1- PYTHON INTRODUCTION - (TASK - 1)

- I hope everybody install Anconda software which i share to you guys right
- Just wanted to know how many of know any programming language
- If you dont know any programming language then you are the best person to learn PYTHON
- python is very easy language
- what is python? Ans - python is highly recommanded programming language & object oriented language
- Father of python - Guido van Rosam
- Python came from fun tv show called "complete monty python's flying circus" - broadcasted in BBC channel
- Python borrowed all concept from c,c++,java,unix (so python is everything) thats why python very very powerfull tool
- Python developed in NRI - (Netherland) & lot of people say that python is new language
- Java released on 1995. python was released on 1989 officialy released on (feb 20th 1991)
- It has a large and comprehensive standard library.

In [100]:
```
b = '5'
type(b)
```
```
str
```

In [101]:
```
a = 6
a
```
```
6
```

In [102]:
```
a1 = 7
a1
```
```
7
```

In [103]:
```
a
```
```
6
```

In [104]:
```
a1
```
```
7
```

In [105]:
```
b
```
```
'5'
```

In [106]:
```
a
```
```
6
```

- Now python is very popular based on software industry requirment because everybody wants to write very less code/concile code
- Current market trend is - Machine learing, Artificial intelligence, data science & Iot(Internet of things)
- which companies are used python - google,nasa,uber,netfliz,reddit,facebook/meta, everywhere python used everywhere
- python code can understand everybody & python is dynamic programming language
- In python everything done by PVM (python virtual machine)

- you can access python in any platform independent- windows, linux, mac one code can run in all the 4 platform & no need to write separate programe for every platform. Once you write code you can run in platform
- Python is dynamically programming language (not required to declared data types)
- Python is freeware and open source. Moving from one platform to other platform without changeing any code
- Python contains rich libray - numpy,pandas so python is the best application for datascience
- which scenario python can't be used - (python can not perform in mobile application like android)
- Flavours of python - cpython(C programming),jpython(java programming),Iron python(c#.net),Ruby python(Ruby based application programme),Anaconda python(Bigdata,datascience)
- Python 1.0 introduce in jan 1994 -- Noorganization is working now
- Python 2.0 introduce in oct 2000 -- Noorganization is working now
- Python 3.0 introduce in Dec 2008, 2016, 2017,---- latest version - 3.6, 3.6, 3.7, 3.8, 3.9, 3.10

In [107]:
```
a = '5'
type(a)
```

```
str
```

In [108]:
```
import sys
sys.version
```

```
'3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]'
```

2- Getting started with Python Language

Python 3.x

Version Release Date --->

3.13 2022-25-08 3.10 2021-10-04 3.9 2020-10-05 3.8 2020-04-29 3.7 2018-06-27 3.6 2016-12-23 3.5 2015-09-13 3.4 2014-03-17 3.3 2012-09-29 3.2 2011-02-20 3.1 2009-06-26 3.0 2008-12-03

Python 2.x

Version Release Date

2.7 2010-07-03 2.6 2008-10-02 2.5 2006-09-19 2.4 2004-11-30 2.3 2003-07-29 2.2 2001-12-21 2.1 2001-04-15 2.0 2000-10-16

In [109]:
```
a = 5
a
type(a)
```

```
int
```

In [110]:
```
6 = b
```

```
  File "<ipython-input-110-eed0fedc6f5c>", line 1
    6 = b
       ^
SyntaxError: can't assign to literal
```

- Two major versions of Python are currently in active use:
- Python 3.x is the current version and is under active development.
- Python 2.x is the legacy version and will receive only security updates until 2020. No new features will be implemented.Note that many projects still use Python 2, although migrating to Python 3 is getting easier.
- If you want to learn python only then better you can use software called - python.org (below is url) https://www.python.org/downloads/
- For data science the best application for datascience models using python is called ANACONDA

In [111]:
```
## Verify if Python is installed
import sys
sys.version
```

```
'3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]'
```

3-Creating variables and assigning values

To create a variable in Python, all you need to do is specify the variable name, and then assign a value to it.

= Python uses = to assign values to variables There's no need to declare a variable in advance (or to assign a data type to it) Assigning a value to a variable itself declares and initializes the variable with that value. There's no way to declare a variable without assigning it an initial value.

In [112]:
```
# Integer
a = 2
type(a)
#a
```

```
int
```

In [113]:
```
# Integer
b = 9223372036
print(b)
```

```
9223372036
```

In [114]:
```
# Floating point
pi = 3.14
print(pi)
```

```
3.14
```

In [115]:
```
type(pi)
```

```
float
```

In [116]:
```
# String
c = 'A'
print(c)
```

```
A
```

In [117]:
```
type(c)
```

```
str
```

In [118]:
```
# String
name = 'John Doe'
print(name)
type(name)
```

```
John Doe
```

```
str
```

In [119]:
```
# Boolean
q = True
print(q)
```

```
True
```

In [120]:

```
# Empty value or null data type
x = None
print(x)
```

```
None
```

In [121]:

```
#Variable assignment works from left to right. So the following will give you an syntax error.
5 = x
```

```
  File "<ipython-input-121-849a83404c4b>", line 2
    5 = x
       ^
SyntaxError: can't assign to literal
```

In [122]:

```
x = 5
x
```

```
5
```

In [123]:

```
# 7th page continue onwards
```

4-PYTHON (IDENTIFIER / VARIBALE / OBJECT) --

- There is a person whose name - Multiple names are to identify person.so finally the Name which can be used for identification purpose.
- Name in the python programme is called IDENTIFIER (x = 10) (X - identifier)

*'''

- Nameing ceremoney we have some rules to naming a child . e.g - Gods name,Ancestor Name,have to do some R & D. you cannot keep the child name as - Cat or dog right.. so parent have to follow some rule and keep their child naming ceremony.

*Rules to define Python Identifier & we will check those rules ==

<1 Alphabet (uppercae & lowercase) <2> Digits (0-9) # should not stat with digit <3> underscore(_)

In [124]:

```
ABC = 50
AB
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-124-ad4b7fcac930> in <module>
      1 ABC = 50
----> 2 AB

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

In [125]:

```
NIT = 15000
nit
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-125-6e7f6d059173> in <module>
      1 NIT = 15000
----> 2 nit

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

In [126]:

```
cash123 = 10
cash123
```

```
10
```

In [127]:

```
123cash = 10
123cash
```

```
  File "<ipython-input-127-10566c0175f5>", line 1
    123cash = 10
          ^
SyntaxError: invalid syntax
```

In [128]:

```
#x = 10 # x is the variable & 10 is thealue
cash = 10 # Identifier ruls alphabet
#ca$h = 10 # $ - symbole is not allowed in python identifier but in java is allowed
#ca$h
cash
```

```
10
```

In [129]:

```
ca$h = 20
ca$h
```

```
  File "<ipython-input-129-46ab00ab9b66>", line 1
    ca$h = 20
      ^
SyntaxError: invalid syntax
```

In [130]:

```
ca*h = 20
ca*h
```

```
  File "<ipython-input-130-1321ee5401b5>", line 1
    ca*h = 20
            ^
SyntaxError: can't assign to operator
```

In [131]:

```
CASH = 20
#Cash
CASH
```

```
20
```

In [132]:

```
CASH1 = 30
#cash1
CASH1
```

```
30
```

In [133]:

```
# <2> Identifiers should not starts with dight ====
#sum123 = 20 # Digit rules identifier
123total = 30
123total
#sum123
```

```
  File "<ipython-input-133-db2dc179e917>", line 3
    123total = 30
           ^
SyntaxError: invalid syntax
```

In [134]:

```
# <3> Identifiers are case sensitive
#total = 10 # Python are case sensitive
Abcde = 20
#total
Abcde
```

```
20
```

In [135]:
```
new = 30
NEW
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-135-d5f52e6d3ddb> in <module>
      1 new = 30
----> 2 NEW

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

In [136]:
```
Total5 = 30
TOTAL
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-136-2e42157cb82b> in <module>
      1 Total5 = 30
----> 2 TOTAL

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

In [137]:
```
def = 4.6
def
```

```
  File "<ipython-input-137-74b2d234418d>", line 1
    def = 4.6
        ^
SyntaxError: invalid syntax
```

In [138]:
```
DEF = 4
DEF
```

```
4
```

In [139]:
```
if = 780
if
```

```
  File "<ipython-input-139-f54330b147fc>", line 1
    if = 780
       ^
SyntaxError: invalid syntax
```

In [140]:
```
if = 780
if
```

```
  File "<ipython-input-140-f54330b147fc>", line 1
    if = 780
       ^
SyntaxError: invalid syntax
```

In [141]:
```
DEF = 5.6
DEF
#def is key work
```

```
5.6
```

In [142]:
```
def = 7
def
```

```
  File "<ipython-input-142-7e3ce81fb797>", line 1
    def = 7
        ^
```

```
SyntaxError: invalid syntax
```

In [143]:
```
# <4> Keywords can not be assigned as identifier
#if = 10   # if is keyword
#DEF = 20  # def is keyword
for = 50
#DEF
#if
for
```

```
  File "<ipython-input-143-0581332fa669>", line 4
    for = 50
        ^
SyntaxError: invalid syntax
```

In [144]:
```
FOR = 58
FOR
```

```
58
```

In [145]:
```
def = 30
def
```

```
  File "<ipython-input-145-d7d2371477df>", line 1
    def = 30
        ^
SyntaxError: invalid syntax
```

In [146]:
```
if = 30
if
```

```
  File "<ipython-input-146-23187ed1a6a1>", line 1
    if = 30
       ^
SyntaxError: invalid syntax
```

In [147]:
```
IFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFdfgsdfgsfgsdfgsdfgfdgsfg = 56
#IFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
IFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFdfgsdfgsfgsdfgsdfgfdgsfg
```

```
56
```

In [148]:
```
# 5> NO length limit in python Identifier
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwxxxx
xxxx = 10
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwxxxx
xxxx
```

```
10
```

In [149]:
```
_abc_def_gef  = 20
_abc_def_gef
```

```
20
```

- Q & A for valid / Invalid identifier - 1>123AMX 2>Amx123 3>ml2ai 4>_abc_def_gef 5>def 6>else 7>ELSE
- ----- RULES OF PYTHON IDENTIFIER ------ 1> A to Z, a to z, 0 - 9 2> Doesnot starts with digit 3> Case sensitive 4> Reserved words or keywords cannot be a identifier 5> Identifier cannot have a lenght limit 6> _ only allowed 7> NO special character is allowed

- ========================= WE LEARNT ABOUT PYTHON IDENTIFIER IN DETAILS ============================

PYTHON RESERVED WORDS -

- if a kid going to school what he/she will learn A,B,C - - - -Z then she will learn A - APPLE, B -BALL, C - CAT. (APPLE,BALL,CAT - Reserved word in english)
- Apple is reserved for the fruit, Ball ==> play, Cat ==> Animal // (Dictionary uncountable reserved words is there).. This type of words are called Reserved word
- In any programming language there is a reserved word are there we gonna learn only python Reserved
- python reserved are => (35 RESERVED WORDS) If you learn 35 reserved words then python is complete
- all reserved words have some meaning & functionality
- Learning python is nothing but learing all this functionality

**35 RESERVED WORDS---

- True, False, None ==> Represent Boolean data types
- and, or, not, is ==> Represent the operators
- if, else, elif ==> Represent the statement (# python switch,do..while statament is not available)
- while, for, break, continue, return, in, yield ==> Represent the loop concept
- try, except, finally, raise, assert ==> Represent for functionallity
- import,from,as,class,def,pass,global,nonlocal,lambda,del,with==>Represent the class,method,function

**NOTES* -- 35 RESERVED WORDS ARE (ALPHABET) // *EXCEPT (True,False,None)

In [150]:
```python
#a = True # hash is used for comment
a = True
a
```
```
True
```

In [151]:
```python
a1 = true
a1
```
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-151-ed32791c6c8c> in <module>
----> 1 a1 = true
      2 a1

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

In [152]:
```python
True = a
```
```
  File "<ipython-input-152-26ab90e49180>", line 1
    True = a
            ^
SyntaxError: can't assign to keyword
```

In [153]:
```python
b = None
#b = none
b
```

In [154]:
```python
c = False
#c = false
c
```
```
False
```

In [155]:
```python
# How to remembr all keywords --- (Interview questions)
# KEYWORD is the module run from IMPORT class
import keyword
keyword.kwlist
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']
```

In [156]:

```
# write a coding to create index of these keywords --- IMP ---

import pandas as pd # pandas is the module to create a dataframe
df = pd.DataFrame(keyword.kwlist)
df

# Always remember python Index begins with '0'

# ================= RESERVED WORDS COMPLETED ================================
```

	0
0	False
1	None
2	True
3	and
4	as
5	assert
6	async
7	await
8	break
9	class
10	continue
11	def
12	del
13	elif
14	else
15	except
16	finally
17	for
18	from
19	global
20	if

	0
20	if
21	import
22	in
23	is
24	lambda
25	nonlocal
26	not
27	or
28	pass
29	raise
30	return
31	try
32	while
33	with
34	yield

PYTHON DATA TYPES // (14) - INBUILD DATA TYPES -

1>int 2>float 3>complex 4>bool 5>str 6>bytes 7>bytearray DATA STRUCTURE ---> 8>range 9>list 10>tuple 11>set 12>frozenset 13>dict 14>None

- python provides some inbuild function like -- <1> print() <2> type() <3> id()
- int,float,complex,boolen is not represent object # Tricky question
- except these 4 everythig object # Tricky question

*NOTE - [**In python all 14 data types are object only]* Thats why we called as python is object oriented program

'hello world'

In [157]:

```
# What is other inbuild datatype available except 14 datatypes-

a = 10
#print(a)       # To find the variable
#print(a)         # To find the value of variable
#type(a) # To find the data type
id(a)           # To find an address of an object
#type(a)
```

```
140719275483824
```

In [158]:

```
b = 10
id(b)
```

```
140719275483824
```

In [159]:

```
c = 20
id(c)
```

```
140719275484144
```

In [160]:

```
a = 10
b = 10
id(a)
```

```
140719275483824
```

In [161]:

```
a = 10
a
```

```
id(a)
```

```
140719275483824
```

int datatypes -

- INT Datatypes - The No.without decimal point are called as INTEGRAL DATATYPES *int datatype how many ways represent values in 3ways -

2> Binary form --- (Base-2) -- (0,1) 3> Octal form --- (Base-8) -- (0,7)

In [162]:

```
a1 = 4809
#a
#print(a)
type(a1)
```

```
int
```

In [163]:

```
#2.Binary form(Base 2)
#a = 1111   # value is declared
b = 0b111 # Now pvm convert value to binary value
b
#a
```

```
7
```

In [164]:

```
b_1 = 0b11
b_1
```

```
3
```

In [165]:

```
b2 = 0b22
b2
```

```
  File "<ipython-input-165-1f01a4962fec>", line 1
    b2 = 0b22
          ^
SyntaxError: invalid token
```

In [166]:

```
b1 = 111
b1
```

```
111
```

In [167]:

```
c = 0b1111
c
```

```
15
```

In [168]:

```
b3 = 0b222
```

```
  File "<ipython-input-168-42cd4ff784e0>", line 1
    b3 = 0b222
          ^
SyntaxError: invalid token
```

In [169]:

```
b = 0b11 # Now pvm convert value to binary value
```

```
b
```

```
3
```

In [170]:
```
#3. Octal form(Base 8)
#a = 111 # Value is declared
b1 = 0o111 # Now pvm covert value to octal value
b1
#a
```

```
73
```

In [171]:
```
# final summary of INTEGRAL DATATYPES
a = 10
b = 0b10
c = 0o100
a
b
c
```

```
64
```

In [172]:
```
c1 = 0o33
c1
```

```
27
```

In [173]:
```
b
```

```
2
```

In [174]:
```
c
```

```
64
```

In [175]:
```
A = 78
type(A)
```

```
int
```

float datatypes -

*employee sal - 5676.76*diesel price - 67.25

- These values are not integral value this is called as decimal value
- Floating datatype you cannot declare Binary,Octal & Hexadecimal because python enterpretur not accept that
- In our schools we learn about EXPONTIAL form -(1.2e3) this you can find in float datatypes & only letter 'e' can allowed

In [176]:
```
b = 67.9
b
type(b)
```

```
float
```

In [177]:
```
b1 = 0b1
b1
```
```
1
```
In [178]:
```
c = 0o11.6
c
```
```
  File "<ipython-input-178-b2060ef25c19>", line 1
    c = 0o11.6
            ^
SyntaxError: invalid syntax
```
In [179]:
```
# float datatypes -
#b2 = 0b1111.22
#b2
#print(type(b))
#type(b2)
#b
d = 0o4567.67 # This is octal
#e = 0b4567.89 # This is binary
d
#e
#b
```
```
  File "<ipython-input-179-3b6d108cd0ab>", line 7
    d = 0o4567.67 # This is octal
              ^
SyntaxError: invalid syntax
```
In [180]:
```
f1 = 1E1
f1
#type(f1)
```
```
10.0
```
In [181]:
```
#f = 1e4# only 'e' letter is allowed
g = 2.4E3 # except 'E' you can't execute any programme
g
#f
#type(g)
```
```
2400.0
```
In [182]:
```
g = 2.4E3
g
```
```
2400.0
```
In [183]:
```
g1 = 23e3
g1
```
```
23000.0
```
In [ ]:
```
e = 5.e3
#type(e)
e
```

complex datatypes -

- Complex datatype format are:-(a+bj) (a--Real part/b--Imaginary part/j^2=-1)
- j is the compulsory value & there is no other value accepted in complex type
- j^2 = -1
- Value of j is (j square is equal to -1) (j =(square root of -1) is equal to (j^2 = -1) pure mathemetics so if you want to develop mathmetic application or scientific application then python is the best option
- Real type any type base can be accepted but imaginary part allow only integer

In [184]:
```
x = 30+40j #assigned int value in real part & imaginary part
x
#type(x)
```
```
(30+40j)
```

In [185]:
```
type(x)
```
```
complex
```

In [188]:
```
y = 20.5+2.3j #assigned float value in real part & imaginary part
z = 30.8+20j  #assigned float value in real part & real value in imaginary part
y + z
y*z
y/z
```
```
(0.5022837821805671-0.25148297544192666j)
```

In [191]:
```
#c = 15+0b111j # Imaginary part cannot be binary,octal
d = 0o11+15j # Real part can be binary,octal
#c
d
```
```
(9+15j)
```

In [192]:
```
d2 = 0b111+15j
d2
```
```
(7+15j)
```

In [193]:
```
e1 = 4 + 15a
e1
```
```
  File "<ipython-input-193-3c54a5edf427>", line 1
    e1 = 4 + 15a
              ^
SyntaxError: invalid syntax
```

In [197]:
```
a1 = 20+30j
b1 = 40+50j
a1+b1
a1-b1
a1*b1
a1/b1
```
```
(0.5609756097560976+0.04878048780487805j)
```

In [198]:
```
a = 2+3J
type(a)
```

```
complex
```

In [201]:
```
a1 = 10+20j  # I want to know what the value of real part & imaginary part
a1.real   # complex data type will use in mathmetic concept not that required for programming language
#a1.imaginary
a1.imag
```

```
20.0
```

bool datatypes -

- True/False - (only allowed boolean values)
- False value -- 0 (internally memory level conversion happened)
- True value -- 1

In [ ]:
```
a = 10
b = 20
c = a>b
c
```

In [ ]:
```
True+True
True*True
True-True
True/True
False+False
False+True
True/False
```

In [ ]:
```
True/True
```

In [ ]:
```
True/False # error
```

str datatypes -

- enclosed in '' (single quote) // "" (double quote)
- singa line we assined as '' // ""
- multiline we assigned as (''' ''')
- single & double quotes are allowed only for single line
- triple quotes are allowed for multi comments & also you can declare triple quotes in single line as well

In [205]:
```
naresh = '''good for datascience'''
naresh
#type(naresh)
```

```
'good for datascience'
```

In [ ]:
```
naresh = '''good for datascience'''
#type(careerera)#' single quotes'
naresh
type(naresh)
```

In [208]:
```
naresh2 = '''good
            for datascience'''
#type(careerera)#' single quotes'
```

```
naresh2
#type(naresh)
```

```
'good \n            for datascience'
```

```
In [209]:
naresh = '''good for datascience'''
#type(careerera)#' single quotes'
naresh
#type(naresh)
```

```
'good for datascience'
```

```
In [210]:
naresh1 = '''good for
 datascience'''
#type(naresh1
naresh1
```

```
'good for \n datascience'
```

```
In [213]:
import keyword
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']
```

```
In [211]:
a = '''hallo
how
are
you'''

a
#type(a) # SINLE & DOUBLE QUOTES ARE EQUAL
```

```
'hallo\nhow \nare \nyou'
```

```
In [ ]:
b = '''hello
    hi'''
b
```

```
In [ ]:
# single quote & double both are equl
# ''' multiline comment'''
```

```
In [ ]:
b = '''('hallo'
  'how'
    'are you')'''

    # ''' ''' ==> triple quotes are used for multiline comments
b
    # is for commenting
```

Type casting or Type conversion -

int() -- float() -- complex() -- bool() -- str()

```
In [218]:
# int(): - you can convert from other type to int type except complex

int(10.123)  # float to int
#int(10+20j) # cannot convert from complex to int
int(True)     # Bool to int
int(False)   # Boll to int
int('10')     # string to int
int('ten') # amx is a character
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-218-5e96a1ce4bb7> in <module>
      6 int(False)   # Boll to int
      7 int('10')     # string to int
----> 8 int('ten') # amx is a character

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

```
In [224]:
# float(): -- convert from any type to float except complex

float(10)      # int to float
#float(10+20j)  # cannot convert complex to float
float(False)     # boolean to float
float('11')     # string to float
```

```
11.0
```

```
In [239]:
# complex(): -- covert any other type to complex type
# --- this only for 1 argument ------

complex(10)      # int to complex
complex(10.5)  # float to complex
complex(True)  # Bool to complex
complex(False) # Bool to complex
complex('10')  # string to complex

# ----- Now we will check for 2 arguments ------
complex(10,20)    # int to complex
complex(10,20.5) # float to complex
#complex(10, 0b11j)
complex('10')     # string to complex, you cannot assign 2 argument
```

```
(10+0j)
```

In [235]:
```
a1 = 10+20j
type(a1)
a1.real
a1.imag
```

```
20.0
```

In [249]:
```
# Bool()  - (0 means false // 1 means non zero)

bool(0)      # int to bool
bool(-10)     # int to bool
bool(0.0)    # float to bool
bool(0.01)   # float to bool
bool(10+20j) # complext to bool
bool(0+1j)   # complex to bool
bool(" ")      # string to bool(if argument is empty string then false)
bool('abc')# string to bool(if argument is not empty string then true)
bool(' ')    # space is also treated as character so non empty string
```

```
True
```

In [ ]:
```
bool(-10)
```

In [ ]:
```
bool(0+1j)
```

In [254]:
```
# str(): --- any type is possible in string

str(10)     # int to string
str(10.50)  # float to string
str(True)   # bool to string
str(10+20j) # complex to string
```

```
'(10+20j)'
```

In [255]:
```
z1 = str(10)
z1
```

```
'10'
```

- Fundamental Datatypes are which we covered so far & also we saw how to work on the type casting from one data type to other -
- We cannot convert our complex data types to int and float

  int() float() complex() bool() str()

Fundamental datatypes vs Immutability --

- All fundamental datatypes are immutable. what is immutable - once we crate the object we are not allow to perform any changes in that object . we can say that (non-changeable behaviour)
- why immutability concept is required -- if you look at below exampl - how many object we created only 1 object which is 10 but how many reference we assinged -- 3 reference indicates to 1 object
- bigest advantage of this approch is memory utilization & performance is also improved (pvm do not want to wsat memory)
- you can create object with different name, but you cannot create object with same name

```
x2 = 10
y2 = 10
z2 = 20
print(id(x2))
print(id(y2))
print(id(z2))
```

- Mutable -- Changeable-- once you create an object
- Immutable -- Non-changeable
- Fundamental data types are IMMUTABLE but (LIST is mutable)
- Everything in python is an object

**This concept reusing same object such type of concept is define following ranges - 1> int ----> 0 to 256 2> bool ---> Always 3> str ----> Always 4> float & complex ----> Can not performe the reusable concept

In [256]:

```
x = 10 #id - adddres of the memory location
y = 10
print(id(x))
print(id(y))
```

```
140719275483824
140719275483824
```

In [257]:

```
id(y)
```

```
140719275483824
```

In [261]:

```
# is operator
x = 20 # x,y = 20
y = 20
x is y
y is x
```

```
True
```

In [264]:

```
x = True
y = True
z = False
x is y
y is z
z is x
z is y
```

```
False
```