtyP
```

Removing spaces from the string:

To remove the blank spaces present at either beginning and end of the string, we can use the following 3 methods:

1. rstrip() ===>To remove blank spaces present at end of the string (i.e.,right hand side)
2. lstrip()===>To remove blank spaces present at the beginning of the string (i.e.,left hand side)
3. strip() ==>To remove spaces both sides

ii.strip():

- Used to remove spaces both sides of the string.

In [1]:

```
city=input("Enter your city Name:")
scity=city.strip()
if scity=='Hyderabad':
    print("Hello Hyderbadi..Adab")
elif scity=='Chennai':
    print("Hello Madrasi...Vanakkam")
elif scity=="Bangalore":
    print("Hello Kannadiga...Shubhodaya")
else:
    print("your entered city is invalid")
```

```
Enter your city Name:Hyderabad
Hello Hyderbadi..Adab
```

In [3]:

```
scity=input("Enter your city Name:")
#scity=city.strip()
if scity=='Hyderabad':
    print("Hello Hyderbadi..Adab")
elif scity=='Chennai':
    print("Hello Madrasi...Vanakkam")
elif scity=="Bangalore":
    print("Hello Kannadiga...Shubhodaya")
else:
    print("your entered city is invalid")
```

```
Enter your city Name:    Hyderabad
your entered city is invalid
```

In [2]:

```
city=input("Enter your city Name:")
scity=city.strip()
if scity=='Hyderabad':
    print("Hello Hyderbadi..Adab")
elif scity=='Chennai':
    print("Hello Madrasi...Vanakkam")
elif scity=="Bangalore":
    print("Hello Kannadiga...Shubhodaya")
else:
    print("your entered city is invalid")
```

```
Enter your city Name:    Hyderabad
Hello Hyderbadi..Adab
```

iii.rstrip():

- Used to remove blank spaces present at end of the string (i.e.,right hand side)

In [4]:

```
scity=input("Enter your city Name:")
#scity=city.strip()
if scity=='Hyderabad':
    print("Hello Hyderbadi..Adab")
elif scity=='Chennai':
    print("Hello Madrasi...Vanakkam")
elif scity=="Bangalore":
    print("Hello Kannadiga...Shubhodaya")
else:
    print("your entered city is invalid")
```

```
Enter your city Name:Hyderabad
your entered city is invalid
```

In [5]:

```
city=input("Enter your city Name:")
scity=city.strip()
if scity=='Hyderabad':
    print("Hello Hyderbadi..Adab")
elif scity=='Chennai':
    print("Hello Madrasi...Vanakkam")
elif scity=="Bangalore":
    print("Hello Kannadiga...Shubhodaya")
else:
    print("your entered city is invalid")
```

```
Enter your city Name:Hyderabad
Hello Hyderbadi..Adab
```

iv.lstrip():

- Used to remove blank spaces present at the beginning of the string (i.e.,left hand side)

In [6]:

```
scity=input("Enter your city Name:")
#scity=city.strip()
if scity=='Hyderabad':
    print("Hello Hyderbadi..Adab")
elif scity=='Chennai':
    print("Hello Madrasi...Vanakkam")
elif scity=="Bangalore":
    print("Hello Kannadiga...Shubhodaya")
else:
    print("your entered city is invalid")
```

```
Enter your city Name:     Hyderabad
your entered city is invalid
```

In [7]:

```
city=input("Enter your city Name:")
scity=city.lstrip()
if scity=='Hyderabad':
    print("Hello Hyderbadi..Adab")
elif scity=='Chennai':
    print("Hello Madrasi...Vanakkam")
elif scity=="Bangalore":
    print("Hello Kannadiga...Shubhodaya")
else:
    print("your entered city is invalid")
```

```
Enter your city Name:     Hyderabad
Hello Hyderbadi..Adab
```

More Test cases:

In [9]:

```
city=input("Enter your city Name:")
scity=city.strip()
if scity=='Hyderabad':
    print("Hello Hyderbadi..Adab")
elif scity=='Chennai':
    print("Hello Madrasi...Vanakkam")
elif scity=="Bangalore":
    print("Hello Kannadiga...Shubhodaya")
else:
    print("your entered city is invalid")
```

```
Enter your city Name:   Bangalore
Hello Kannadiga...Shubhodaya
```

In [10]:

```
city=input("Enter your city Name:")
scity=city.lstrip()
if scity=='Hyderabad':
    print("Hello Hyderbadi..Adab")
elif scity=='Chennai':
    print("Hello Madrasi...Vanakkam")
elif scity=="Bangalore":
    print("Hello Kannadiga...Shubhodaya")
else:
    print("your entered city is invalid")
```

```
Enter your city Name:    Chennai
Hello Madrasi...Vanakkam
```

Finding Substrings:

- If you want to find whether the substring is available in the given string or not in Python, we have 4 methods.

For forward direction:

1. find()
2. index()

For backward direction:

1. rfind()
2. rindex()

v.find():

Syntax:

```
s.find(substring) (Without Boundary)
```

- Returns index of first occurrence of the given substring. If it is not available then we will get -1

In [11]:

```
s="Learning Python is very easy"
print(s.find("Python")) # 9
print(s.find("Java")) # -1
print(s.find("r")) # 3
print(s.rfind("r")) # 21
```

```
9
-1
3
21
```

- By default find() method can search total string. We can also specify the boundaries to search.

Syntax:

```
s.find(substring,begin,end) (With Boundary)
```

- It will always search from begin index to end-1 index.

In [12]:

```
s="karthikeyasahasra"
print(s.find('a')) #1
print(s.find('a',7,15)) #9
print(s.find('z',7,15)) #-1
```

```
1
9
-1
```

vi. rfind():

In [13]:

```
s="Learning Python is very easy"
print(s.rfind("Python")) # 9
print(s.rfind("Java")) # -1
print(s.find("r")) # 3
print(s.rfind("r")) # 21
```

```
9
-1
3
21
```

vii. index():

- index() method is exactly same as find() method except that if the specified substring is not available then we will get ValueError.

In [14]:

```
s = 'abbaaaaaaaaaaaaaaaaabbababa'
print(s.index('bb',2,15))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-14-136a30f266f9> in <module>
      1 s = 'abbaaaaaaaaaaaaaaaaabbababa'
----> 2 print(s.index('bb',2,15))

ValueError: substring not found
```

In [15]:

```
s = 'abbaaaaaaaaaaaaaaaaabbababa'
print(s.index('bb'))
```

```
1
```

In [17]:

```
s=input("Enter main string:")
subs=input("Enter sub string:")
try:
    n=s.index(subs)
except ValueError:
    print("substring not found")
else:
    print("substring found")
```

```
Enter main string:SMVDU
Enter sub string:MVD
substring found
```

In [18]:

```
s=input("Enter main string:")
subs=input("Enter sub string:")
try:
    n=s.index(subs)
except ValueError:
    print("substring not found")
else:
    print("substring found")
```

```
Enter main string:SMVDU
Enter sub string:MDV
substring not found
```

viii. rindex():

In [16]:

```
s = 'abbaaaaaaaaaaaaaaaaabbababa'
print(s.rindex('bb'))
```

```
20
```

ix. count():

- We can find the number of occurrences of substring present in the given string by using **count()** method.

Different forms of count() function/method:

1. s.count(substring) ==> It will search through out the string
2. s.count(substring, begin, end) ===> It will search from begin index to end-1 index

In [19]:

```
s="abcabcabcabcadda"
print(s.count('a')) #6
print(s.count('ab')) #4
print(s.count('a',3,7)) #2
```

```
6
4
2
```

In [20]:

```
s = 'abcdcdckk'
print(s.count('cdc'))
```

```
1
```

Q. Write a Python Program to display all positions of substring in a given main string.

In [21]:

```
s=input("Enter main string:")
subs=input("Enter sub string:")
flag=False
pos=-1
n=len(s)
c = 0
while True:
    pos=s.find(subs,pos+1,n)
    if pos==-1:
        break
    c = c+1
    print("Found at position",pos)
    flag=True
if flag==False:
    print("Not Found")
print('The number of occurrences : ',c)
```

```
Enter main string:abcabcabcaaa
Enter sub string:abc
Found at position 0
Found at position 3
Found at position 6
The number of occurrences :  3
```

In [22]:

```
s=input("Enter main string:")
subs=input("Enter sub string:")
flag=False
pos=-1
n=len(s)
c = 0
while True:
    pos=s.find(subs,pos+1,n)
    if pos==-1:
        break
    c = c+1
    print("Found at position",pos)
    flag=True
if flag==False:
    print("Not Found")
print('The number of occurrences : ',c)
```

```
Enter main string:bb
Enter sub string:a
Not Found
The number of occurrences :  0
```

In [23]:

```
s=input("Enter main string:")
subs=input("Enter sub string:")
flag=False
pos=-1
n=len(s)
c = 0
while True:
    pos=s.find(subs,pos+1,n)
    if pos==-1:
        break
    c = c+1
    print("Found at position",pos)
    flag=True
if flag==False:
    print("Not Found")
print('The number of occurrences : ',c)
```

```
Enter main string:abcabcabcaaa
Enter sub string:bb
Not Found
The number of occurrences :  0
```

Alternate Way:

In [24]:

```
s=input("Enter main string:")
subs=input("Enter sub string:")
i = s.find(subs)
if i == -1:
    print('Specified Substring is not found')
    c = 0
while i !=- 1:
    c = c + 1
    print('{} is present at index: {}'.format(subs,i))
    i = s.find(subs,i+len(subs),len(s))
print('The number of occurrences : ',c)
```

```
Enter main string:Python Programming
Enter sub string:ram
ram is present at index: 11
The number of occurrences :  1
```

In [25]:

```
s=input("Enter main string:")
subs=input("Enter sub string:")
i = s.find(subs)
if i == -1:
    print('Specified Substring is not found')
    c = 0
while i !=- 1:
    c = c + 1
    print('{} is present at index: {}'.format(subs,i))
    i = s.find(subs,i+len(subs),len(s))
print('The number of occurrences : ',c)
```

```
Enter main string:Python Programming
Enter sub string:raj
Specified Substring is not found
The number of occurrences :  0
```

x. replace():

- We can repalce a string with another string in python using a library function **replace().**

Syntax:

```
s.replace(oldstring,newstring)
```

Here, inside 's', every occurrence of oldstring will be replaced with new string.

In [26]:

```
s="Learning Python is very difficult"
s1=s.replace("difficult","easy")
print(s1)
```

```
Learning Python is very easy
```

In [28]:

```
s="abababababab"
print(id(s))
s1=s.replace("a","b")  # All occurrences will be replaced
print(id(s1))
print(s1)
```

```
2374771477808
2374771516016
bbbbbbbbbbbbb
```

In [29]:

```
s="abababababab"
print(id(s))
s=s.replace("a","b") # two objcets are created
print(id(s))
print(s)
```

```
2374771477808
2374771517552
bbbbbbbbbbbbb
```

Q. String objects are immutable then how we can change the content by using replace() method.

Ans: Once we creates string object, we cannot change the content.This non changeable behaviour is nothing but immutability. If we are trying to change the content by using any method, then with those changes a new object will be created and changes won't be happend in existing object.

Hence with replace() method also a new object got created but existing object won't be changed.

In [30]:

```
s="abab"
s1=s.replace("a","b")
print(s,"is available at :",id(s))
print(s1,"is available at :",id(s1))
```

```
abab is available at : 2374771519408
bbbb is available at : 2374771517552
```

In the above example, original object is available and we can see new object which was created because of replace() method.

Eg : Consider the string : Python is easy but Java is difficult.

How can you replace the string 'difficult' with 'easy' and 'easy' with 'difficult'?

In [31]:

```
s = 'Python is easy but Java is difficult'
s = s.replace('difficult','easy')
s = s.replace('easy','difficult')
print(s) # it is not giving correct output
```

```
Python is difficult but Java is difficult
```

In [32]:

```
s = 'Python is easy but Java is difficult'
s = s.replace('difficult','d1')
s = s.replace('easy','e1')
print(s)
```

```
Python is e1 but Java is d1
```

In [33]:

```
s = 'Python is easy but Java is difficult'
s = s.replace('difficult','d1')
s = s.replace('easy','e1')
s = s.replace('d1','easy')
s = s.replace('e1','difficult')
print(s)
```

```
Python is difficult but Java is easy
```

xi. split():

- We can split the given string according to specified seperator by using **split()** method.
- We can split the given string according to specified seperator in reverse direction by using **rsplit()** method.

Syntax :

```
l=s.split(seperator, Maximum splits)
```

Here,

- Both parameters are optional.
- The default seperator is space.
- Maximum split defines maximum number of splits
- The return type of split() method is List.

Note:

- rsplit() breaks the string at the seperator staring from the right and returns a list of strings.

In [34]:

```
s="cse ece eee"
l=s.split()
for x in l:
    print(x)
```

```
cse
ece
eee
```

In [35]:

```
s="22-02-2018"
l=s.split('-')
for x in l:
    print(x)
```

```
22
02
2018
```

In [36]:

```
s="22-02-2018"
l=s.split() # no space in the string , so output is same as the given string
for x in l:
    print(x)
```

```
22-02-2018
```

In [37]:

```
s = 'rgm nandyal cse ece eee'
l=s.split()
for x in l:
    print(x)
```

```
rgm
nandyal
cse
ece
eee
```

In [38]:

```
s = 'rgm nandyal cse ece eee'
l=s.rsplit(' ',3)
for x in l:
    print(x)
```

```
rgm nandyal
cse
ece
eee
```

In [39]:

```
s = 'rgm nandyal cse ece eee me ce'
l=s.rsplit(' ',3)
for x in l:
    print(x)
```

```
rgm nandyal cse ece
eee
me
ce
```

In [40]:

```
s = 'rgm nandyal cse ece eee me ce'
l=s.lsplit(' ',3)
for x in l:
    print(x)
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
<ipython-input-40-24e277deda26> in <module>
      1 s = 'rgm nandyal cse ece eee me ce'
----> 2 l=s.lsplit(' ',3)
      3 for x in l:
      4     print(x)

AttributeError: 'str' object has no attribute 'lsplit'
```

In [41]:

```
s = '10,20,30,40,50,60,70,80'
l = s.split(',',3)
for x in l:
    print(x)
```

```
10
20
30
40,50,60,70,80
```

In [42]:

```
s = '10,20,30,40,50,60,70,80'
l = s.rsplit(',',3)
for x in l:
    print(x)
```

```
10,20,30,40,50
60
70
80
```

In [43]:

```
s = '10,20,30,40,50,60,70,80'
l = s.split(',',-1)
for x in l:
    print(x)
```

```
10
20
30
40
50
60
70
80
```

xii. join():

- We can join a group of strings(list or tuple) with respect to the given seperator.

Syntax:

```
s=seperator.join(group of strings)
```

In [44]:

```
t=('sunny','bunny','chinny')
s='-'.join(t)
print(s)
```

```
sunny-bunny-chinny
```

In [45]:

```
l=['hyderabad','singapore','london','dubai']
s=':'.join(l)
print(s)
```

```
hyderabad:singapore:london:dubai
```

In [46]:

```
l=['hyderabad','singapore','london','dubai']
s=''.join(l)
print(s)
```

```
hyderabadsingaporelondondubai
```

In [47]:

```
l=['hyderabad','singapore','london','dubai']
s=' '.join(l)
print(s)
```

```
hyderabad singapore london dubai
```

Changing case of a String:

- We can change case of a string by using the following methods.

xiii. upper():

- Used to convert all characters to upper case in the given string.

xiv. lower():

- Used to convert all characters to lower case in the given string.

xv. swapcase():

- Used to convert all lower case characters to upper case and all upper case characters to lower case in the given string.

xvi. title():

- Used to convert all characters to title case. (i.e first character in every word should be upper case and all remaining characters should be in lower case in the given string).

xvii. capitalize():

- Only first character will be converted to upper case and all remaining characters can be converted to lower case.

In [48]:

```
s='learning Python is very Easy'
print(s.upper())
print(s.lower())
print(s.swapcase())
print(s.title())
print(s.capitalize())
```

```
LEARNING PYTHON IS VERY EASY
learning python is very easy
LEARNING pYTHON IS VERY eASY
Learning Python Is Very Easy
Learning python is very easy
```

Q. Write a Python program to Convert the uppercase characters into lowercase and remove spaces.

In [49]:

```
s='Learning Python Is Very Easy'
s = s.lower().replace(' ','')
print(s)
```

```
learningpythonisveryeasy
```

In [50]:

```
# Above example with join() & split() functions
s='Learning Python Is Very Easy'
s = s.lower()
s1 = s.split()
s = ''.join(s1)
print(s)
```

```
learningpythonisveryeasy
```

Checking starting and ending part of the string:

Python contains the following methods for this purpose.

1. s.startswith(substring)
2. s.endswith(substring)

xviii. startswith():

- Used to check the starting of the string.

xix. endswith():

- Used to check the ending of the string.

In [51]:

```
s='learning Python is very easy'
print(s.startswith('learning'))
print(s.endswith('learning'))
print(s.endswith('easy'))
```

```
True
False
True
```

Good Luck

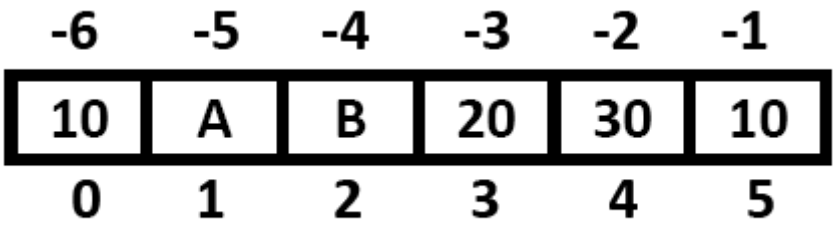
Experiment 7: List Data type

a) Demonstrate the different ways of creating list objects with suitable example programs.

If we want to represent a group of individual objects as a single entity where insertion order is preserved and duplicates are allowed, then we should go for List.

- Insertion order preserved.
- Duplicate objects are allowed
- Heterogeneous objects are allowed.
- List is dynamic because based on our requirement we can increase the size and decrease the size.
- In List the elements will be placed within square brackets and with comma seperator.
- We can differentiate duplicate elements by using index and we can preserve insertion order by using index.Hence index will play very important role.
- Python supports both positive and negative indexes. +ve index means from left to right where as negative index means right to left.

Eg: [10,"A","B",20, 30, 10]



- List objects are mutable.(i.e., we can change the content.)

Creation of List Objects:

1. We can create empty list object:

In [1]:

```
list=[]
print(list)
print(type(list))
```

```
[]
<class 'list'>
```

2. If we know elements already then we can create list object:

In [2]:

```
list = [10,20,30,40]
print(list)
print(type(list))
```

```
[10, 20, 30, 40]
<class 'list'>
```

3. Creation of list object With dynamic input:

In [3]:

```
list=(input("Enter List:")) # Entire input is considered as string
print(list)
print(type(list))
```

```
Enter List:10,20,30,40
10,20,30,40
<class 'str'>
```

In [4]:

```
list=eval(input("Enter List:"))
print(list)
print(type(list))
```

```
Enter List:[10,20,30,40]
[10, 20, 30, 40]
<class 'list'>
```

In [5]:

```
list=eval(input("Enter List:"))
print(list)
print(type(list))
```

```
Enter List:[ram,raj]

---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-5-5a3b608c2f72> in <module>
----> 1 list=eval(input("Enter List:"))
      2 print(list)
      3 print(type(list))

<string> in <module>

NameError: name 'ram' is not defined
```

In [6]:

```
list=eval(input("Enter List:"))
print(list)
print(type(list))
```

```
Enter List:['ram','raj']
['ram', 'raj']
<class 'list'>
```

4. We can create a list object using list() function:

In [8]:

```
l=list(range(0,10,2))
print(l)
# Not working in jupyter notebook but works in any standard editor
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-8-70c9fdd81115> in <module>
----> 1 l=list(range(0,10,2))
      2 print(l)
      3 # Not working in jupyter notebook but works in any standard editor

TypeError: 'list' object is not callable
```

In [9]:

```
[0, 2, 4, 6, 8]
```

Out[9]:

```
[0, 2, 4, 6, 8]
```

In [ ]:

```
s="rgmiet"
l=list(s)
print(l)
```

In [10]:

```
['r','g','m','c','e','t']
```

Out[10]:

```
['r', 'g', 'm', 'c', 'e', 't']
```

5. We can create list object with split() function:

In [11]:

```
s="Learning Python is very very easy !!!"
l=s.split()
print(l)
print(type(l))
```

```
['Learning', 'Python', 'is', 'very', 'very', 'easy', '!!!']
<class 'list'>
```

b) Demonstrate the following functions/methods which operates on lists in Python with suitable examples.

```
i) list( )     ii) len( )       iii) count( )    iv) index ( )

v) append( )   vi) insert( )    vii) extend()    viii) remove( )

ix) pop( )      x) reverse( )     xi) sort( )    xii) copy( )

xiii) clear( )
```

Important functions of List:

i. list():

In [ ]:

```
l=list(range(0,10,2))
print(l)
# Not working in jupyter notebook but works in any standard editor
```

ii) len():

- It returns the number of elements present in the list.

In [1]:

```
n=[10,20,30,40]
print(len(n))
```

4

In [2]:

```
n=[10,20,30,40,'rgm']
print(len(n))
```

5

iii) count():

- It returns the number of occurrences of specified item in the list.

In [3]:

```
n=[1,2,2,2,2,3,3]
print(n.count(1))
print(n.count(2))
print(n.count(3))
print(n.count(4))
```

```
1
4
2
0
```

iv) index():

- It returns the index of first occurrence of the specified item.

In [4]:

```
n=[1,2,2,2,2,3,3]
print(n.index(1)) # 0
print(n.index(2)) # 1
print(n.index(3)) # 5
print(n.index(4))
```

```
0
1
5

---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-4-b7a2785b40f1> in <module>
      3 print(n.index(2)) # 1
      4 print(n.index(3)) # 5
----> 5 print(n.index(4))

ValueError: 4 is not in list
```

Note:

If the specified element not present in the list then we will get **ValueError**. Hence before index() method we have to check whether item present in the list or not by using in operator.

Example Program:

In [5]:

```
l = [10,20,30,40,10,20,10,10]
target = int(input('Enter value to search : '))
if target in l:
    print(target,'available and its first occurrence is at ',l.index(target))
else:
    print(target,' is not available')
```

```
Enter value to search : 500
500  is not available
```

In [6]:

```
l = [10,20,30,40,10,20,10,10]
target = int(input('Enter value to search : '))
if target in l:
    print(target,'available and its first occurrence is at ',l.index(target))
else:
    print(target,' is not available')
```

```
Enter value to search : 20
20 available and its first occurrence is at  1
```

In [7]:

```
l = [10,20,30,40,10,20,10,10]
target = int(input('Enter value to search : '))
if target in l:
    print(target,'available and its first occurrence is at ',l.index(target))
else:
    print(target,' is not available')
```

```
Enter value to search : 10
10 available and its first occurrence is at  0
```

v) append():

- We can use append() function to add item at the end of the list.
- By using this append function, we always add an element at last position.

In [8]:

```
list=[]
list.append("A")
list.append("B")
list.append("C")
print(list)
```

```
['A', 'B', 'C']
```

Q. Write a Python Program to add all elements to list upto 100 which are divisible by 10.

In [9]:

```
list=[]
for i in range(101):
    if i%10==0:
        list.append(i)
print(list)
```

```
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

Another Way:

In [10]:

```
list= []
for i in range(0,101,10):
    list.append(i)
print(list)
```

```
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

vi) insert():

- It is used to insert item at specified index position.

In [11]:

```
n=[1,2,3,4,5]
n.insert(1,888)
print(n)
```

```
[1, 888, 2, 3, 4, 5]
```

In [12]:

```
n=[1,2,3,4,5]
n.insert(10,777)
n.insert(-10,999)
print(n)
print(n.index(777))
print(n.index(999))
```

```
[999, 1, 2, 3, 4, 5, 777]
6
0
```

Note:

- If the specified index is greater than max index then element will be inserted at last position.
- If the specified index is smaller than min index then element will be inserted at first position.

Differences between append() and insert()

append()	insert()
In List when we add any element it will come in last i.e. it will be last element.	In List we can insert any element in particular index number

vii) extend():

- If we waant to add all items of one list to another list,we use **extend()** method.

Eg:

```
l1.extend(l2)
```

- All items present in **l2** will be added to **l1**.

In [13]:

```
order1=["Chicken","Mutton","Fish"]
order2=["RC","KF","FO"]
order1.extend(order2)
print(order1)
print(order2)
```

```
['Chicken', 'Mutton', 'Fish', 'RC', 'KF', 'FO']
['RC', 'KF', 'FO']
```

In [14]:

```
order1=["Chicken","Mutton","Fish"]
order2=["RC","KF","FO"]
order3 = order1 + order2
print(order1)
print(order2)
print(order3)
```

```
['Chicken', 'Mutton', 'Fish']
['RC', 'KF', 'FO']
['Chicken', 'Mutton', 'Fish', 'RC', 'KF', 'FO']
```

In [15]:

```
l1 = [10,20,30]
l2 = [40,50,60]
l1.extend(l2)
print(l1)
```

```
[10, 20, 30, 40, 50, 60]
```

In [16]:

```
order=["Chicken","Mutton","Fish"]
order.extend("Mushroom")
print(order) # It adds every character as a single element to the list
```

```
['Chicken', 'Mutton', 'Fish', 'M', 'u', 's', 'h', 'r', 'o', 'o', 'm']
```

Explanation:

- Here, 'Mushroom' is a string type, in this string 8 elements are there. These elements are added seperately.

In [17]:

```
order=["Chicken","Mutton","Fish"]
order.append("Mushroom") # It adds this string as a single element to the list
print(order)
```

```
['Chicken', 'Mutton', 'Fish', 'Mushroom']
```

viii) remove():

- We can use this function to remove specified item from the list.
- If the item present multiple times then only first occurrence will be removed.

In [18]:

```
n=[10,20,10,30]
n.remove(10)
print(n)
```

```
[20, 10, 30]
```

- If the specified item not present in list then we will get **ValueError**.

In [19]:

```
n=[10,20,10,30]
n.remove(40)
print(n)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-19-c4e183b90359> in <module>
      1 n=[10,20,10,30]
----> 2 n.remove(40)
      3 print(n)

ValueError: list.remove(x): x not in list
```

Note:

Hence before using remove() method first we have to check specified element present in the list or not by using **in** operator.

In [20]:

```
l1= [10,20,30,40,50,60,70]
x = int(input('Enter the element to be removed : '))
if x in l1:
    l1.remove(x)
    print('Element removed Successfully ')
    print(l1)
else:
    print('Specified element is not available ')
```

```
Enter the element to be removed : 10
Element removed Successfully
[20, 30, 40, 50, 60, 70]
```

In [21]:

```
l1= [10,20,30,40,50,60,70]
x = int(input('Enter the element to be removed : '))
if x in l1:
    l1.remove(x)
    print('Element removed Successfully ')
    print(l1)
else:
    print('Specified element is not available ')
```

```
Enter the element to be removed : 90
Specified element is not available
```

ix) pop():

- It removes and returns the last element of the list.
- This is only function which manipulates list and returns some element.

In [22]:

```
n=[10,20,30,40]
print(n.pop())
print(n.pop())
print(n)
```

```
40
30
[10, 20]
```

- If the list is empty then pop() function raises **IndexError**.

In [23]:

```
n=[]
print(n.pop())
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-23-bfeb9843d2be> in <module>
      1 n=[]
----> 2 print(n.pop())

IndexError: pop from empty list
```

Note:

1. **pop()** is the only function which manipulates the list and returns some value.
2. In general we can use append() and pop() functions to implement stack datastructure by using list,which follows LIFO(Last In First Out) order.
3. In general we can use pop() function to remove last element of the list. But we can also use pop() function to remove elements based on specified index.

We can use pop() function in following two ways:

- n.pop(index) ==> To remove and return element present at specified index.
- n.pop() ==> To remove and return last element of the list.

In [24]:

```
n=[10,20,30,40,50,60]
print(n.pop()) #60
print(n.pop(1)) #20
print(n.pop(10)) # IndexError: pop index out of range
```

```
60
20

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-24-47504bb9f619> in <module>
      2 print(n.pop()) #60
      3 print(n.pop(1)) #20
----> 4 print(n.pop(10)) # IndexError: pop index out of range

IndexError: pop index out of range
```

Differences between remove() and pop()

remove()	pop()
1) We can use to remove special element from the List.	1) We can use to remove last element from the List.
2) It can't return any value.	2) It returned removed element.
3) If special element not available then we get VALUE ERROR.	3) If List is empty then we get Index Error.

Note: In the above table, wherever **special** is there, consider it as **specific**.

Note:

List objects are dynamic (i.e., based on our requirement we can increase and decrease the size).

- **append(),insert(),extend()** ===>for increasing the size/growable nature
- **remove(),pop()** ======>for decreasing the size /shrinking nature

x) reverse():

- It is used to reverse the order of elements in the list.

In [25]:

```
n=[10,20,30,40]
n.reverse()
print(n)
```

```
[40, 30, 20, 10]
```

xi) sort():

- In list by default insertion order is preserved.
- If you want to sort the elements of list according to default natural sorting order then we should go for **sort()** method.

```
For numbers ==> default natural sorting order is Ascending Order

For Strings ==> default natural sorting order is Alphabetical Order
```

In [26]:

```
n=[20,5,15,10,0]
n.sort()
print(n)
```

```
[0, 5, 10, 15, 20]
```

In [27]:

```
s=["Dog","Banana","Cat","Apple"]
s.sort()
print(s)
```

```
['Apple', 'Banana', 'Cat', 'Dog']
```

In [28]:

```
s=["Dog","Banana","Cat","apple"]
s.sort() # Unicode values are used during comparison of alphbets
print(s)
```

```
['Banana', 'Cat', 'Dog', 'apple']
```

Note:

- To use sort() function, compulsory list should contain only homogeneous elements, otherwise we will get **TypeError.**

In [29]:

```
n=[20,10,"A","B"]
n.sort()
print(n)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-29-76d0c10ed9e7> in <module>
      1 n=[20,10,"A","B"]
----> 2 n.sort()
      3 print(n)

TypeError: '<' not supported between instances of 'str' and 'int'
```

How to sort the elements of list in reverse of default natural sorting order?

One Simple Way:

In [30]:

```
n=[40,10,30,20]
n.sort()
n.reverse()
print(n)
```

```
[40, 30, 20, 10]
```

Alternate Way:

- We can sort according to reverse of default natural sorting order by using **reverse = True** argument.

In [31]:

```
n=[40,10,30,20]
n.sort()
print(n) #[10,20,30,40]
n.sort(reverse=True)
print(n) #[40,30,20,10]
n.sort(reverse=False)
print(n) #[10,20,30,40]
```

```
[10, 20, 30, 40]
[40, 30, 20, 10]
[10, 20, 30, 40]
```

In [32]:

```
s=["Dog","Banana","Cat","Apple"]
s.sort(reverse= True) # reverse of Alphabetical order
print(s)
```

```
['Dog', 'Cat', 'Banana', 'Apple']
```

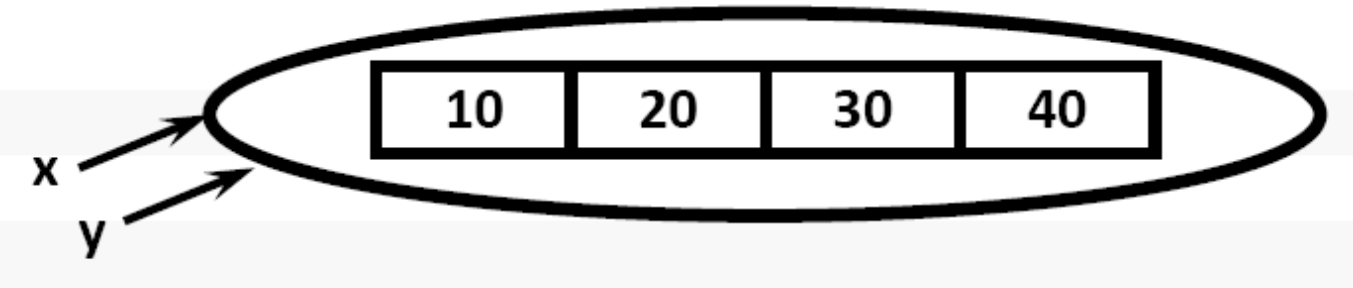
Aliasing and Cloning of List objects:

- The process of giving another reference variable to the existing list is called aliasing.

In [33]:

```
x=[10,20,30,40]
y=x
print(id(x))
print(id(y))
```

```
1979461271296
1979461271296
```

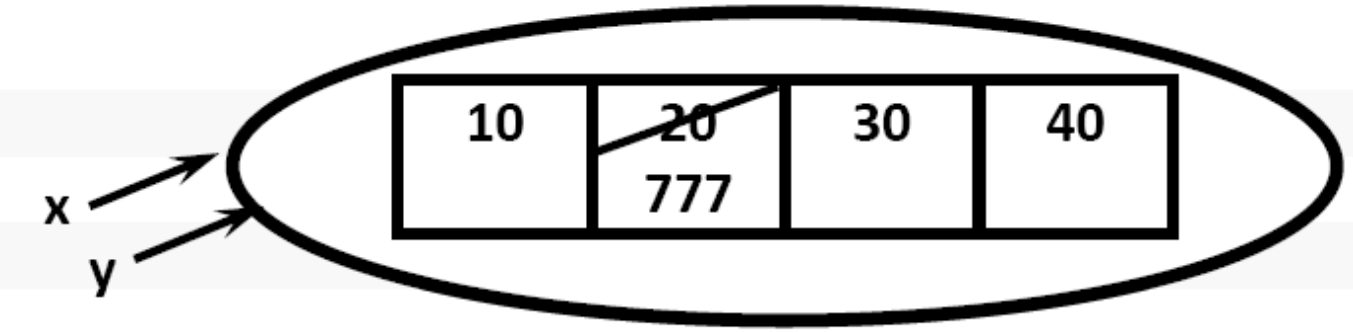


The problem in this approach is by using one reference variable if we are changing content,then those changes will be reflected to the other reference variable.

In [34]:

```
x=[10,20,30,40]
y=x
y[1]=777
print(x)
```

```
[10, 777, 30, 40]
```



To overcome this problem we should go for **cloning.**

Cloning: The process of creating exactly duplicate independent object is called cloning.

We can implement cloning by using the following ways:

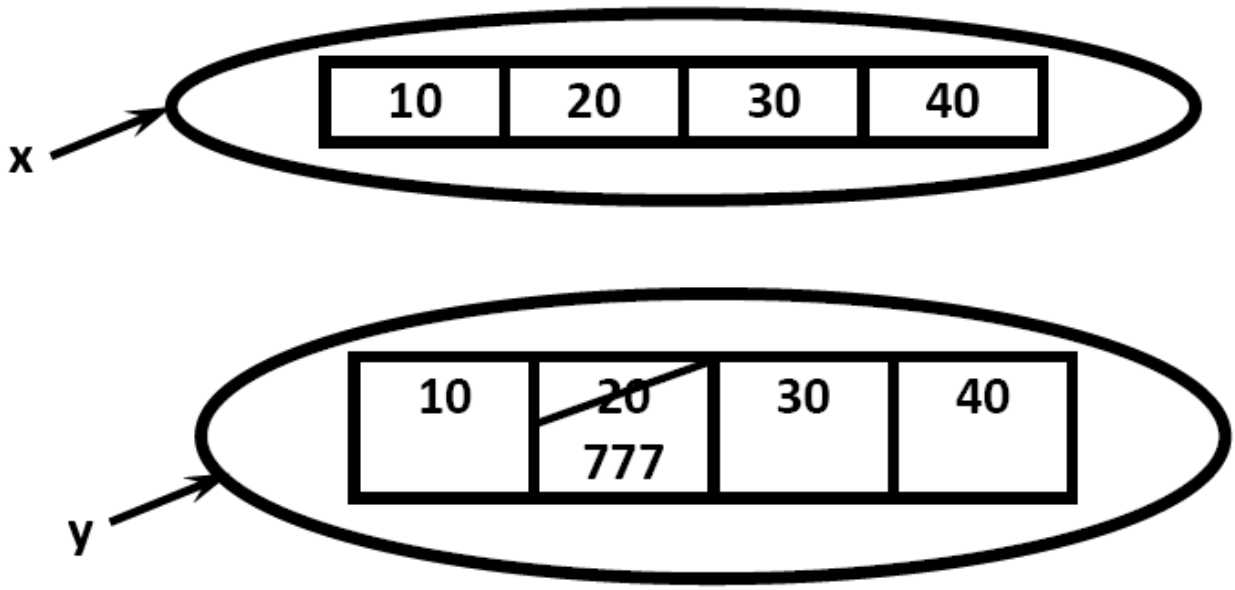
1. slice operator
2. copy() function

1. By using slice operator:

In [35]:

```
x=[10,20,30,40]
y=x[:]
y[1]=777
print(x) #[10,20,30,40]
print(y) #[10,777,30,40]
```

```
[10, 20, 30, 40]
[10, 777, 30, 40]
```



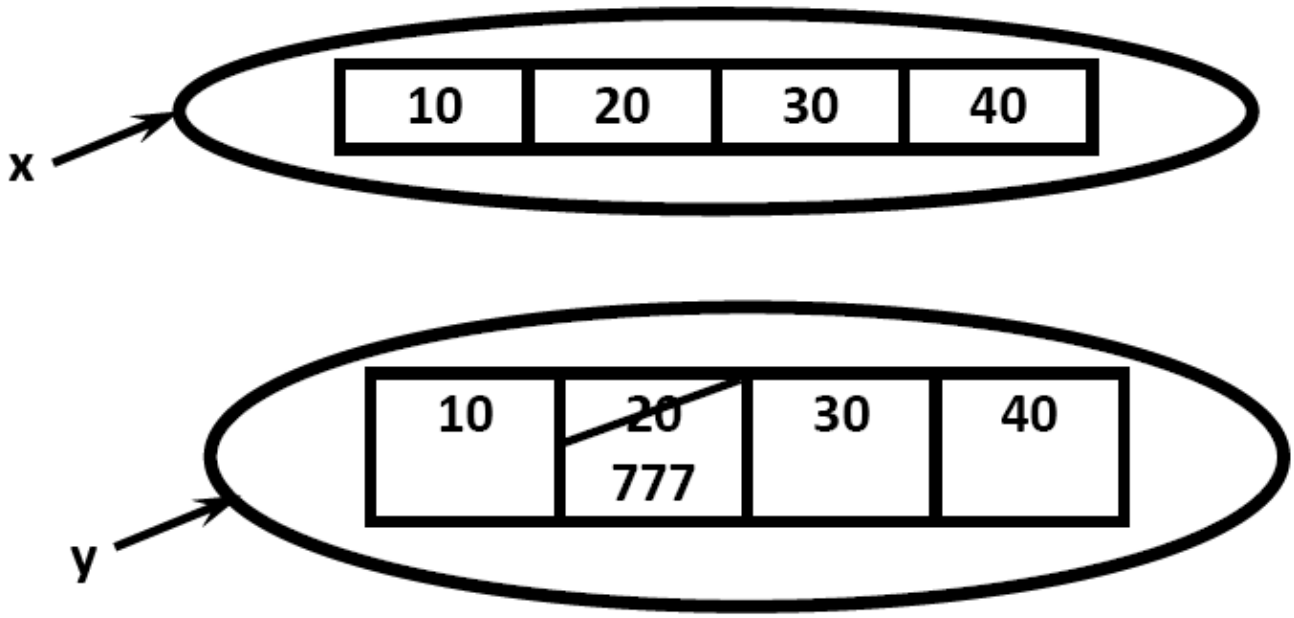
2. By using copy() function:

xii) copy():

In [36]:

```
x=[10,20,30,40]
y=x.copy()
y[1]=777
print(x) # [10,20,30,40]
print(y) # [10,777,30,40]
```

```
[10, 20, 30, 40]
[10, 777, 30, 40]
```



Q. What is the difference between = operator and copy() function?

Ans: = operator meant for aliasing copy() function meant for cloning.

xiii) clear():

- We can use **clear()** function to remove all elements of List.

In [37]:

```
n=[10,20,30,40]
print(n)
n.clear()
print(n)
```

```
[10, 20, 30, 40]
[]
```

c) Demonstrate the following with suitable example programs:

```
i) List slicing

ii) List Comprehensions
```

i) List slicing:

Syntax:

list2= list1[start:stop:step]

- start ==>it indicates the index where slice has to start default value is 0
- stop ===>It indicates the index where slice has to end default value is max allowed index of list ie length of the list
- step ==>increment value (step default value is 1)

In [1]:

```
l = [10,20,30,40,50,60]
print(l[::])
```

```
[10, 20, 30, 40, 50, 60]
```

In [2]:

```
l = [10,20,30,40,50,60]
l1=l[::]
print(l1)
```

```
[10, 20, 30, 40, 50, 60]
```

In [3]:

```
l = [10,20,30,40,50,60]
print(l[::2])
```

```
[10, 30, 50]
```

In [4]:

```
l = [10,20,30,40,50,60]
print(l[::-1])
```

```
[60, 50, 40, 30, 20, 10]
```

In [5]:

```
l = [10,20,[30,40],50,60]
print(l[0:3:])
```

```
[10, 20, [30, 40]]
```

In [6]:

```
n=[1,2,3,4,5,6,7,8,9,10]
print(n[2:7:2])          #3,5,7
print(n[4::2])           # 5,7,9
print(n[3:7])            #4,5,6,7
print(n[8:2:-2])         # 9,7,5
print(n[4:100])          # 5,6,7,8,9,10
```

```
[3, 5, 7]
[5, 7, 9]
[4, 5, 6, 7]
[9, 7, 5]
[5, 6, 7, 8, 9, 10]
```

ii) List Comprehensions:

- It is very easy and compact way of creating list objects from any iterable objects(like list,tuple,dictionary,range etc) based on some condition.

Syntax:

```
list=[expression for item in list if condition]
```

Consider an example, If you want to store squares of numbers form 1 to 10 in a list,

In [7]:

```
l1=[]
for x in range(1,11):
    l1.append(x*x)
print(l1)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In the above case, the program consisting 4 lines of code. Now for the same purpose we will write the following code in more concised way.

In [8]:

```
l1 = [x*x for x in range(1,21)]
l2 = [x for x in l1 if x % 2 == 0]
print(l1)
print(l2)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289,
324, 361, 400]
[4, 16, 36, 64, 100, 144, 196, 256, 324, 400]
```

Few more examples on List comprehensions:

In [9]:

```
l1 = [x*x for x in range(1,11)]
print(l1)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [10]:

```
l =[2**x for x in range(1,11)]
print(l)
```

```
[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
```

In [11]:

```
l = [x for x in range(1,11) if x%2==0]
print(l)
```

```
[2, 4, 6, 8, 10]
```

In [12]:

```
l = [x for x in range(1,11) if x%2==1]
print(l)
```

```
[1, 3, 5, 7, 9]
```

In [13]:

```
l = [x**2 for x in range(1,11) if (x**2)%2==1]
print(l)
```

```
[1, 9, 25, 49, 81]
```

In [14]:

```
words=["Balaiah","Nag","Venkatesh","Chiranjeevi"]
l=[w[0] for w in words]
print(l)
```

```
['B', 'N', 'V', 'C']
```

In [15]:

```
words=["Balaiah","Nag","Venkatesh","Chiranjeevi"]
l=[w for w in words if len(w)>6]
print(l)
```

```
['Balaiah', 'Venkatesh', 'Chiranjeevi']
```

In [16]:

```
num1=[10,20,30,40]
num2=[30,40,50,60]
num3=[ i for i in num1 if i not in num2]
print(num3)
```

```
[10, 20]
```

In [17]:

```
words="the quick brown fox jumps over the lazy dog".split()
print(words)
l=[[w.upper(),len(w)] for w in words]
print(l)
```

```
['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
[['THE', 3], ['QUICK', 5], ['BROWN', 5], ['FOX', 3], ['JUMPS', 5], ['OVE
R', 4], ['THE', 3], ['LAZY', 4], ['DOG', 3]]
```

More Example Programs:

Q1. Write a Python program to find the maximum number of a list of numbers.

In [36]:

```
n=int(input("Enter the list size:"))
a=[]
for i in range(n):
    num=int(input("Enter the number"))
    a.append(num)
print (a)
max=a[0]
for i in range(n):
    if(max<a[i]):
        max=a[i]
print ("Maximum number in the list is : ",max)
```

```
Enter the list size:5
Enter the number33
Enter the number77
Enter the number221
Enter the number5
Enter the number7
[33, 77, 221, 5, 7]
maximum number in the list is :  221
```

Q2. Write a Python program to find the minimum number of a list of numbers.

In [39]:

```
n=int(input("Enter the list size:"))
a=[]
for i in range(n):
    num=int(input("Enter the number"))
    a.append(num)
print (a)
min=a[0]
for i in range(n):
    if(min > a[i]):
        min = a[i]
print ("Minimum number in the list is : ",min)
```

```
Enter the list size:5
Enter the number33
Enter the number6
Enter the number27
Enter the number7
Enter the number44
[33, 6, 27, 7, 44]
Minimum number in the list is :  6
```

Good Luck

Experiment 8: Tuple Data type

a) Demonstrate the different ways of creating tuple objects with suitable example programs.

Introduction:

1. Tuple is exactly same as List except that it is immutable. i.e., once we creates Tuple object,we cannot perform any changes in that object. Hence Tuple is Read Only Version of List.
2. If our data is fixed and never changes then we should go for Tuple.
3. Insertion Order is preserved.
4. Duplicates are allowed.
5. Heterogeneous objects are allowed.
6. We can preserve insertion order and we can differentiate duplicate objects by using index. Hence index will play very important role in Tuple also.
7. Tuple support both +ve and -ve index. +ve index means forward direction(from left to right) and -ve index means backward direction(from right to left).
8. We can represent Tuple elements within Parenthesis and with comma seperator.

Note:

- Parenethesis are optional but recommended to use.

In [1]:

```
t=10,20,30,40
print(t)
print(type(t))
```

```
(10, 20, 30, 40)
<class 'tuple'>
```

In [2]:

```
t=(10,20,30,40)
print(t)
print(type(t))
```

```
(10, 20, 30, 40)
<class 'tuple'>
```

In [3]:

```
t = ()
print(type(t))
```

```
<class 'tuple'>
```

Note:

We have to take special care about single valued tuple.compulsary the value should ends with comma,otherwise it is not treated as tuple.

In [4]:

```
t=(10)
print(t)
print(type(t))
```

```
10
<class 'int'>
```

In [5]:

```
t=(10,)
print(t)
print(type(t))
```

```
(10,)
<class 'tuple'>
```

Q. Which of the following are valid/Invalid tuples?

In [6]:

```
t=()                    # valid
t=10,20,30,40           # valid
t=10                    # not valid
t=10,                   # valid
t=(10)                  # notvalid
t=(10,)                 # valid
t=(10,20,30,40)         # valid
t= (10,20,30,)          # valid
```

In [7]:

```
t = (10,20,30,)
print(t)
print(type(t))
```

```
(10, 20, 30)
<class 'tuple'>
```

Creation of Tuple Objects:

1. We can create empty tuple object:

In [8]:

```
t=()
print(t)
print(type(t))
```

```
()
<class 'tuple'>
```

2. Creation of single valued tuple object:

In [9]:

```
t = (10,)
print(t)
print(type(t))
```

```
(10,)
<class 'tuple'>
```

3. creation of multi values tuples & parenthesis are optional:

In [10]:

```
t = 10,20,30
print(t)
print(type(t))
```

```
(10, 20, 30)
<class 'tuple'>
```

In [11]:

```
t=eval(input("Enter Tuple:"))
print(t)
print(type(t))
```

```
Enter Tuple:(10,20,30)
(10, 20, 30)
<class 'tuple'>
```

In [12]:

```
t=eval(input("Enter Tuple:"))
print(t)
print(type(t))
```

```
Enter Tuple:10,20,30
(10, 20, 30)
<class 'tuple'>
```

4. We can create a tuple object using tuple() function:

If you have any sequence (i.e., string, list, range etc.,) which can be easily converted into a tuple by using **tuple()** function.

In [13]:

```
list=[10,20,30]
t=tuple(list)
print(t)
print(type(t))
```

```
(10, 20, 30)
<class 'tuple'>
```

In [14]:

```
t=tuple(range(10,20,2))
print(t)
print(type(t))
```

```
(10, 12, 14, 16, 18)
<class 'tuple'>
```

In [15]:

```
t = tuple('karthi')
print(t)
print(type(t))
```

```
('k', 'a', 'r', 't', 'h', 'i')
<class 'tuple'>
```

b) Demonstrate the following functions/methods which operates on tuples in Python with suitable examples.

```
i) len()     ii) count()    iii) index ()     iv) sorted()

v) min()     vi) max()      vii) cmp()        viii) reversed()
```

Important functions of Tuple:

i. len():

- It is an in-built function of Python, if you provide any sequnce (i.e., strings, list,tuple etc.,), in that how many elements are there that will be returned this function.
- It is used to return number of elements present in the tuple.

In [16]:

```
t=(10,20,30,40)
print(len(t))         # 4
```

```
4
```

ii) count():

- It returns the number of occurrences of specified item in the list.

In [17]:

```
t=(10,20,10,10,20)
print(t.count(10)) #3
```

```
3
```

In [18]:

```
n=(1,2,2,2,2,3,3)
print(n.count(1))
print(n.count(2))
print(n.count(3))
print(n.count(4))
```

```
1
4
2
0
```

In [19]:

```
t=(10,20,10,10,20)
print(t.count(100))
```

```
0
```

iii) index():

- It returns the index of first occurrence of the specified item.
- If the specified element is not available then we will get **ValueError.**

In [20]:

```
t=(10,20,10,10,20)
print(t.index(10)) # 0
print(t.index(30)) # ValueError: tuple.index(x): x not in tuple
```

```
0

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-20-c98072186526> in <module>
      1 t=(10,20,10,10,20)
      2 print(t.index(10)) # 0
----> 3 print(t.index(30)) # ValueError: tuple.index(x): x not in tuple

ValueError: tuple.index(x): x not in tuple
```

In [21]:

```
n=(1,2,2,2,2,3,3)
print(n.index(1)) # 0
print(n.index(2)) # 1
print(n.index(3)) # 5
print(n.index(4))
```

```
0
1
5

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-21-e2f70118d739> in <module>
      3 print(n.index(2)) # 1
      4 print(n.index(3)) # 5
----> 5 print(n.index(4))

ValueError: tuple.index(x): x not in tuple
```

iv) sorted():

- It is used to sort elements based on default natural sorting order (Ascending order).

In [22]:

```
t =(10,30,40,20)
print(sorted(t)) # sorted() is going to return list
```

```
[10, 20, 30, 40]
```

In [23]:

```
t =(10,30,40,20)
t.sort()
print(t)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-23-6dd56d99cf24> in <module>
      1 t =(10,30,40,20)
----> 2 t.sort()
      3 print(t)

AttributeError: 'tuple' object has no attribute 'sort'
```

In [24]:

```
t=(40,10,30,20)
t1=tuple(sorted(t))
print(type(t1))
print(t1)
print(type(t1))
print(t)
```

```
<class 'tuple'>
(10, 20, 30, 40)
<class 'tuple'>
(40, 10, 30, 20)
```

- We can sort according to reverse of default natural sorting order is as follows:

In [25]:

```
t=(40,10,30,20)
t1=tuple(sorted(t))
t1=sorted(t,reverse=True)
print(t1)           #[40, 30, 20, 10]
```

```
[40, 30, 20, 10]
```

v) min():

- min() function return the minimum value according to default natural sorting order.
- This function will works on tuple with respect to homogeneous elements only.

In [26]:

```
t=(40,10,30,20)
print(min(t)) #10
```

```
10
```

In [27]:

```
t = ('karthi') # based on unicode values these functions will work.
print(min(t))
```

```
a
```

In [28]:

```
t = ('kArthi')
print(min(t))
```

```
A
```

vi) max():

- max() function return the maximum value according to default natural sorting order.
- This function will works on tuple with respect to homogeneous elements only.

In [29]:

```
t=(40,10,30,20)
print(max(t)) #40
```

```
40
```

In [30]:

```
t = ('karthi') # based on unicode values these functions will work.
print(max(t))
```

```
t
```

In [31]:

```
t = ('kArthi')
print(max(t))
```

```
t
```

vii) cmp():

- It compares the elements of both tuples.
- If both tuples are equal then returns 0.
- If the first tuple is less than second tuple then it returns -1.
- If the first tuple is greater than second tuple then it returns +1.

In [32]:

```
t1=(10,20,30)
t2=(40,50,60)
t3=(10,20,30)
print(cmp(t1,t2)) # -1
print(cmp(t1,t3)) # 0
print(cmp(t2,t3)) # +1
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-32-848450ec0e9e> in <module>
      2 t2=(40,50,60)
      3 t3=(10,20,30)
----> 4 print(cmp(t1,t2)) # -1
      5 print(cmp(t1,t3)) # 0
      6 print(cmp(t2,t3)) # +1

NameError: name 'cmp' is not defined
```

Note: cmp() function is available only in Python 2 but not in Python 3.

In [33]:

```
t1=(10,20,30)
t2=(40,50,60)
t3=(10,20,30)
print(t1==t2)
print(t1==t3)
print(t2==t3)
print(t1<t2) # true, because it compares only first element.
```

```
False
True
False
True
```

In [47]:

```
t1=(10,20,30)
t2=(5,50,60)
print(t1<t2)
```

```
False
```

viii) reversed():

- It is used to reverse the elements of the given tuple.

In [49]:

```
t1=(10,20,30)
t2 = reversed(t1)
for i in t2:
    print(i,end=' ')
```

```
30 20 10
```

Good Luck

Experiment 9: Set Data type

a) Demonstrate the different ways of creating set objects with suitable example programs.

Introduction:

If we want to represent a group of unique values as a single entity then we should go for set.

Key features of Set Data Type:

1. Duplicates are not allowed.
2. Insertion order is not preserved.But we can sort the elements.
3. Indexing and slicing not allowed for the set.
4. Heterogeneous elements are allowed.
5. Set objects are mutable i.e once we creates set object we can perform any changes in that object based on our requirement.
6. We can represent set elements within curly braces and with comma seperation.
7. We can apply mathematical operations like union,intersection,difference etc on set objects.

Creation of Set Objects:

1.Creation of set object with single value:

In [1]:

```
s = {10}
print(type(s))
print(s)
```

```
<class 'set'>
{10}
```

2. Creation of set object with multiple values:

In [2]:

```
s = {30,40,10,5,20} # In the output order not preserved
print(type(s))
print(s)
```

```
<class 'set'>
{5, 40, 10, 20, 30}
```

In [3]:

```
s = {30,40,10,5,20} # In the output order not preserved
print(type(s))
print(s[0])
```

```
<class 'set'>

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-87d1e6948aef> in <module>
      1 s = {30,40,10,5,20} # In the output order not preserved
      2 print(type(s))
----> 3 print(s[0])

TypeError: 'set' object is not subscriptable
```

In [4]:

```
s = {30,40,10,5,20}  # In the output order not preserved
print(type(s))
print(s[0:6])
```

```
<class 'set'>

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-bf084a8b7575> in <module>
      1 s = {30,40,10,5,20}  # In the output order not preserved
      2 print(type(s))
----> 3 print(s[0:6])

TypeError: 'set' object is not subscriptable
```

3. Creation of set objects using set() function:

- We can create set objects by using set() function.

Syntax:

```
s=set(any sequence)
```

In [5]:

```
l = [10,20,30,40,10,20,10]
s=set(l)
print(s) # {40, 10, 20, 30} because duplicates are not allowed in set
```

```
{40, 10, 20, 30}
```

In [6]:

```
s=set(range(5))
print(s) #{0, 1, 2, 3, 4}
```

```
{0, 1, 2, 3, 4}
```

In [7]:

```
s = set('karthi')
print(s)
```

```
{'i', 'r', 'a', 'h', 'k', 't'}
```

In [8]:

```
s= set('aaabbbb')
print(s)
```

```
{'b', 'a'}
```

In [9]:

```
st=eval(input("Enter Set:"))
print(st)
print(type(st))
```

```
Enter Set:{10,20,30}
{10, 20, 30}
<class 'set'>
```

Note:

- While creating empty set we have to take special care. Compulsory we should use **set()** function.

s={} ==>It is treated as dictionary but not empty set.

In [10]:

```
s = {}
print(type(s))
```

```
<class 'dict'>
```

In [11]:

```
s = set() # set function without any arguments
print(s)
print(type(s))
```

```
set()
<class 'set'>
```

b) Demonstrate the following functions/methods which operates on sets in Python with suitable examples.

```
i) add()     ii) update()       iii)copy()    iv) pop()

v) remove()   vi)discard()    vii)clear()    viii)union()

ix)intersection()      x)difference()
```

Important functions of Set:

i. add():

- It Adds an item 'x' to the set.

In [12]:

```
s={10,20,30}
s.add(40);              # ';' is optional for python statements
print(s)                # {40, 10, 20, 30}
```

```
{40, 10, 20, 30}
```

In [13]:

```
s={10,20,30}
s.add('karthi'); # ';' is optional for python statements
print(s)
```

```
{10, 'karthi', 20, 30}
```

ii) update():

- This method is used to add multiple items to the set.
- Arguments are not individual elements and these are Iterable objects like List,range etc.
- All elements present in the given Iterable objects will be added to the set.

In [14]:

```
s={10,20,30}
s.update('karthi'); # ';' is optional for python statements
print(s)
```

```
{'i', 'r', 10, 'a', 'h', 20, 'k', 't', 30}
```

In [15]:

```
s={10,20,30}
l=[40,50,60,10]
s.update(l,range(5))
print(s)
```

```
{0, 1, 2, 3, 4, 40, 10, 50, 20, 60, 30}
```

In [16]:

```
s={10,20,30}
l=[40,50,60,10]
s.update(l,range(5),100)
print(s)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-16-96e519440e16> in <module>
      1 s={10,20,30}
      2 l=[40,50,60,10]
----> 3 s.update(l,range(5),100)
      4 print(s)

TypeError: 'int' object is not iterable
```

In [17]:

```
s={10,20,30}
l=[40,50,60,10]
s.update(l,range(5),'100')
print(s)
```

```
{0, 1, 2, 3, 4, '0', 40, 10, 50, '1', 20, 60, 30}
```

In [18]:

```
s={10,20,30}
l=[40,50,60,10]
s.update(l,range(5),'karthi')
print(s)
```

```
{0, 1, 2, 3, 4, 'i', 'r', 40, 10, 'a', 'h', 50, 20, 'k', 60, 't', 30}
```

In [19]:

```
s =set()
s.update(range(1,10,2),range(0,10,2))
print(s)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

Q1. What is the difference between add() and update() functions in set?

- We can use add() to add individual item to the Set,where as we can use update() function to add multiple items to Set.
- add() function can take only one argument where as update() function can take any number of arguments but all arguments should be iterable objects.

Q2. Which of the following are valid for set s?

1. s.add(10) ==> Valid
2. s.add(10,20,30) ==> TypeError: add() takes exactly one argument (3 given)
3. s.update(10) ==>TypeError: 'int' object is not iterable
4. s.update(range(1,10,2),range(0,10,2)) ==>Valid

iii) copy():

- It returns copy of the set. It is cloned object (Backup copy).

In [21]:

```
s={10,20,30}
s1=s.copy()
print(s1)
print(s)
```

```
{10, 20, 30}
{10, 20, 30}
```

iv) pop():

- It removes and returns some random element from the set.

In [22]:

```
s={40,10,30,20}
print(s)
print(s.pop())
print(s.pop())
print(s.pop())
print(s)
print(s.pop())
print(s)            # Empty set
print(s.pop())
```

```
{40, 10, 20, 30}
40
10
20
{30}
30
set()

---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
<ipython-input-22-3f6a1609f80b> in <module>
      7 print(s.pop())
      8 print(s)            # Empty set
----> 9 print(s.pop())

KeyError: 'pop from an empty set'
```

Consider the following case :

In [23]:

```
s={40,10,30,20}
print(s)
print(s.pop())
print(s.pop())
print(s)
```

```
{40, 10, 20, 30}
40
10
{20, 30}
```

In [24]:

```
s={40,10,30,20}
print(s)
print(s.pop())
print(s.pop())
print(s)
```

```
{40, 10, 20, 30}
40
10
{20, 30}
```

In [25]:

```
s={40,10,30,20}
print(s)
print(s.pop())
print(s.pop())
print(s)
```

```
{40, 10, 20, 30}
40
10
{20, 30}
```

Note:

How many times you may execute the code, the elements which are popped from the set in same order. The reason is ---

- All the elements of set are inserted based on some hashcode.
- If that order is fixed then it is always going to return one by one. But in which order these elements are inserted we don't know.

v) remove():

- It removes specified element from the set.
- If the specified element not present in the Set then we will get **KeyError.**

In [26]:

```
s={40,10,30,20}
s.remove(30)
print(s) # {40, 10, 20}
s.remove(50) # KeyError: 50
```

```
{40, 10, 20}

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-26-77437864d839> in <module>
      2 s.remove(30)
      3 print(s) # {40, 10, 20}
----> 4 s.remove(50) # KeyError: 50

KeyError: 50
```

vi) discard():

- It removes the specified element from the set.
- If the specified element not present in the set then we won't get any error.

In [1]:

```
s={10,20,30}
s.discard(10)
print(s) #{20, 30}
s.discard(50)
print(s) #{20, 30}
```

```
{20, 30}
{20, 30}
```

vii) clear():

- It is used to remove all elements from the Set.

In [2]:

```
s={10,20,30}
print(s)
s.clear()
print(s)
```

```
{10, 20, 30}
set()
```

viii) union():

x.union(y) ==> We can use this function to return all elements present in both x and y sets

Wecan perform union operation in two ways:

1. x.union(y) ==> by calling through union() method.

2. x|y ==> by using '|' operator.

This operation returns all elements present in both sets x and y (without duplicate elements).

In [3]:

```
x={10,20,30,40}
y={30,40,50,60}
print(x.union(y)) #{10, 20, 30, 40, 50, 60} #Order is not preserved
print(x|y) #{10, 20, 30, 40, 50, 60}
```

```
{40, 10, 50, 20, 60, 30}
{40, 10, 50, 20, 60, 30}
```

ix) intersection():

Wecan perform intersection operation in two ways:

1. x.intersection(y) ==> by calling through intersection() method.

2. x&y ==> by using '&' operator.

This operation returns common elements present in both sets x and y.

In [4]:

```
x={10,20,30,40}
y={30,40,50,60}
print(x.intersection(y)) #{40, 30}
print(x&y) #{40, 30}
```

```
{40, 30}
{40, 30}
```

x) difference():

Wecan perform difference operation in two ways:

1. x.difference(y) ==> by calling through difference() method.

2. x-y ==> by using '-' operator.

This operation returns the elements present in x but not in y.

In [5]:

```
x={10,20,30,40}
y={30,40,50,60}
print(x.difference(y)) #{10, 20}
print(x-y) #{10, 20}
print(y-x) #{50, 60}
```

```
{10, 20}
{10, 20}
{50, 60}
```

Good Luck

Experiment 10: Dictionary Data type

a) Demonstrate the different ways of creating Dictionary objects with suitable example programs.

Introduction:

- We can use List,Tuple and Set to represent a group of individual objects as a single entity.
- If we want to represent a group of objects as key-value pairs then we should go for Dictionary.

Eg:

```
rollno----name

phone number--address

ipaddress---domain name
```

Key features of Dictionary Data type:

1. Duplicate keys are not allowed but values can be duplicated.
2. Hetrogeneous objects are allowed for both key and values.
3. insertion order is not preserved.
4. Dictionaries are mutable.
5. Dictionaries are dynamic in nature.
6. indexing and slicing concepts are not applicable.

Creation of Set Objects:

1.Creation of dict object with single value:

In [1]:

```
d = {'Karthi':99}
print(type(d))
print(d)
```

```
<class 'dict'>
{'Karthi': 99}
```

2. Creation of dict object with multiple values:

In [2]:

```
d = {'Karthi':99,'saha':100,'Rahul':98}
print(type(d))
print(d)
```

```
<class 'dict'>
{'Karthi': 99, 'saha': 100, 'Rahul': 98}
```

3. Creation of set objects using dict() function:

- We can create dict objects by using dict() function.

In [3]:

```
d = dict()
print(type(d))
```

```
<class 'dict'>
```

In [5]:

```
d=eval(input("Enter Dictionay:"))
print(d)
print(type(d))
```

```
Enter Dictionay:{'a':100,'b':200,'c':300}
{'a': 100, 'b': 200, 'c': 300}
<class 'dict'>
```

4.We can create an empty dictionary by using following approach also:

In [6]:

```
d = {}
print(type(d))
```

```
<class 'dict'>
```

We can add entries into a dictionary as follows:

d[key] = value

In [7]:

```
d[100]="karthi"
d[200]="sahasra"
d[300]="sri"
d['rgm'] = 'Nandyal'
print(d) #{100: 'karthi', 200: 'sahasra', 300: 'sri', 'rgm' : 'Nandyal'}
```

```
{100: 'karthi', 200: 'sahasra', 300: 'sri', 'rgm': 'Nandyal'}
```

b) Demonstrate the following functions/methods which operates on dictionary in Python with suitable examples.

```
i) dict()     ii) len()       iii)clear()     iv) get()

v) pop()      vi)popitem()    vii)keys()    viii)values()

ix)items()     x)copy()        xi)update()
```

Example Program:

Q. Write a Python program to enter name and percentage marks in a dictionary and display information on the screen.

In [8]:

```
rec={}
n=int(input("Enter number of students: "))
i=1
while i <= n:
    name=input("Enter Student Name: ")
    marks=input("Enter % of Marks of Student: ")
    rec[name]=marks
    i=i+1
print("Name of Student","\t","% of Marks")
for x in rec:
    print("\t",x,"\t",rec[x]) # x ===> key rec[x] =====> value
```

```
Enter number of students: 3
Enter Student Name: Karthi
Enter % of Marks of Student: 98
Enter Student Name: Sourav
Enter % of Marks of Student: 97
Enter Student Name: Afridi
Enter % of Marks of Student: 34
Name of Student          % of Marks
         Karthi          98
         Sourav          97
         Afridi          34
```

Important functions of Dictionary:

i. dict():

- This function is used to create a dictionary.

In [10]:

```
d=dict() #It creates empty dictionary
print(d)
d=dict({100:"karthi",200:"saha"})
print(d)
d=dict([(100,"karthi"),(200,"saha"),(300,"sri")])
print(d)
d=dict(((100,"karthi"),(200,"saha"),(300,"sri")))
print(d)
d=dict({(100,"karthi"),(200,"saha"),(300,"sri")})
print(d)
d=dict({[100,"karthi"],[200,"saha"],[300,"sri"]})
print(d)
```

```
{}
{100: 'karthi', 200: 'saha'}
{100: 'karthi', 200: 'saha', 300: 'sri'}
{100: 'karthi', 200: 'saha', 300: 'sri'}
{300: 'sri', 200: 'saha', 100: 'karthi'}

---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<ipython-input-10-b19e5b872e1c> in <module>
      9 d=dict({(100,"karthi"),(200,"saha"),(300,"sri")})
     10 print(d)
---> 11 d=dict({[100,"karthi"],[200,"saha"],[300,"sri"]})
     12 print(d)

TypeError: unhashable type: 'list'
```

Note:

- Compulsory internally we need to take tuple only is acceptable. If you take list it gives the above specified error.

- If the key & values are available in the form of tuple, then all those tuple values can be coverted into dictionary by using 'dict()' function.

ii) len():

- It returns the number of items in the dictionary.

In [11]:

```
d=dict({100:"karthi",200:"saha"}) #It creates dictionary with specified elements
print(d)
print(len(d))
```

```
{100: 'karthi', 200: 'saha'}
2
```

iii) clear():

- This function is used to remove all entries from the dictionary.

In [9]:

```
d={100:"karthi",200:"sahasra",300:"sri"}
print(d)
d.clear()
print(d)
```

```
{100: 'karthi', 200: 'sahasra', 300: 'sri'}
{}
```

iv) get():

- It is used tTo get the value associated with the specified key.

There are two forms of get() method is available in Python.

i. d.get(key):

- If the key is available then returns the corresponding value otherwise returns None.It wont raise any error.

In [12]:

```
d=dict({100:"karthi",200:"saha"}) #It creates dictionary with specified elements
print(d.get(100))
```

```
karthi
```

In [13]:

```
d=dict({100:"karthi",200:"saha"}) #It creates dictionary with specified elements
print(d.get(500))
```

```
None
```

ii. d.get(key,defaultvalue):

- If the key is available then returns the corresponding value otherwise returns default value.

In [14]:

```
d=dict({100:"karthi",200:"saha"}) #It creates dictionary with specified elements
print(d.get(100,'ravan'))
```

```
karthi
```

In [15]:

```
d=dict({100:"karthi",200:"saha"}) #It creates dictionary with specified elements
print(d.get(500,'ravan'))
print(d)
```

```
ravan
{100: 'karthi', 200: 'saha'}
```

Another Example:

In [16]:

```
d={100:"karthi",200:"saha",300:"sri"}
print(d[100]) #karthi
print(d[400]) #KeyError:400
print(d.get(100)) #karthi
print(d.get(400)) #None
print(d.get(100,"Guest")) #karthi
print(d.get(400,"Guest")) #Guest
```

```
karthi

---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
<ipython-input-16-b4151f9f1cde> in <module>
      1 d={100:"karthi",200:"saha",300:"sri"}
      2 print(d[100]) #karthi
----> 3 print(d[400]) #KeyError:400
      4 print(d.get(100)) #karthi
      5 print(d.get(400)) #None

KeyError: 400
```

In [17]:

```
d={100:"karthi",200:"saha",300:"sri"}
print(d[100]) #karthi
#print(d[400]) #KeyError:400
print(d.get(100)) #karthi
print(d.get(400)) #None
print(d.get(100,"Guest")) #karthi
print(d.get(400,"Guest")) #Guest
```

```
karthi
karthi
None
karthi
Guest
```

v) pop():

- It removes the entry associated with the specified key and returns the corresponding value.
- If the specified key is not available then we will get KeyError.

Syntax:

```
d.pop(key)
```

In [18]:

```
d={100:"karthi",200:"saha",300:"sri"}
print(d)
print(d.pop(100))
print(d)
print(d.pop(400))
```

```
{100: 'karthi', 200: 'saha', 300: 'sri'}
karthi
{200: 'saha', 300: 'sri'}

---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
<ipython-input-18-82136391b748> in <module>
      3 print(d.pop(100))
      4 print(d)
----> 5 print(d.pop(400))

KeyError: 400
```

vi) popitem():

- It removes an arbitrary item(key-value) from the dictionaty and returns it.

In [19]:

```
d={100:"karthi",200:"saha",300:"sri"}
print(d)
print(d.popitem())
print(d.popitem())
print(d)
print(d.pop(400)) # KeyError
```

```
{100: 'karthi', 200: 'saha', 300: 'sri'}
(300, 'sri')
(200, 'saha')
{100: 'karthi'}

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-19-ef041cdb3d72> in <module>
      4 print(d.popitem())
      5 print(d)
----> 6 print(d.pop(400)) # KeyError

KeyError: 400
```

If the dictionary is empty then we will get KeyError.

In [20]:

```
d ={}
print(d.popitem()) #KeyError: 'popitem(): dictionary is empty'
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-20-052c88c1625e> in <module>
      1 d ={}
----> 2 print(d.popitem()) #KeyError: 'popitem(): dictionary is empty'

KeyError: 'popitem(): dictionary is empty'
```

Another example:

In [21]:

```
d={100:"karthi",200:"saha",300:"sri"}
print(d)
print(d.popitem())
print(d.popitem())
print(d.popitem())
print(d.popitem())
print(d)
```

```
{100: 'karthi', 200: 'saha', 300: 'sri'}
(300, 'sri')
(200, 'saha')
(100, 'karthi')

---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
<ipython-input-21-45eab152fd1e> in <module>
      4 print(d.popitem())
      5 print(d.popitem())
----> 6 print(d.popitem())
      7 print(d)

KeyError: 'popitem(): dictionary is empty'
```

vii) keys():

- It returns all keys associated with dictionary.

In [22]:

```
d={100:"karthi",200:"saha",300:"sri"}
print(d.keys())
for key in d.keys():
    print(key)
```

```
dict_keys([100, 200, 300])
100
200
300
```

viii) values():

- It returns all values associated with the dictionary.

In [23]:

```
d={100:"karthi",200:"saha",300:"sri"}
print(d.values())
for key in d.values():
    print(key)
```

```
dict_values(['karthi', 'saha', 'sri'])
karthi
saha
sri
```

ix) items():

- It returns list of tuples representing key-value pairs like as shown below.

[(k,v),(k,v),(k,v)]

In [24]:

```
d={100:"karthi",200:"saha",300:"sri"}
list = d.items()
print(list)
```

```
dict_items([(100, 'karthi'), (200, 'saha'), (300, 'sri')])
```

In [26]:

```
d={100:"karthi",200:"saha",300:"sri"}
for k,v in d.items():
    print(k,"-->",v)
```

```
100 --> karthi
200 --> saha
300 --> sri
```

x) copy():

- This method is used to create exactly duplicate dictionary(cloned copy).

In [27]:

```
d={100:"karthi",200:"saha",300:"sri"}
d1=d.copy()
print(d1)
print(d)
```

```
{100: 'karthi', 200: 'saha', 300: 'sri'}
{100: 'karthi', 200: 'saha', 300: 'sri'}
```

xi) update():

Syntax:

```
d.update(x)
```

- All items present in the dictionary 'x' will be added to dictionary 'd'.

In [28]:

```
d={100:"karthi",200:"saha",300:"sri"}
d1 ={'a':'apple', 'b':'banana'}
d.update(d1)
print(d)
```

```
{100: 'karthi', 200: 'saha', 300: 'sri', 'a': 'apple', 'b': 'banana'}
```

In [29]:

```
d={100:"karthi",200:"saha",300:"sri"}
d1 ={'a':'apple', 'b':'banana'}
d2 = {777:'A', 888:'B'}
d.update(d1,d2) # For update method. you need to pass single argument only.
print(d)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-29-b0832a652cd0> in <module>
      2 d1 ={'a':'apple', 'b':'banana'}
      3 d2 = {777:'A', 888:'B'}
----> 4 d.update(d1,d2) # For update method. you need to pass single argument only.
      5 print(d)

TypeError: update expected at most 1 argument, got 2
```

In [30]:

```
d={100:"karthi",200:"saha",300:"sri"}
d1 ={'a':'apple', 'b':'banana'}
d2 = {777:'A', 888:'B'}
d.update([(777,'A')]) # For uipdate method. you can pass list of tuple as an argument.
print(d)
```

```
{100: 'karthi', 200: 'saha', 300: 'sri', 777: 'A'}
```

In [31]:

```
d={100:"karthi",200:"saha",300:"sri"}
d1 ={'a':'apple', 'b':'banana'}
d2 = {777:'A', 888:'B'}
d.update([(777,'A'),(888,'B'),(999,'C')]) # you can add any no.of list of tuple elements.
print(d)
```

```
{100: 'karthi', 200: 'saha', 300: 'sri', 777: 'A', 888: 'B', 999: 'C'}
```

Few More Example Programs on Dictioary data type:

Q1. Write a Python program to take dictionary from the keyboard and print the sum of values.

In [32]:

```
d=eval(input("Enter dictionary:"))
s=sum(d.values())
print("Sum= ",s)
```

```
Enter dictionary:{'A':100,'B':200,'c':300}
Sum=  600
```

In [33]:

```
d=eval(input("Enter dictionary:"))
s=sum(d.values())
print("Sum= ",s)
```

```
Enter dictionary:'A':100,'B':200,'c':300

Traceback (most recent call last):

  File "C:\Users\HP\anaconda3\lib\site-packages\IPython\core\interactivesh
ell.py", line 3343, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)

  File "<ipython-input-33-51a5d22abba9>", line 1, in <module>
    d=eval(input("Enter dictionary:"))

  File "<string>", line 1
    'A':100,'B':200,'c':300
       ^
SyntaxError: invalid syntax
```

Sum() function:

In [34]:

```
l = [10,20,30,40]
s = sum(l)              # sum() function works on list also
print('Sum is : ',s)
```

```
Sum is :  100
```

In [35]:

```
l = (10,20,30,40)
s = sum(l)             # sum() function works on tuple also
print('Sum is : ',s)
l
```

```
Sum is :  100
```

Out[35]:

```
(10, 20, 30, 40)
```

In [36]:

```
l = {10,20,30,40}
s = sum(l)            # sum() function works on set also
print('Sum is : ',s)
```

```
Sum is :  100
```

Note: sum() function can work on any sequence.

Q2. Write a Python program to find number of occurrences of each letter present in the given string.

In [38]:

```
word=input("Enter any word: ")
d={}
for x in word:
    d[x]=d.get(x,0)+1
for k,v in d.items():
    print(k,"occurred ",v," times")
```

```
Enter any word: mississippi
m occurred  1  times
i occurred  4  times
s occurred  4  times
p occurred  2  times
```

In [39]:

```
word=input("Enter any word: ")
d={}
for x in word:
    d[x]=d.get(x,0)+1
for k,v in sorted(d.items()): # To sort all the items of the dictionary in alphabetical
order
    print(k,"occurred ",v," times")
```

```
Enter any word: mississippi
i occurred  4  times
m occurred  1  times
p occurred  2  times
s occurred  4  times
```

Q3. Write a Python program to find number of occurrences of each vowel present in the given string.

In [40]:

```
word=input("Enter any word: ")
vowels={'a','e','i','o','u'}
d={}
for x in word:
    if x in vowels:
        d[x]=d.get(x,0)+1
for k,v in sorted(d.items()):
    print(k,"occurred ",v," times")
```

```
Enter any word: doganimaldoganimal
a occurred  4  times
i occurred  2  times
o occurred  2  times
```

Q4. Write a program to accept student name and marks from the keyboard and creates a dictionary. Also display student marks by taking student name as input.

In [41]:

```
n=int(input("Enter the number of students: "))
d={}
for i in range(n):
    name=input("Enter Student Name: ")
    marks=input("Enter Student Marks: ")
    d[name]=marks                 # assigninng values to the keys of the dictionary 'd'
while True:
    name=input("Enter Student Name to get Marks: ")
    marks=d.get(name,-1)
    if marks== -1:
        print("Student Not Found")
    else:
        print("The Marks of",name,"are",marks)
    option=input("Do you want to find another student marks[Yes|No]")
    if option=="No":
        break
print("Thanks for using our application")
```

```
Enter the number of students: 5
Enter Student Name: Karthi
Enter Student Marks: 87
Enter Student Name: Sahasra
Enter Student Marks: 88
Enter Student Name: Sourav
Enter Student Marks: 77
Enter Student Name: Rahul
Enter Student Marks: 65
Enter Student Name: Virat
Enter Student Marks: 87
Enter Student Name to get Marks: Karthi
The Marks of Karthi are 87
Do you want to find another student marks[Yes|No]y
Enter Student Name to get Marks: karthi
Student Not Found
Do you want to find another student marks[Yes|No]y
Enter Student Name to get Marks: Virat
The Marks of Virat are 87
Do you want to find another student marks[Yes|No]y
Enter Student Name to get Marks: Robin
Student Not Found
Do you want to find another student marks[Yes|No]No
Thanks for using our application
```

Good Luck