1. Python Output

Theory:- - Three types:- markup(HTML,XML,YAML), Scripting(JS,Perl),Programming language(python,java,c++ etc)

- In Dynamic languages typecheck happens at run time whereas in Static languages it occurs at compile time.
- a(reference variable in stack)=10(object in heap)
- if a new object is created with the same ref variable name, the reference variable starts pointing to the new object and the previous one is cleared by the garbage collection.
- In strictly typed languages like python typecheck is strictly enforced whereas in loosely typed its not.

-Data types are of two types- mutable(list,set,dictionaries) and immutable(tuple).

- In python 0 means false, any other number means true(even -ves or decimals. Empty lists,tuples etc are also considered false.
- In python blocks are identified using indentations.
- ''' xyz ''' should not be used as comments

```
# Python is a case sensitive language
print('Hello World')
```

```
Hello World
```

```
print('Vanshika')
```

```
Vanshika
```

```
print(vanshika)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-1916897944.py in <cell line: 0>()
----> 1 print(vanshika)

NameError: name 'vanshika' is not defined
```

```
print(7)
```

```
print(7.7)
```

```
print(True)

print('Hello',1,4.5,True)

print('Hello',1,4.5,True,sep='/')

print('hello')
print('world')
```

```
hello
world
```

```
print('hello',end='-')
print('world')
```

```
hello-world
```

2. Data Types

```
# Integer
print(8)
# 1*10^308
print(1e309)
```

```
8
inf
```

```
# Decimal/Float
print(8.55)
print(1.7e309)
```

```
8.55
inf
```

```
# Boolean
print(True)
print(False)
```

```
True
False
```

```
# Text/String
print('Hello World')
```

```
Hello World
```

```
# complex
print(5+6j)
```

```
(5+6j)
```

```
# List-> C-> Array
print([1,2,3,4,5])
```

```
[1, 2, 3, 4, 5]
```

```
# Tuple
print((1,2,3,4,5))
```

```
(1, 2, 3, 4, 5)
```

```
# Sets
print({1,2,3,4,5})
```

```
{1, 2, 3, 4, 5}
```

```
# Dictionary
print({'name':'Nitish','gender':'Male','weight':70})
```

```
{'name': 'Nitish', 'gender': 'Male', 'weight': 70}
```

```
# type
type([1,2,3])
```

```
list
```

3. Variables

```
# Static Vs Dynamic Typing
# Static Vs Dynamic Binding
# stylish declaration techniques

# C/C++
name = 'vanshika'
print(name)

a = 5
b = 6

print(a + b)
```

```
vanshika
11
```

```
# Dynamic Typing
a = 5
# Static Typing
int a = 5
```

```
  File "/tmp/ipython-input-1639363930.py", line 4
    int a = 5
        ^
```

```
SyntaxError: invalid syntax
```

```
# Dynamic Binding
a = 5
print(a)
a = 'vanshika'
print(a)
#shows dynamic allocation - previous object which was pointed by a is cleared by garbage collection
# Static Binding
int a = 5
```

```
a = 1
b = 2
c = 3
print(a,b,c)
```

```
1 2 3
```

```
a,b,c = 1,2,3
print(a,b,c)
```

```
1 2 3
```

```
a=b=c= 5
print(a,b,c)
```

```
5 5 5
```

```
a=100000    #points to diff objects
b=100000
a is b
```

```
False
```

```
a=10        #points to same object  -6 to 256 is reserved in python for better memory management
b=10
a is b
```

```
True
```

```
a=2**33    #size of int is same as size of RAM of computer hence python is slow
type(a)
```

```
int
```

Comments

```
# this is a comment
# second line
a = 4
b = 6 # like this
# second comment
print(a+b)
```

```
10
```

4. Keywords & Identifiers

```
# Keywords

# Identifiers
# You can't start with a digit
name1 = 'Vanshika'
print(name1)
# You can use special chars -> _
_ = 'vanshika'
print(_)
# identifiers can not be keyword
```

```
Vanshika
vanshika
```

5. User Input

```
# Static Vs Dynamic
input('Enter Email')
```

```
Enter Emailmvanshika23@gmail.com
```

```
{"type":"string"}
```

```
# take input from users and store them in a variable
fnum = int(input('enter first number'))
snum = int(input('enter second number'))
#print(type(fnum),type(snum))
# add the 2 variables
result = fnum + snum
# print the result
print(result)
print(type(fnum))
```

```
enter first number2
enter second number3
5
<class 'int'>
```

```
a=input()
b=input()
```

```
23
45
```

```
print(a+b)  #input by default is string type
```

```
2345
```

```
type(a),type(b)
```

```
(str, str)
```

```
ans=int(a)+int(b)
print(ans)
```

```
68
```

```
ans2=int(a+b) #wrong
print(ans2)
```

```
2345
```

6. Type Conversion

```
# Implicit Vs Explicit
print(5+5.6)
print(type(5),type(5.6))
```

```
print(4 + '4')
```

```
10.6
<class 'int'> <class 'float'>
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
/tmp/ipython-input-3295153562.py in <cell line: 0>()
      3 print(type(5),type(5.6))
      4
----> 5 print(4 + '4')

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

```
# Explicit
# str -> int
#int(4+5j)

# int to str
str(5)

# float
float(4)
```

```
4.0
```

7. Literals

```
a = 0b1010 #Binary Literals
b = 100 #Decimal Literal
c = 0o310 #Octal Literal
d = 0x12c #Hexadecimal Literal

#Float Literal
float_1 = 10.5
float_2 = 1.5e2 # 1.5 * 10^2
float_3 = 1.5e-3 # 1.5 * 10^-3

#Complex Literal
x = 3.14j

print(a, b, c, d)
print(float_1, float_2,float_3)
print(x, x.imag, x.real)
```

```
10 100 200 300
10.5 150.0 0.0015
3.14j 3.14 0.0
```

```
# binary
x = 3.14j
print(x.imag)
```

```
3.14
```

```
string = 'This is Python'
strings = "This is Python"
char = "C"
multiline_str = """This is a multiline string with more than one line
code."""
unicode = u"\U0001f600\U0001F606\U0001F923"
raw_str = r"raw \n string"
```

```
print(string)
print(strings)
print(char)
print(multiline_str)
print(unicode)
print(raw_str)
```

```
This is Python
This is Python
C
This is a multiline string with more than one line code.
😀😆
raw \n string
```

```
a = True + 4
b = False + 10

print("a:", a)
print("b:", b)
```

```
a: 5
b: 10
```

```
k = None
a = 5
b = 6
print('Program exe')
```

```
Program exe
```

8. Operators

```
# Arithmetic
# Relational
# Logical
# Bitwise
# Assignment
# Membership

# Arithmetric Operators
print(5+6)

print(5-6)

print(5*6)

print(5/2)

print(5//2)
```

```
print(5%2)

print(5**2)
```

```
11
-1
30
2.5
2
1
25
```

```
# Relational Operators
print(4>5)

print(4<5)

print(4>=4)

print(4<=4)

print(4==4)

print(4!=4)
```

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

```
# Logical Operators
print(1 and 0)

print(1 or 0)

print(not 1)
```

```
0
1
False
```

```
# Bitwise Operators

# bitwise and
print(2 & 3)

# bitwise or
print(2 | 3)
```

```
# bitwise xor
print(2 ^ 3)

print(~3)

print(4 >> 2)

print(5 << 2)
```

```
2
3
1
-4
1
20
```

```
# Assignment Operators

# =
# a = 2

a = 2

# a = a % 2
a %= 2

# a++ ++a

print(a)
```

```
0
```

```
# Membership Operators

# in/not in

print('D' not in 'Delhi')

print(1 in [2,3,4,5,6])
```

```
False
False
```

```
# Program - Find the sum of a 3 digit number entered by the user

number = int(input('Enter a 3 digit number'))

# 345%10 -> 5
a = number%10

number = number//10
```

```
# 34%10 -> 4
b = number % 10

number = number//10
# 3 % 10 -> 3
c = number % 10

print(a + b + c)
```

```
Enter a 3 digit number453
12
```

9. If-Else

```
# login program and indentation
# email -> mvanshika23@gmail.com.com
# password -> 1234

email = input('enter email')
password = input('enter password')

if email == 'mvanshika23@gmail.com' and password == '1234':
  print('Welcome')
elif email == 'mvanshika23@gmail.com' and password != '1234':
  # tell the user
  print('Incorrect password')
  password = input('enter password again')
  if password == '1234':
    print('Welcome,finally!')
  else:
    print('beta tumse na ho paayega!')
else:
  print('Not correct')
```

```
enter emailmvanshika23@gmail.com
enter password123
Incorrect password
enter password again1234
Welcome,finally!
```

```
# if-else examples
# 1. Find the min of 3 given numbers
# 2. Menu Driven Program

# min of 3 number

a = int(input('first num'))
b = int(input('second num'))
c = int(input('third num'))
```

```
if a<b and a<c:
  print('smallest is',a)
elif b<c:
  print('smallest is',b)
else:
  print('smallest is',c)
```

```
first num3
second num6
third num4
smallest is 3
```

```
# menu driven calculator
menu = input("""
Hi! how can I help you.
1. Enter 1 for pin change
2. Enter 2 for balance check
3. Enter 3 for withdrawl
4. Enter 4 for exit
""")

if menu == '1':
  print('pin change')
elif menu == '2':
  print('balance')
else:
  print('exit')
```

```
Hi! how can I help you.
1. Enter 1 for pin change
2. Enter 2 for balance check
3. Enter 3 for withdrawl
4. Enter 4 for exit
2
balance
```

Modules in Python math keywords random datetime

```
# math
import math

math.sqrt(196)
```

```
14.0
```

```
# keyword
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await',
'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except',
'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is',
'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try',
'while', 'with', 'yield']
```

```
# random
import random
print(random.randint(1,100))
```

```
51
```

```
# datetime
import datetime
print(datetime.datetime.now())
```

```
2025-08-08 07:03:56.378220
```

```
#help('modules')
```

Loops in Python

Need for loops

While Loop

For Loop

```
number = int(input('enter the number'))

i = 1

while i<11:
  print(number,'*',i,'=',number * i)
  i += 1
```

```
enter the number10
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
```

```
# while loop with else

x = 1

while x < 3:
  print(x)
  x += 1

else:
  print('limit crossed')
```

```
1
2
limit crossed
```

```
# For loop demo

for i in {1,2,3,4,5}:
  print(i)
```

```
1
2
3
4
5
```

Program - The current population of a town is 10000. The population of the town is increasing at the rate of 10% per year. You have to write a program to find out the population at the end of each of the last 10 years.

```
curr_pop = 10000

for i in range(10,0,-1):
  print(i,curr_pop)
  curr_pop = curr_pop - 0.1*curr_pop
```

```
10 10000
9 9000.0
8 8100.0
7 7290.0
6 6561.0
5 5904.9
4 5314.41
3 4782.969
2 4304.6721
1 3874.20489
```

Sequence sum

1/1! + 2/2! + 3/3! + ...

```
# Code here

n = int(input('enter n'))

result = 0
fact = 1

for i in range(1,n+1):
  fact = fact * i
  result = result + i/fact

print(result)
```

```
enter n5
2.708333333333333
```

Nested Loops

```
# Examples -> unique pairs

for i in range(1,5):
  for j in range(1,5):
    print(i,j)
```

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

Pattern 1

**

```
# code here

rows = int(input('enter number of rows'))

for i in range(1,rows+1):
  for j in range(1,i+1):
    print('*',end='')
  print()
```

```
enter number of rows5
*
**
***
****
*****
```

Loop Control Statement

Break

Continue

Pass

```
for i in range(1,10):
  if i == 5:
    break
  print(i)
```

```
1
2
3
4
```

```
lower = int(input('enter lower range'))
upper = int(input('enter upper range'))

for i in range(lower,upper+1):
  for j in range(2,i):
    if i%j == 0:
      break
  else:
    print(i)
```

```
enter lower range10
enter upper range100
11
```

```
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
```

```
# Continue
for i in range(1,10):
  if i == 5:
    continue
  print(i)
```

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

```
for i in range(1,10):
  pass
```

Strings are sequence of Characters

In Python specifically, strings are a sequence of Unicode Characters

- Creating Strings
- Accessing Strings
- Adding Chars to Strings
- Editing Strings
- Deleting Strings
- Operations on Strings
- String Functions

Creating Stings

```
s = 'hello'
s = "hello"
# multiline strings
s = '''hello'''
s = """hello"""
s = str('hello')
print(s)
```

```
hello
```

```
"it's raining outside"
```

Accessing Substrings from a String

```
# Positive Indexing
s = 'hello world'
print(s[4])
```

```
o
```

```
# Negative Indexing
s = 'hello world'
print(s[-3])
```

```
r
```

```
# Slicing s[start:stop:step]
s = 'hello world'
print(s[6:0:-2])
```

```
wol
```

```
print(s[::-1])
```

```
dlrow olleh
```

```
s = 'hello world'
print(s[-1:-6:-1])
```

```
dlrow
```

Editing and Deleting in Strings

```
s = 'hello world'
s[0] = 'H'

# Python strings are immutable
```

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
/tmp/ipython-input-2237226474.py in <cell line: 0>()
      1 s = 'hello world'
----> 2 s[0] = 'H'
      3
      4 # Python strings are immutable

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

```
s = 'hello world'
del s
print(s)
```

```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
/tmp/ipython-input-2968721042.py in <cell line: 0>()
      1 s = 'hello world'
      2 del s
----> 3 print(s)

NameError: name 's' is not defined
```

```
s = 'hello world'
del s[-1:-5:2]
print(s)
```

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
/tmp/ipython-input-266792670.py in <cell line: 0>()
      1 s = 'hello world'
----> 2 del s[-1:-5:2]
      3 print(s)

TypeError: 'str' object does not support item deletion
```

Operations on Strings

- Arithmetic Operations
- Relational Operations
- Logical Operations
- Loops on Strings
- Membership Operations

```
print('delhi' + ' ' + 'mumbai')
```

```
delhi mumbai
```

```
print('delhi'*5)
```

```
delhidelhidelhidelhidelhi
```

```
print("*"*50)
```

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

```
'delhi' != 'delhi'
```

```
False
```

```
'mumbai' > 'pune'
# lexiographically
```

```
False
```

```
'Pune' > 'pune'
```

```
False
```

```
'hello' and 'world'
```

```
{"type":"string"}
```

```
'hello' or 'world'
```

```
{"type":"string"}
```

```
'' and 'world'
```

```
{"type":"string"}
```

```
'' or 'world'
```

```
{"type":"string"}
```

```
'hello' or 'world'
```

```
{"type":"string"}
```

```
not 'hello'
```

```
False
```

```
for i in 'hello':
  print(i)
```

```
h
e
l
```

```
l
o
```

```
for i in 'delhi':
  print('pune')
```

```
pune
pune
pune
pune
pune
```

```
'D' in 'delhi'
```

```
False
```

Common Functions

- len
- max
- min
- sorted
- Capitalize
- Title
- Upper
- Lower
- Swapcase

```
len('hello world')
```

```
11
```

```
max('hello world')
```

```
{"type":"string"}
```

```
min('hello world')
```

```
{"type":"string"}
```

```
sorted('hello world',reverse=True)
```

```
['w', 'r', 'o', 'o', 'l', 'l', 'l', 'h', 'e', 'd', ' ']
```

```
s = 'hello world'
print(s.capitalize())
print(s)
```

```
Hello world
hello world
```

```
s.title()
```

```
{"type":"string"}
```

```
s.upper()
```

```
{"type":"string"}
```

```
'Hello World'.lower()
```

```
{"type":"string"}
```

```
'HeLlO WorLD'.swapcase()
```

```
{"type":"string"}
```

Count/Find/Index/Starts With/Ends With

```
'my name is vanshika'.count('i')
```

```
2
```

```
'my name is vanshika'.find('x')
```

```
-1
```

```
'my name is vanshika'.index('v')
```

```
11
```

```
'my name is vanshika'.endswith('ka')
```

```
True
```

```
'my name is vanshika'.startswith('My')
```

```
False
```

Format

```
name = 'vanshika'
gender = 'female'

'Hi my name is {1} and I am a {0}'.format(gender,name)
```

```
{"type":"string"}
```

```
a=30
b=45
c=34
d=23
```

```
e=23
f=34
g=23
h=23
print("The eight numbers are ",a," and ",b,"and",c," and ",d) #boring
```

```
The eight numbers are  30  and  45 and 34  and  23
```

```
print('''The numbers are {} and {} and {}
and {}   and {}
                        and{}'''.format(a,b,c,d,e,f)) #bad
```

```
The numbers are 30 and 45 and 34
and 23   and 23
                        and34
```

```
print(f'''The numbers are {a} and {b}
and {c} and {d}
and {e} and {f} and {g}''' ) #better
```

```
The numbers are 30 and 45
and 34 and 23
and 23 and 34 and 23
```

```
my_str='Vanshika123'
print(my_str.isalnum()) #check if all char are numbers and alphabets
print(my_str.isalpha()) #check if all char in the string are
alphabetic
print(my_str.isdigit()) #test if string has only digits
print(my_str.istitle()) #test if string contains title words
print(my_str.isupper()) #test if string contains upper case
print(my_str.islower()) #test if string contains lower case
print(my_str.isspace()) #test if string contains spaces
```

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

```
'hi my name is vanshika'.split()
```

```
['hi', 'my', 'name', 'is', 'vanshika']
```

```
" ".join(['hi', 'my', 'name', 'is', 'vanshika'])
```

```
{"type":"string"}
```

```
'hi my name is vanshikagewrhgh'.replace('vanshikagewrhgh','vanshika')
```

```
{"type":"string"}

'         example                              '.strip()

{"type":"string"}
```

Programs

```
# Find the length of a given string without using the len() function

s = input('enter the string')

for i in s:
  counter += 1

print('length of string is',counter)

enter the stringVanshika
length of string is 8

# Extract username from a given email.
# Eg if the email is mvanshika23@@gmail.com
# then the username should be mvanshika23

s = input('enter the email')

pos = s.index('@')
print(s[0:pos])

enter the emailmvanshika23@gmail.com
mvanshika23

# Count the frequency of a particular character in a provided string.
# Eg 'hello how are you' is the string, the frequency of h in this
string is 2.

s = input('enter the string')
term = input('what would like to search for')

counter = 0
for i in s:
  if i == term:
    counter += 1

print('frequency',counter)

enter the stringVanshika
what would like to search fora
frequency 2
```

```
# Write a program which can remove a particular character from a
string.
s = input('enter the string')
term = input('what would you like to remove')

result = ''

for i in s:
  if i != term:
    result = result + i

print(result)
```

```
enter the stringVanshikaM
what would you like to removeM
Vanshika
```

```
# Write a program that can check whether a given string is palindrome
or not.

s = input('enter the string')
flag = True
for i in range(0,len(s)//2):
  if s[i] != s[len(s) - i -1]:
    flag = False
    print('Not a Palindrome')
    break

if flag:
  print('Palindrome')
```

```
enter the stringmalyalam
Not a Palindrome
```

```
# Write a program to count the number of words in a string without
split()

s = input('enter the string')
L = []
temp = ''
for i in s:

  if i != ' ':
    temp = temp + i
  else:
    L.append(temp)
    temp = ''

L.append(temp)
print(L)
```

```
enter the stringmy name is vanshika
['my', 'name', 'is', 'vanshika']
```

```
# Write a python program to convert a string to title case without
using the title()
s = input('enter the string')

L = []
for i in s.split():
  L.append(i[0].upper() + i[1:].lower())

print(" ".join(L))
```

```
enter the stringVanshika Mishra
Vanshika Mishra
```

```
# Write a program that can convert an integer to string.

number = int(input('enter the number'))

digits = '0123456789'
result = ''
while number != 0:
  result = digits[number % 10] + result
  number = number//10

print(result)
print(type(result))
```

```
enter the number87967899
87967899
<class 'str'>
```

Lists

- What are Lists?
- Lists Vs Arrays
- Characterstics of a List
- How to create a list
- Access items from a List
- Editing items in a List
- Deleting items from a List
- Operations on Lists
- Functions on Lists

What are Lists

List is a data type where you can store multiple items under 1 name. More technically, lists act like dynamic arrays which means you can add more items on the fly.

Array Vs Lists

- Fixed Vs Dynamic Size
- Convenience -> Hetrogeneous
- Speed of Execution
- Memory

```
L = [1,2,3]

print(id(L))
print(id(L[0]))
print(id(L[1]))
print(id(L[2]))
print(id(1))
print(id(2))
print(id(3))
```

```
134661189784000
10757736
10757768
10757800
10757736
10757768
10757800
```

Characterstics of a List

- Ordered
- Changeble/Mutable
- Hetrogeneous
- Can have duplicates
- are dynamic
- can be nested
- items can be accessed
- can contain any kind of objects in python

```
L = [1,2,3,1]
L1 = [3,2,1]

L == L1
```

```
False
```

Creating a List

```
# Empty
print([])
# 1D -> Homo
print([1,2,3,4,5])
# 2D
print([1,2,3,[4,5]])
# 3D
print([[[1,2],[3,4]],[[5,6],[7,8]]])
# Hetrogenous
print([1,True,5.6,5+6j,'Hello'])
# Using Type conversion
print(list('hello'))
```

```
[]
[1, 2, 3, 4, 5]
[1, 2, 3, [4, 5]]
[[[1, 2], [3, 4]], [[5, 6], [7, 8]]]
[1, True, 5.6, (5+6j), 'Hello']
['h', 'e', 'l', 'l', 'o']
```

```
a=[i for i in range(1,6)]    #list of first 5 natural no.s
a
```

```
[1, 2, 3, 4, 5]
```

```
t=[2*i for i in range(1,11) if i%2==0]   #table of 2
t
```

```
[4, 8, 12, 16, 20]
```

```
b=[[i*n for i in range(1,11)] for n in range(1,11)]
b
```

```
[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
 [2, 4, 6, 8, 10, 12, 14, 16, 18, 20],
 [3, 6, 9, 12, 15, 18, 21, 24, 27, 30],
 [4, 8, 12, 16, 20, 24, 28, 32, 36, 40],
 [5, 10, 15, 20, 25, 30, 35, 40, 45, 50],
 [6, 12, 18, 24, 30, 36, 42, 48, 54, 60],
 [7, 14, 21, 28, 35, 42, 49, 56, 63, 70],
 [8, 16, 24, 32, 40, 48, 56, 64, 72, 80],
 [9, 18, 27, 36, 45, 54, 63, 72, 81, 90],
 [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]]
```

Accessing Items from a List

```
# Indexing
L = [[[1,2],[3,4]],[[5,6],[7,8]]]
#positive
```

```
#print(L[0][0][1])

# Slicing
L = [1,2,3,4,5,6]

print(L[::-1])
```

```
[6, 5, 4, 3, 2, 1]
```

Adding Items to a List

```
# append
L = [1,2,3,4,5]
L.append(True)
print(L)
```

```
[1, 2, 3, 4, 5, True]
```

```
# extend
L = [1,2,3,4,5]
L.extend([6,7,8])
print(L)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

```
#append
L = [1,2,3,4,5]
L.append([6,7,8])
print(L)
```

```
[1, 2, 3, 4, 5, [6, 7, 8]]
```

```
L = [1,2,3,4,5]
L.extend('delhi')
print(L)
```

```
[1, 2, 3, 4, 5, 'd', 'e', 'l', 'h', 'i']
```

```
# insert
L = [1,2,3,4,5]

L.insert(1,100)
print(L)
```

```
[1, 100, 2, 3, 4, 5]
```

Editing items in a List

```
L = [1,2,3,4,5]
```

```
# editing with indexing
L[-1] = 500

# editing with slicing
L[1:4] = [200,300,400]

print(L)
```

```
[1, 200, 300, 400, 500]
```

Deleting items from a List

```
# del
L = [1,2,3,4,5]

# indexing
del L[-1]

# slicing
del L[1:3]
print(L)
```

```
[1, 4]
```

```
# remove

L = [1,2,3,4,5]

L.remove(5)

print(L)
```

```
[1, 2, 3, 4]
```

```
# pop
L = [1,2,3,4,5]

L.pop()

print(L)
```

```
[1, 2, 3, 4]
```

```
# clear
L = [1,2,3,4,5]

L.clear()

print(L)
```

```
[]
```

Operations on Lists

- Arithmetic
- Membership
- Loop

```
# Arithmetic (+ ,*)

L1 = [1,2,3,4]
L2 = [5,6,7,8]

# Concatenation/Merge
print(L1 + L2)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

```
print(L1*3)
```

```
[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
```

```
L1 = [1,2,3,4,5]
L2 = [1,2,3,4,[5,6]]

print(5 not in L1)
print([5,6] in L2)
```

```
False
True
```

```
# Loops
L1 = [1,2,3,4,5]
L2 = [1,2,3,4,[5,6]]
L3 = [[[1,2],[3,4]],[[5,6],[7,8]]]

for i in L3:
  print(i)
```

```
[[1, 2], [3, 4]]
[[5, 6], [7, 8]]
```

List Functions

```
# len/min/max/sorted
L = [2,1,5,7,0]

print(len(L))
print(min(L))
print(max(L))
print(sorted(L,reverse=True))
```

```
5
0
```

```
7
[7, 5, 2, 1, 0]

# count
L = [1,2,1,3,4,1,5]
L.count(5)

1

# reverse
L = [2,1,5,7,0]
# permanently reverses the list
L.reverse()
print(L)

[0, 7, 5, 1, 2]

# index
L = [1,2,1,3,4,1,5]
L.index(1)

0

# sort (vs sorted)
L = [2,1,5,7,0]
print(L)
print(sorted(L))
print(L)
L.sort()
print(L)

[2, 1, 5, 7, 0]
[0, 1, 2, 5, 7]
[2, 1, 5, 7, 0]
[0, 1, 2, 5, 7]

# copy -> shallow
L = [2,1,5,7,0]
print(L)
print(id(L))
L1 = L.copy()
print(L1)
print(id(L1))

[2, 1, 5, 7, 0]
140104858076992
[2, 1, 5, 7, 0]
140104858075264
```

List Comprehension

List Comprehension provides a concise way of creating lists.

```
# Add 1 to 10 numbers to a list
L = []

for i in range(1,11):
  L.append(i)

print(L)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
# scalar multiplication on a vector
v = [2,3,4]
s = -3
# [-6,-9,-12]

[s*i for i in v]
```

```
[-6, -9, -12]
```

```
# Print all numbers divisible by 5 in the range of 1 to 50

[i for i in range(1,51) if i%5 == 0]
```

```
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
```

```
# find languages which start with letter p
languages = ['java','python','php','c','javascript']

[language for language in languages if language.startswith('p')]
```

```
['python', 'php']
```

```
# cartesian products -> List comprehension on 2 lists together
L1 = [1,2,3,4]
L2 = [5,6,7,8]

[i*j for i in L1 for j in L2]
```

```
[5, 6, 7, 8, 10, 12, 14, 16, 15, 18, 21, 24, 20, 24, 28, 32]
```

```
# itemwise
L = [1,2,3,4]

for i in L:
  print(i)
```

```
1
2
```

```
3
4
```

```
# indexwise
L = [1,2,3,4]

for i in range(0,len(L)):
  print(L[i])
```

```
1
2
3
4
```

Zip

The zip() function returns a zip object, which is an iterator of tuples where the first item in each passed iterator is paired together, and then the second item in each passed iterator are paired together.

If the passed iterators have different lengths, the iterator with the least items decides the length of the new iterator.

```
# Write a program to add items of 2 lists indexwise

L1 = [1,2,3,4]
L2 = [-1,-2,-3,-4]

list(zip(L1,L2))

[i+j for i,j in zip(L1,L2)]
```

```
[0, 0, 0, 0]
```

```
L = [1,2,print,type,input]

print(L)
```

```
[1, 2, <built-in function print>, <class 'type'>, <bound method
Kernel.raw_input of <google.colab._kernel.Kernel object at
0x7f6cb904c6d0>>]
```

```
a = [1,2,3]
b = a.copy()

print(a)
print(b)

a.append(4)
print(a)
print(b)
```

```
# lists are mutable

[1, 2, 3]
[1, 2, 3]
[1, 2, 3, 4]
[1, 2, 3]
```

Tuples

A tuple in Python is similar to a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list.

In short, a tuple is an immutable list. A tuple can not be changed in any way once it is created.

Characterstics

- Ordered
- Unchangeble
- Allows duplicate

Plan of attack

- Creating a Tuple
- Accessing items
- Editing items
- Adding items
- Deleting items
- Operations on Tuples
- Tuple Functions

Creating Tuples

```
# empty
t1 = ()
print(t1)
# create a tuple with a single item
t2 = ('hello',)
print(t2)
print(type(t2))
# homo
t3 = (1,2,3,4)
print(t3)
# hetro
t4 = (1,2.5,True,[1,2,3])
print(t4)
# tuple
t5 = (1,2,3,(4,5))
print(t5)
# using type conversion
t6 = tuple('hello')
print(t6)
```

```
()
('hello',)
<class 'tuple'>
```

```
(1, 2, 3, 4)
(1, 2.5, True, [1, 2, 3])
(1, 2, 3, (4, 5))
('h', 'e', 'l', 'l', 'o')
```

Accessing Items

- Indexing
- Slicing

```
print(t3)
print(t3[0])
print(t3[-1])
```

```
(1, 2, 3, 4)
1
4
```

```
t5[-1][0]
```

```
4
```

Editing items

```
print(t3)
t3[0] = 100
# immutable just like strings
```

```
(1, 2, 3, 4)
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-30-49d9e1416ccf> in <module>
      1 print(t3)
----> 2 t3[0] = 100

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

Adding items

```
print(t3)
# not possible
```

```
(1, 2, 3, 4)
```

Deleting items

```
print(t3)
del t3
print(t3)
```

```
(1, 2, 3, 4)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-33-0a67b29ad777> in <module>
      1 print(t3)
      2 del t3
----> 3 print(t3)

NameError: name 't3' is not defined
```

```
t = (1,2,3,4,5)

t[-1:-4:-1]
```

```
(5, 4, 3)
```

```
print(t5)
del t5[-1]
```

```
(1, 2, 3, (4, 5))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-35-2b39d140e8ae> in <module>
      1 print(t5)
----> 2 del t5[-1]

TypeError: 'tuple' object doesn't support item deletion
```

Operations on Tuples

```
# + and *
t1 = (1,2,3,4)
t2 = (5,6,7,8)

print(t1 + t2)

print(t1*3)
# membership
1 in t1
# iteration
```

```
for i in t1:
  print(i)
```

```
(1, 2, 3, 4, 5, 6, 7, 8)
(1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4)
1
2
3
4
```

Tuple Functions

```
# len/sum/min/max/sorted
t = (1,2,3,4)
len(t)
```

```
sum(t)
```

```
min(t)
```

```
max(t)
```

```
sorted(t,reverse=True)
```

```
[4, 3, 2, 1]
```

```
# count

t = (1,2,3,4,5)
```

```
t.count(50)
```

```
0
```

```
# index
t.index(50)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-51-cae2b6ba49a8> in <module>
      1 # index
----> 2 t.index(50)

ValueError: tuple.index(x): x not in tuple
```

Difference between Lists and Tuples

- Syntax
- Mutability
- Speed

- Memory
- Built in functionality
- Error prone
- Usability

```
import time

L = list(range(100000000))
T = tuple(range(100000000))

start = time.time()
for i in L:
  i*5
print('List time',time.time()-start)

start = time.time()
for i in T:
  i*5
print('Tuple time',time.time()-start)
```

```
List time 9.853569507598877
Tuple time 8.347511053085327
```

```
import sys

L = list(range(1000))
T = tuple(range(1000))

print('List size',sys.getsizeof(L))
print('Tuple size',sys.getsizeof(T))
```

```
List size 9120
Tuple size 8056
```

```
a = [1,2,3]
b = a

a.append(4)
print(a)
print(b)
```

```
[1, 2, 3, 4]
[1, 2, 3, 4]
```

```
a = (1,2,3)
b = a

a = a + (4,)
print(a)
print(b)
```

```
(1, 2, 3, 4)
(1, 2, 3)
```

Why use tuple?

Special Syntax

```
# tuple unpacking
a,b,c = (1,2,3)
print(a,b,c)
```

```
1 2 3
```

```
a,b = (1,2,3)
print(a,b)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-55-22f327f11d4b> in <module>
----> 1 a,b = (1,2,3)
      2 print(a,b)

ValueError: too many values to unpack (expected 2)
```

```
a = 1
b = 2
a,b = b,a

print(a,b)
```

```
2 1
```

```
a,b,*others = (1,2,3,4)
print(a,b)
print(others)
```

```
1 2
[3, 4]
```

```
# zipping tuples
a = (1,2,3,4)
b = (5,6,7,8)

tuple(zip(a,b))
```

```
((1, 5), (2, 6), (3, 7), (4, 8))
```

Sets

A set is an unordered collection of items. Every set element is unique (no duplicates) and must be immutable (cannot be changed).

However, a set itself is mutable. We can add or remove items from it.

Sets can also be used to perform mathematical set operations like union, intersection, symmetric difference, etc.

Characterstics:

- Unordered
- Mutable
- No Duplicates
- Can't contain mutable data types

Creating Sets

```
# empty
s = set()
print(s)
print(type(s))
# 1D and 2D
s1 = {1,2,3}
print(s1)
#s2 = {1,2,3,{4,5}}
#print(s2)
# homo and hetro
s3 = {1,'hello',4.5,(1,2,3)}
print(s3)
# using type conversion

s4 = set([1,2,3])
print(s4)
# duplicates not allowed
s5 = {1,1,2,2,3,3}
print(s5)
# set can't have mutable items
s6 = {1,2,[3,4]}
print(s6)
```

```
set()
<class 'set'>
{1, 2, 3}
{1, 4.5, (1, 2, 3), 'hello'}
{1, 2, 3}
{1, 2, 3}
```

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-71-ab3c7dde6aed> in <module>
     19 print(s5)
     20 # set can't have mutable items
---> 21 s6 = {1,2,[3,4]}
     22 print(s6)

TypeError: unhashable type: 'list'
```

```
s1 = {1,2,3}
s2 = {3,2,1}

print(s1 == s2)
```

```
True
```

Accessing Items

```
s1 = {1,2,3,4}
s1[0:3]
```

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-75-4c49b6b6050d> in <module>
      1 s1 = {1,2,3,4}
----> 2 s1[0:3]

TypeError: 'set' object is not subscriptable
```

Editing Items

```
s1 = {1,2,3,4}
s1[0] = 100
```

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-76-bd617ce25076> in <module>
      1 s1 = {1,2,3,4}
----> 2 s1[0] = 100

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

Adding Items

```
S = {1,2,3,4}
# add
# S.add(5)
# print(S)
# update
S.update([5,6,7])
print(S)
```

```
{1, 2, 3, 4, 5, 6, 7}
```

Deleting Items

```
# del
s = {1,2,3,4,5}
# print(s)
# del s[0]
# print(s)
# discard
# s.discard(50)
# print(s)
# remove
# s.remove(50)
# print(s)
# pop
# s.pop()
# clear
s.clear()
print(s)
```

```
set()
```

Set Operation

```
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}
s1 | s2
# Union(|)
# Intersection(&)
s1 & s2
# Difference(-)
s1 - s2
s2 - s1
# Symmetric Difference(^)
s1 ^ s2
# Membership Test
1 not in s1
# Iteration
```

```
for i in s1:
  print(i)
```

```
1
2
3
4
5
```

Set Functions

```
# len/sum/min/max/sorted
s = {3,1,4,5,2,7}
len(s)
sum(s)
min(s)
max(s)
sorted(s,reverse=True)
```

```
[7, 5, 4, 3, 2, 1]
```

```
# union/update
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

# s1 | s2
s1.union(s1)

s1.update(s2)
print(s1)
print(s2)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{4, 5, 6, 7, 8}
```

```
# intersection/intersection_update
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

s1.intersection(s2)

s1.intersection_update(s2)
print(s1)
print(s2)
```

```
{4, 5}
{4, 5, 6, 7, 8}
```

```
# difference/difference_update
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}
```

```
s1.difference(s2)

s1.difference_update(s2)
print(s1)
print(s2)
```

```
{1, 2, 3}
{4, 5, 6, 7, 8}
```

```
# symmetric_difference/symmetric_difference_update
s1 = {1,2,3,4,5}
s2 = {4,5,6,7,8}

s1.symmetric_difference(s2)

s1.symmetric_difference_update(s2)
print(s1)
print(s2)
```

```
{1, 2, 3, 6, 7, 8}
{4, 5, 6, 7, 8}
```

```
# isdisjoint/issubset/issuperset
s1 = {1,2,3,4}
s2 = {7,8,5,6}

s1.isdisjoint(s2)
```

```
True
```

```
s1 = {1,2,3,4,5}
s2 = {3,4,5}

s1.issuperset(s2)
```

```
True
```

```
# copy
s1 = {1,2,3}
s2 = s1.copy()

print(s1)
print(s2)
```

```
{1, 2, 3}
{1, 2, 3}
```

Frozenset

Frozen set is just an immutable version of a Python set object

```
# create frozenset
fs1 = frozenset([1,2,3])
fs2 = frozenset([3,4,5])

fs1 | fs2
```

```
frozenset({1, 2, 3, 4, 5})
```

```
# what works and what does not
# works -> all read functions
# does't work -> write operations

# When to use
# 2D sets
fs = frozenset([1,2,frozenset([3,4])])
fs
```

```
frozenset({1, 2, frozenset({3, 4})})
```

Set Comprehension

```
# examples

{i**2 for i in range(1,11) if i>5}
```

```
{36, 49, 64, 81, 100}
```

Dictionary

Dictionary in Python is a collection of keys values, used to store data values like a map, which, unlike other data types which hold only a single value as an element.

In some languages it is known as map or assosiative arrays.

dict = { 'name' : 'nitish' , 'age' : 33 , 'gender' : 'male' }

Characterstics:

- Mutable
- Indexing has no meaning
- keys can't be duplicated
- keys can't be mutable items

Create Dictionary

```
# empty dictionary
d = {}
d
```

```
# 1D dictionary
d1 = { 'name' : 'nitish' ,'gender' : 'male' }
d1
# with mixed keys
d2 = {(1,2,3):1,'hello':'world'}
d2
# 2D dictionary -> JSON
s = {
    'name':'nitish',
     'college':'bit',
     'sem':4,
     'subjects':{
         'dsa':50,
         'maths':67,
         'english':34
     }
}
s
# using sequence and dict function
d4 = dict([('name','nitish'),('age',32),(3,3)])
d4
# duplicate keys
d5 = {'name':'nitish','name':'rahul'}
d5
# mutable items as keys
d6 = {'name':'nitish',(1,2,3):2}
print(d6)
```

```
{'name': 'nitish', (1, 2, 3): 2}
```

Accessing items

```
my_dict = {'name': 'Jack', 'age': 26}
# []
my_dict['age']
# get
my_dict.get('age')

s['subjects']['maths']
```

```
67
```

Adding key-value pair

```
d4['gender'] = 'male'
d4
d4['weight'] = 72
d4
```

```
s['subjects']['ds'] = 75
s
```

```
{'name': 'nitish',
 'college': 'bit',
 'sem': 4,
 'subjects': {'dsa': 50, 'maths': 67, 'english': 34, 'ds': 75}}
```

Remove key-value pair

```
d = {'name': 'nitish', 'age': 32, 3: 3, 'gender': 'male', 'weight':
72}
# pop
#d.pop(3)
#print(d)
# popitem
#d.popitem()
# d.popitem()
# print(d)
# del
#del d['name']
#print(d)
# clear
d.clear()
print(d)

del s['subjects']['maths']
s
```

```
{}
```

```
{'name': 'nitish',
 'college': 'bit',
 'sem': 4,
 'subjects': {'dsa': 50, 'english': 34, 'ds': 75}}
```

Editing key-value pair

```
s['subjects']['dsa'] = 80
s
```

```
{'name': 'nitish',
 'college': 'bit',
 'sem': 5,
 'subjects': {'dsa': 80, 'english': 34, 'ds': 75}}
```

Dictionary Operations

- Membership
- Iteration

```
print(s)

'name' in s
```

```
{'name': 'nitish', 'college': 'bit', 'sem': 5, 'subjects': {'dsa': 80,
'english': 34, 'ds': 75}}

True
```

```
d = {'name':'nitish','gender':'male','age':33}

for i in d:
  print(i,d[i])
```

```
name nitish
gender male
age 33
```

Dictionary Functions

```
# len/sorted
len(d)
print(d)
sorted(d,reverse=True)
max(d)
```

```
{'name': 'nitish', 'gender': 'male', 'age': 33}

{"type":"string"}
```

```
# items/keys/values
print(d)

print(d.items())
print(d.keys())
print(d.values())
```

```
{'name': 'nitish', 'gender': 'male', 'age': 33}
dict_items([('name', 'nitish'), ('gender', 'male'), ('age', 33)])
dict_keys(['name', 'gender', 'age'])
dict_values(['nitish', 'male', 33])
```

```
# update
d1 = {1:2,3:4,4:5}
d2 = {4:7,6:8}

d1.update(d2)
print(d1)
```

```
{1: 2, 3: 4, 4: 7, 6: 8}
```

Dictionary Comprehension

```
{ key: value for vars in iterable }
```

```
# print 1st 10 numbers and their squares
{i:i**2 for i in range(1,11)}
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

```
distances = {'delhi':1000,'mumbai':2000,'bangalore':3000}
print(distances.items())
```

```
dict_items([('delhi', 1000), ('mumbai', 2000), ('bangalore', 3000)])
```

```
# using existing dict
distances = {'delhi':1000,'mumbai':2000,'bangalore':3000}
{key:value*0.62 for (key,value) in distances.items()}
```

```
{'delhi': 620.0, 'mumbai': 1240.0, 'bangalore': 1860.0}
```

```
# using zip
days = ["Sunday",
"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"]
temp_C = [30.5,32.6,31.8,33.4,29.8,30.2,29.9]

{i:j for (i,j) in zip(days,temp_C)}
```

```
{'Sunday': 30.5,
 'Monday': 32.6,
 'Tuesday': 31.8,
 'Wednesday': 33.4,
 'Thursday': 29.8,
 'Friday': 30.2,
 'Saturday': 29.9}
```

```
# using if condition
products = {'phone':10,'laptop':0,'charger':32,'tablet':0}

{key:value for (key,value) in products.items() if value>0}
```

```
{'phone': 10, 'charger': 32}
```

```
# Nested Comprehension
# print tables of number from 2 to 4
{i:{j:i*j for j in range(1,11)} for i in range(2,5)}
```

```
{2: {1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18, 10:
20},
 3: {1: 3, 2: 6, 3: 9, 4: 12, 5: 15, 6: 18, 7: 21, 8: 24, 9: 27, 10:
30},
```

```
 4: {1: 4, 2: 8, 3: 12, 4: 16, 5: 20, 6: 24, 7: 28, 8: 32, 9: 36, 10:
40}}

{
    2:{1:2,2:4,3:6,4:8},
    3:{1:3,2:6,3:9,4:12},
    4:{1:4,2:8,3:12,4:16}
}
```