Language Fundamental:-

Introduction:-

1.Python is a general purpose of high level programming laguage.

> There are 3 types of level of programming

>low level(dev:- slow, prod:- very fast)

>middle level(dev:- fast, prod:- slow)

>high level(oops(object oriented:- (Rule and regulation very hard)), pops(procedure oriented):- scrpting lang(Rule and regulation low))

2.Python was developed by guido van rossam in 1989, while working at Nation Research institute at Netherland.(C++,Perl= abc)

3.1991 feb 20th offically lunch , (python)

4. Pythin is scripting lang.

5.Python is very easy understand and user friendly, when we compare python with other languagaes.

Ex:- Print :- 'Hye, How are you?'

Java:-

```
public class Hye
{
	p s v[String[]args]
	{
		SOP("Hye, How are you?")
	}
}
```

C:-

```
#include<stdio.h>
void main()
{
	printf("Hye, How are you?")
}
```

Python:-

```
print("Hye, How are you?")
```

Where we can use python:-

1.Developing Web application.(search engine)

2.Developing desktop application.

3.Developing Database application.(Database server.application)

4.Network Prpgramming.(Telecom)

5.Automation Testing for all type of application.(everytype of application testing)

6.Data Analysis application(makeing data analysis apps)

7.Machine Learning(coomand/code machines)

8.Deep learning(coomand/code machines)

9.AI(Artificial Intelligence) application.(Football cameara,Robotics,srini,selfi camera,Tesla car)

10.IOT(Internet Of Thing)

11.Developing the Games

Limitation Of python:-

1.Mobile application and Mobile Realated Softwares

2.Banking Application

Features Of Python:-

1.simple and easy to learn.

2.Freeware and Open source.

3.High level Programming Language.

4.Platform Indepenedent(PVM).

5.Poratbility.

6.Dynamically Typed.

7.Both Procedure oriented and Object Oriented.

8.Extensive lib:- 65k+

Chapter:- 2

Identifiers:-

A name in python program is called identifier.

It can be (class name,function name, module name and variable name).

what is variable ?

ans:- variable is a storage area, where we can store our data, it's also provide us data's data type.

ex:-

variable_name = data

Rules to define identifiers in python:-

1.The only allowed charcters in python are:-

> alphabet symbols(upper case and lower case).
>
> digits(0 to 9)
>
> Underscore symbol(_), No other special Char

2.Identifiers should not start with the digits.

Ex:-

```
        >>> cuttack1 = "silver city"
>>> print(cuttack1)
silver city
>>> 1cuttack = "silver city"
SyntaxError: invalid syntax
>>>
```

3.Indentifiers are case sensitive, of course python language is case sensitive language.

Ex:-

```
> >>> result = 100
>>> Result = 90
>>>
```

```
>>> RESULT
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    RESULT
NameError: name 'RESULT' is not defined
>>> result
100
>>> Result
90
>>>
```

4.There is no length limit for python identifiers,But not recommended to use too length identifiers.

Casing:-

Camel Casing:- Folder name,Class Name, Module Name

Fisrt Char. of the word is always upper case, No space between the words and no special are used here.

Ex:-

StateBankOfIndia123

Snkae Casing:-File name,Variable name and function name

Upper case is not used in snake casing, and underscore(_) used in between the words.

Ex:-

state_bank_of_india

Python's Variable Types:-

----

x = 10 (Normal Variable)

_x = 10 (procted variable)

__x = 10 (Private Variable)

___x = 10 (Magic Variable)

Reserved Words(keyword):-

In python some words are reserved to represent some meaning of functionality. Such type of words are called Reserved words.

Before P3.6 :- 33 reserved words

After P3.6 :- 35 Reserved words

New Reserved words:- async and await.

Call keywords

```
>>> import keyword

>>> keyword.kwlist

['False', 'None', 'True', '__peg_parser__', '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']
```

Chapter:-3

Data Types:-

Data Type represent the type of data present inside a variable

In Python we are not required to specify the type explicily, Because python is a dynamically typed laguage.

variable = data

There are in 14 types od data avialable in Python(Inbuit Data types)

Name   Use in Python

1.Integer :- int()

2.Float   :- float()

3.Complex :- complex()

4.Boolean :- bool()

5.String   :- str()

6.List   :- list()

7.Tuple   :- tuple()

8.Set   :- set()

9.Frozenset :- frozensset()

10.Bytes   :- bytes()

11.Bytearray :- bytearray()

12.Range   :- range()

13.Dictionary :- dict()

14.None   :- None

Variable and Data:-

Data is fixed in python, and have a specific location. We create varibale and store data in varibale.

So that variable indicate data's location , so that data's location and variable's loaction is same.

Ex:-

```
>>> id(10)

2747102816848

>>> a = 10

>>> id(a)

2747102816848

>>> a = 20

>>> id(a)

2747102817168

>>>
```

Garbage Collection:-

In Old lanuages like C++,Java programmer is responsible for unnessary object destruction.

Usally programmer taking very much time while destructing object and mine while it's creating issues,

If any neglectance , total memory can be filled with useless object which creates meomory problems.

In python, We have some assitance which is always running in the backgound to destroy useless object.

Because this assitance the chance of falling python program with memory problems is very less.

This assiatant is nothing but Garbage Collector.

It's always delets old same variable objects,Hence the main object of Garbage collector is to destroy useless objects.

Integer:-int()

We can use int data type to represent whole number(Integral Values)

type():- check the data type of a variable.

Ex:-

```
>>> a = -10
>>> type(a)
<class 'int'>
>>>
>>> b = 99
>>> type(b)
<class 'int'>
>>>
```

Other Formats:-

Decimal Form

Binary Form

Octal   Form

Hexal   Form

Decimal Form:-(base-10)

It is the default number system in python. (float)

The allowed digits are :- 0 to 9

2.Binary Form:- (Base-2)

The allowed digits are :- 0 and 1

conver int to binary, we will use :- bin()

Literal value should be prefixed with '0b'

```
 >>> a = 10
>>> bin(a)
'0b1010'
>>> b = 11
>>> bin(b)
'0b1011'
```

```
>>> c = 12
>>> bin(c)
'0b1100'
>>>
```

Note:- After converted int to bin, converted data type is string.

3.Octal Form:-(Base-8)

The allowed digits are :- 0 to 7

conver int to octal, we will use :- oct()

Literal value should be prefixed with '0o'

```
 >>> a = 10
>>> oct(a)
'0o12'
>>> b = 11
>>> oct(b)
'0o13'
>>>
```

Note:- After convertinf int to oct, converted data type is string.

4.Hexal Form(Base-16)

The allowed digits are (0-9 and a-f)

conver int to gexal, we will use :- hex()

Literal value should be prefixed with '0x'

ex:-

```
>>> a = 10
>>> hex(a)
'0xa'
>>> b = 180
>>> hex(b)
'0xb4'
>>>
```

Float:- float()

We can use float data type to represent floating point values(decimal Values).

ex:-

```
>>> a = 1.0
>>> type(a)
<class 'float'>
>>>
>>> b = 78.896
>>> type(b)
<class 'float'>
>>>
>>> c = -9.0
>>> type(c)
<class 'float'>
>>>
>>> d = -99.87
>>> type(d)
<class 'float'>
>>>
```

Interger

```
>>> a = 98.5678
>>> int(a)
98
```

```
>>>
```

binary,octal,hexal

-we can not convert.(TypeError: 'float' object cannot be interpreted as an integer)

```
>>> a = 10.10
>>> type(a)
<class 'float'>
>>> bin(a)
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    bin(a)
TypeError: 'float' object cannot be interpreted as an integer
>>> ALL Mathmetical activity, we will impliment int and float.
```

complex Data Type:- complex()

Real + imaginary = Complex

a   +   bj

ex:-

```
>>> a = 2+3j
>>> type(a)
<class 'complex'>
```

```
>>>

>>> b = -2-3j

>>> type(b)

<class 'complex'>

>>>

>>> c = 3.8+8.9j

>>> type(c)

<class 'complex'>

>>>

>>> d = -8.9-9.9j

>>> type(d)

<class 'complex'>

>>>
```

In the real part if we use int value then we can specify that either by decimal,octal,binary or hexa form,

But imaginary part should be specified only by using decimal form(float()).

Ex:-

```
>>> c = 0b1010+3j

>>> c

(10+3j)

>>> d = 0o12+9j

>>> d
```

```
(10+9j)

>>>

>>> e = 78+0b1010j

SyntaxError: invalid syntax

>>>
```

Use:-

```
>>> a = 2+3j

>>> b = 4+7j

>>> a+b

(6+10j)

>>>

>>> a-b

(-2-4j)

>>>

>>> a = 2

>>> a = 2+3j

>>> b = 5+6j

>>> a*b

(-8+27j)

>>>
```

4. Boolean:- bool()

We can use this data type to represent boolean values.

The only allowed values for this data types are.

True and False

```
>>> a = True
>>> type(a)
<class 'bool'>

>>> b = False
>>> type(b)
<class 'bool'>
>>>
```

Note:- Internally Python represent True as 1 and False as 0.

```
>>> True+ True
2
>>> True+False
1
>>> False+False
0
```

```
>>>
```

String :- str()

A string is a sequenece of keyboard char enclosed within single quotes or Double quotes.

ex:-

```
>>> a = ''
>>> type(a)
<class 'str'>
>>>
\
>>> b = ' '
>>> type(b)
<class 'str'>
>>>
>>> c = ""
>>> type(c)
<class 'str'>
>>>
>>> d = "12468900!@$&**((((*^%$@   VYUJJJMNNFF hgfdrthhnbh"
>>> type(d)
<class 'str'>
```

Note:- Triple quotes to use for document string.

```
def sum():
        ''' This function is for sumation purpose'''

        a = 20
        b = 30
        c = a+b
        return c
```

Slicing of string:-

Slice means a piece.

[] operator is called slice operator, Which can be used to retrive parts of string.

Python Follows zero based index.

The index can be either +ve or -ve.

+ve :- left to right

-ve :- right to left

Formual :-

variable_name[start:end+1:step]

+ve index:-

----

ex:-

```
>>> A = "Scodeen"

>>> #Sco

>>> A[0:3:1]

'Sco'

>>> A[0:3]

'Sco'

>>> A[2:6]

'odee'

>>> A[2:6:1]

'odee'

>>> A[2:6]

'odee'

>>> A[1:6:2]

'cde'

>>> A[0:7:3]
```

```
'Sdn'

>>> A[0::3]

'Sdn'

>>>
```

-ve:-

```
>>> A = "Scodeen"

>>> A[-5:-1]

'odee'

>>> A[-7::2]

'Soen'

>>>
```

HW :-

```
Name = "Rahul Dravid"

Dob = "19/10/2022"

car_no = "KA-52-AG-0108"
```

Name(First_Char first + and sure name last) +DOB(Year)+ car_no(RTO No)

out:- Rd202252

Mathmetical Implimatation in string:-

Addition(+)

Sub(-)

mul(*)

div(/)

Note:- Only addition and multiplication is available in python.

Add:-

```
>>> st_1= 'Rahul'
>>> st_2 = 'Dravid'
>>>
>>> st_1+st_2
'RahulDravid'
>>>
```

String Methods:

----

dir() :- fetch methods

```
>>> s = "Scohool"

>>> dir(s)

['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find',
'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower',
'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition',
'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

>>> Note:- __ is private method, no need to impliment. Fouce on (capitalize-- zfill)

Note:- https://www.w3schools.com/python/python_ref_string.asp

How to use method or function in the data type?

Using dot(.) method, we will use data type functions.

syntax:-

variable.function_name()

Important Methods in string:-

1.capitalize():- Capital the first char in the string.

Ex:-

```
>>> Name = "krishna"
>>> Name.capitalize()

'Krishna'
```

2.upper():-Return the string in upper case.

```
>>> Name = "krishna"

>>> Name.upper()

'KRISHNA'

>>>
```

3.lower():- Return the string in lower case.

```
>>> Name = "Krishna NAYAK"

>>> Name.lower()

'krishna nayak'
```

4.swapcase():-lower case become upper acse and vice versa.

```
>>> Name = "Krishna NAYAK"

>>> Name.swapcase()

'kRISHNA nayak'

>>>
```

5.endswith(): Return true if the string ends with the specified values.

```
>>> data = "Hi All, How are you?"
>>> data.endswith('?')
True
>>> data.endswith('/')
False
```

6.startswith():- Return true if the string start with the specified values.

```
>>> data.startswith("H")
True
>>> data.startswith('h')
False
>>>
```

Very very imp

7.strip():- Removing the whitespaces from left and right side.

```
>>> first_name = '    krishna Nayak    '
>>> first_name.strip()
```

8.replace():- Replace string with another string.

```
>>> s = 'ji'

>>> s.replace('j','h')

'hi'

>>>

>>> s = 'ji jerry'

>>> s.replace('j','h')

'hi herry'

>>> s.replace('ji','hi')

'hi jerry'

>>>
```

9.count():- Return the number of times a specified values accurs in a string.

```
>>> s = 'apple is best, i love apple'

>>> len(s)

27

>>> s.count('p')

4

>>> s.count('i')

2

>>> s.count('e')

4
```

```
>>> s.count('best')

1

>>> s.count('apple')

2

>>>
```

10.find():-

11.index():-

```
2

>>> #find

>>>

>>> s = 'Harry is a magincian'

>>> s.find('y')

4

>>> s.find('i')

6

>>> s.index('y')

4

>>> s.index('i')

6

>>> s.index('ia')

17

>>> s.index('in')
```

```
14
>>>
>>> s.find('b')
-1
>>> s.index('b')
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    s.index('b')
ValueError: substring not found
>>>
```

12.join

13.split

14.format

12.split():- The split() method splits the string to the substring, using comman seprater.

```
>>> first_name = "Abhiram Abhiman Abhishek Abhilash Abhilipsa"
>>> first_name.split(' ')
['Abhiram', 'Abhiman', 'Abhishek', 'Abhilash', 'Abhilipsa']
```

if common separater is not avilabale use format()

```
>>> first_name = "Abhiram Abhiram Abhishek Abhilash Abhilipsa"

>>> first_name = "Abhiram Abhiman Abhishek Abhilash Abhilipsa"

>>> first_name.split(' ')

['Abhiram', 'Abhiman', 'Abhishek', 'Abhilash', 'Abhilipsa']
```

13.join():- join the different string in a single.

```
>>> li  = ['Krishna','Nayak']

>>> ''.join(li)

'KrishnaNayak'

>>> ' '.join(li)

'Krishna Nayak'

>>>
```

```
>>> li  = ['Krishna','Nayak']

>>> ''.join(li)

'KrishnaNayak'

>>> ' '.join(li)

'Krishna Nayak'

>>>
```

14.format():- The format method takes the passed arguments, format then add places them in the string where the placeholde {} are.

```
>>> #Name = "Chandra Sekhar Nayak"

>>> Name = "Chadr Sehar Naak"

>>> Name = "Cha{}dr{} Se{}har Na{}ak"

>>> Name.format('n','a','k','y')

'Chandra Sekhar Nayak'

>>>

>>> dob = "1{}/03{}1997"

>>> dob.format('7','/')

'17/03/1997'

>>>
```

flat file------------- char/.txt/.json/.bson

config file ------------ .exe cd /code/prod_cone/still_bayer/py_34.exe :- data

log file ----------------- error

Pycharm Download link:- https://www.jetbrains.com/pycharm/download/#section=windows

list:-

[45,89.90,7+9j,True,'scodeen',[12],(67,89)] = list

variable = ['Chandra','Sekhar','Nayak',25,1997,'Bhubaneswar',1997]

If we want to represent a group of values in a single entity then we represent this group as a list,

where insertion order is preserved and duplicate value also acceptable and every element seprate with each through comma(,) it's enclosed with in squre bracket. It'a mutable data type.

1.Hetrogenous object allowed.

2.Duplicate elements allowed.

3.Insertion order is preserved.

4.Growable in nature.

```
>>> li = [12,24,36]

>>> dir(li)

[ 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

```
li = [12,56,89]

#print(type(li))


# append:- add element in list last postion.

li = [12,56,89]

li.append(60)

print(li)

# output:- [12, 56, 89, 60]


# insert :- Insert an element of specified postion.
```

```
li = [12,56,89]

li.insert(0,'scodeen')

print(li)


#  output:- ['scodeen', 12, 56, 89]


#clear:- delete all the element in list and provide a empty list.


li = [56,89,78,90]

li.clear()

print(li)


#output :- []


#pop():- Delete last element of the list.


li = [12,34,67,89,70]

print(li.pop())

print(li)


# remove():-

li = [12,34,67,89,70]

li.remove(67)

print(li)
```

```
# output :- [12, 34, 89, 70]


#extend():- merge two list.


tenth_sec_a = ["Abhishek","Abhiram","Abhiman"]

tenth_sec_b = ["Akash","Akhankya","Alpana"]


tenth_sec_a.extend(tenth_sec_b)

print(tenth_sec_a)


#output:- ["Abhishek","Abhiram","Abhiman","Akash","Akhankya","Alpana"]


#copy():- copy the data one variable to another variable.


a = [89,90]

b = a.copy()

print(b)

print(id(a))

print(id(b))
```

Class:-12

```
#count:- A single element used a list, count this perticular elelment.
```

```
li = [10,20,30,40,50,60,60,20,20,10,10,40,20,20]

print(len(li))

print(li.count(50))


#index:- Return the index of first occurance starts and end index are not nessary parameter.


li = [10,20,30,40,50,60,60,20,20,10,10,40,20,20]

print(li.index(20))


#sort():- convert accending order


li = [344,87,567,123,90,78,56,34,987656]

li.sort() # accending

li.sort(reverse = True) # decending

print(li)


#reverse:- Reverse the list.

li = [67,90,56,878,545,8786,45]

li.reverse()

print(li)
```

link :- https://www.w3resource.com/python-exercises/list/

Tuple:- Tuple is a immutable data type, It's same as list but enclosed in Paranthesis '()', so that we can not modifie or change or update list elements.

```
t = (10,20,30)

type(t)

<class,'tuple'>
```

list:-

```
[45,89.90,7+9j,True,'scodeen',[12],(67,89)] = list
```

```
variable = ['Chandra','Sekhar','Nayak',25,1997,'Bhubaneswar',1997]
```

If we want to represent a group of values in a single entity then we represent this group as a list, where insertion order is preserved and duplicate value also acceptable and every element seprate with each through comma(,) it's enclosed with in squre bracket.

1.Hetrogenous object allowed.

2.Duplicate elements allowed.

3.Insertion order is preserved.

4.Growable in nature.

```
>>> li = [12,24,36]

>>> dir(li)

[ 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

```
li = [12,56,89]

#print(type(li))


# append:- add element in list last postion.

li = [12,56,89]

li.append(60)

print(li)

# output:- [12, 56, 89, 60]


# insert :- Insert an element of specified postion.

li = [12,56,89]

li.insert(0,'scodeen')

print(li)


#  output:- ['scodeen', 12, 56, 89]


#clear:- delete all the element in list and provide a empty list.


li = [56,89,78,90]

li.clear()

print(li)


#output :- []
```

```
#pop():- Delete last element of the list.


li = [12,34,67,89,70]

print(li.pop())

print(li)


# remove():-

li = [12,34,67,89,70]

li.remove(67)

print(li)


# output :- [12, 34, 89, 70]


#extend():- merge two list.


tenth_sec_a = ["Abhishek","Abhiram","Abhiman"]

tenth_sec_b = ["Akash","Akhankya","Alpana"]


tenth_sec_a.extend(tenth_sec_b)

print(tenth_sec_a)


#output:- ["Abhishek","Abhiram","Abhiman","Akash","Akhankya","Alpana"]


#copy():- copy the data one variable to another variable.
```

```
a = [89,90]

b = a.copy()

print(b)

print(id(a))

print(id(b))
```

Set:-

If we want to represent a group of values without duplicates, where order is not preserved or imp, then we should go for set data type.

1. Insertion order is not preserved.
2. Duplicates are not allowed.
3. Hetrogeneous object allowed.
4. Index concept is not applicable
5. It is mutable collection.
6. It's growable in nature.

add', 'clear', 'copy', 'difference', '#difference_update', 'discard',

'intersection', '#intersection_update', '#isdisjoint', '#issubset', '#issuperset', 'pop', 'remove',

'#symmetric_difference', '#symmetric_difference_update', 'union', 'update']

note: those who mark as # are not important

```
se = {34,87,56,46,78,34,34,87,87,90,90,90,65,56,56,34}


# add:-  Add an element to the set.

se = {34,87,56,46,78,34}

se.add(123)
```

```
print(se)

# op:- {34, 87, 56, 78, 123, 46}


#clear() :- Remove all the element from the set.

se = {34,87,56,46,78,34}

se.clear()

print(se)


#op:- set()


# copy():- Return a copy of the set.


a = {34,87,56,46,78,34}

b = a.copy()

print('b data:-',b)


# difference():- Return  a set that contain the item that only only exits in the first_set

            # not in second


a = {1,2,3,7,8}

b = {4,5,6,7,8}

print(a.difference(b)) # op :- {1, 2, 3}

print(b.difference(a)) # op :- {4, 5, 6}


# intersection :- Common element in the set.
```

```
a = {1,2,3,7,8}

b = {4,5,6,7,8}


print(b.intersection(a)) # op:- {8, 7}


# discard() :- Remove the specified item.

a = {1,2,3,7,8}

a.discard(3)

print(a) # {1, 2, 7, 8}


# remove() :- Remove the specified item.

a = {1,2,3,7,8}

a.remove(3)

print(a) # {1, 2, 7, 8}


#pop():- delete any element of the set(Randomly)

a = {1,2,3,7,8}

a.pop()

print(a) # {2, 3, 7, 8}


#union :- Return a set containing the union of the set.


a = {1,2,3,4,5,6}

b = {5,6,7,8,9,10}
```

```
print(a.union(b)) # {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

#update

a = {12,3,78,98,56}
b = {10,11,97,22}
a.update(b)
print(a) # {97, 98, 3, 10, 11, 12, 78, 22, 56}
```

Type Casting:-

We can convert one type value to another type , this conversion is called type casting or type coresion.

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

>>> list = [1,2,3,1,2,3]
>>> type(list)
<class 'list'>
>>>
>>> tu = (6,7,8,9)
>>> type(tu)
```

```
<class 'tuple'>

>>>

>>> li_1 = tuple(list)

>>> li_1

(1, 2, 3, 1, 2, 3)

>>> type(li_1)

<class 'tuple'>
```

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

>>> s = set(a)

>>> s

{1, 2, 3}

>>> type(s)

<class 'set'>

>>>
```

Frozenset():-

It is exactly same as set except that it is immutable, and we create a frozenset the help of typecasting.

```
>>> a = [1,2,3,4,5,5,5,6]

>>> fz = frozenset(a)

>>> fz

frozenset({1, 2, 3, 4, 5, 6})

>>> type(fz)<class 'frozenset'>>>>
```

#bytes():- bytes is a collection, it's contain 0-255 limit of values, if we put above of 255 then we will get the error, For creating bytes we required type casting method. it's a immutable data type

ex:-

```
>>> #bytes

>>> li = [12,56,78,45]

>>> by = bytes(li)

>>> by

b'\x0c8N-'

>>> for i in by:

        print(i)




12

56

78

45

>>> li = [12,56,78,45,255]

>>> by = bytes(li)

>>> by

b'\x0c8N-\xff'

>>>

>>> li = [12,56,78,45,255,256]

>>> by = bytes(li)

Traceback (most recent call last):

  File "<pyshell#17>", line 1, in <module>
```

```
    by = bytes(li)
ValueError: bytes must be in range(0, 256)
```

#bytearray():- same as byte, but it's mutable

Note:- When we complete the for the loop, then we start range data type.

# Dictionary():-

if we want to represent a group of values as key-value pairs then we should go for dict data type.

ex:- {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}

Duplicate keys are not allowed but values can be duplicate, If we are trying to insert an

entry duplicate key then we will get old key is replace new key. Dictionary is a mutable data type.

```
>>> di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}
>>> type(di)
<class 'dict'>
>>>
---------
```

```
>>> di['name']

'mahesh'

>>> di['name'] = 'Ranjan'

>>> di

{'name': 'Ranjan', 'age': 24, 'salary': 500000, 'address': 'mahesh'}


>>> dir(di)

['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__ior__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__',
'__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__ror__', '__setattr__', '__setitem__',
'__sizeof__', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop',
'popitem', 'setdefault', 'update', 'values']

>>>
```

```
di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}


#clear():- delete all the element from dictionary.

di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}

di.clear()

print(di) # op{}


#copy :- Return a copy in another variable


di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}

di_1 = di.copy()

print(di_1) # {'name': 'mahesh', 'age': 24, 'salary': 500000, 'address': 'mahesh'}
```

```
#fromkeys():- Return a dict with a specified key and value, and here value is same

li = ['Kiran','Sonal','Krishna','Javed']
marks = 80

z = dict.fromkeys(li,marks)
print(z) # {'Kiran': 80, 'Sonal': 80, 'Krishna': 80, 'Javed': 80}

# get():- Return the value of specified key,

di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}
print(di.get('salary')) # 500000

#items():- Return a list of tuple containing dictionary values.

di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}
print(di.items()) # dict_items([('name', 'mahesh'), ('age', 24), ('salary', 500000), ('address', 'mahesh')])

#keys():- Return a list of keys.

di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}
print(di.keys()) # dict_keys(['name', 'age', 'salary', 'address'])
```

```
#Values():- Return a list of values

di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}

print(di.values()) # dict_values(['mahesh', 24, 500000, 'mahesh'])


#pop():- Remove the element with the specified key.

di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}

di.pop('age')

print(di) # {'name': 'mahesh', 'salary': 500000, 'address': 'mahesh'}


#popitem():- Remove the last key and value from the dictionary

di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}

di.popitem()

print(di) # {'name': 'mahesh', 'age': 24, 'salary': 500000}


# setdefault():- Return the value of specofied key, If the key doesnot exits, insert key with

#specified values


di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}

di.setdefault('ph_no',9989786686)

print(di) # {'name': 'mahesh', 'age': 24, 'salary': 500000, 'address': 'mahesh', 'ph_no': 9989786686}


#update :-


di = {'name':'mahesh','age':24,'salary':500000,'address':'mahesh'}

new_di = {'ph_no': 986775,'hike_salary':250000,'company':'IBM'}
```

```
di.update(new_di)

print(di) #

 {'name': 'mahesh', 'age': 24, 'salary': 500000, 'address': 'mahesh', 'ph_no': 986775, 'hike_salary': 250000, 'company': 'IBM'}
```

#None Data type:-

None means nothing or No value associated

If the value is not available , then to handel such type of cases it's None.

```
def m1():

   a = 10

print(m1())
```

Escape Char:-

In string literal we can use escape char to associate a special meaning

\n = new line

\t = Horizental tab

\b = back space

\' = single quote

\" = double quote

\\ = back slash

Opeartor

Operator is a symbol that perform certain opeartion

1.Arithmetic

2.Relational or Comparision

3.Logical

4.Bitwise

5.Assignment

6.Special:- Membership and Identity

```
#Arthimetic

'''Symbols :- +,-,*,/,%(modulo),//(floor divison) ,**(Exponent or power)'''

a= 10
b = 20

print(a+b)
print(a-b)
print(a*b)
print(10/10)
print(10%5) # reminder == modulo
print(2**4) #power

>>> 10//3(When we are required flaot value this time we will use floar division)


#Relational operator


'''symbols:- >,>=,<,<=,==,!=(Not equal)'''


a = 10

b = 20

print(a>b) # False

print(20>=20) #True

print(a<b) # True

print(77<=45)#False


print(10 == 10) # True

print(10 == 11) #False


print(10!=10) # False

print(10!=11) # True


#Logical


'''symbol:- and,or,not'''
```

and :- If both arguments are True then only resuly is True.

or :- if atleast one argument is True then result is True

not :- Compliment(change the original Result)

```
>>> a = 10
>>> b = 20
>>> c = 30
>>> if a<b and a<c:
        print('print the value')



print the value
>>> if a>b and a<c: # first condition false
        print('print the value')



>>> if a<b or a<c:# both True
        print('print the value')



print the value
>>> if a<b or a>c:# one first one True
        print('print the value')



print the value
```

```
>>> if a>b or a>c:# both false

            print('print the value')




>>> not True

False

>>> not False
```

# Assignment:-

We can use assignment operator to assign value/data to the variable, with the help of = symbol.

a = 10

we can combine assigment operator with some other operators(arthimetic) to form compound assigment opeartor.

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

y = y-90 ---> y-=90

z = z*10---> z*=10

```
>>>

>>> x = 2
```

```
>>>
>>> x = x+10
>>> x
12
>>>
>>> y = 10
>>> y+=10
>>> y
20
```

#special:-

1.Idenity Operator

2.Membership Operator

1.Idenity Operator:-

we can use identity operator for location/address comparison

1.is

2. is not

```
>>> a = 10
>>> b = 20
>>> a is b

False

>>> b is a

False

>>> b = 10

>>> a is b

True

>>> b is a

True

>>> a is not b

False

>>>
```

2.Membership opeartor:

we acn use membership opearator to check whether the given object present in the collection or not(string,list,set,tuple,frozenset,dict)

in

not in

```
>>> a = [10,20,30,40]

>>> 40 in a

True

>>> 50 in a

False
```

```
>>> s = 'Today is karnataka Diwas'

>>> 'karnataka' in s

True

>>> 'Bihar' in s

False

>>>
```