BASIC CODE

In [10]:
```
print(3+2)    #addition
print(3-2)    #substraction
print(3*2)    #multiplication
print(3/2)    #division
print(3**2)   #exponential
print(3 % 2)   # modulus
print(3//2)    #  floor division operator
```

```
5
1
6
1.5
9
1
1
```

In [12]:
```
print(type(10))
print(type(3.1))
print(type(1 + 3j))
print(type('anuja pawar'))
print(type([1,2,3]))
print(type({'name':'pawar'}))
print(type({9.8,3.14,2.7}))
print(type((9.8,3.14,2.7)))
print(type(3 == 3))
print(type(3 >= 3))
```

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

DATATYPES ,VARIABLES

In [17]:
```
9
```

Out[17]:
```
9
```

In [19]:
```
9 + 9
```

Out[19]:
```
18
```

```
In [21]: 9 + 9 - (10 - 3) + 3

Out[21]: 14

In [23]: 9 + 9 - 10 - 3 + 3

Out[23]: 8
```

arithmetic operator

```
In [36]: 10 + 5

Out[36]: 15

In [38]: 10 - 5

Out[38]: 5

In [40]: 10 * 5

Out[40]: 50

In [42]: 10 * 2

Out[42]: 20

In [44]: 10 ** 2

Out[44]: 100

In [46]: 10 *** 2

  Cell In[46], line 1
    10 *** 2
         ^
SyntaxError: invalid syntax

In [52]: 10 / 5    #float division

Out[52]: 2.0

In [54]: 10 // 5      # int division

Out[54]: 2

In [60]: 10 % 5      #reminder after division

Out[60]: 0

In [62]: 15 % 6

Out[62]: 3

In [64]: 15 %% 6
```

```
  Cell In[64], line 1
    15 %% 6
        ^
SyntaxError: invalid syntax
```

```
In [66]: 15 / 6
Out[66]: 2.5
```

```
In [68]: 15 // 6
Out[68]: 2
```

assignment operator

```
In [133…: x =10
          x
Out[133…: 10
```

```
In [135…: x + 2
Out[135…: 12
```

```
In [137…: x += 2
          x
Out[137…: 12
```

```
In [139…: x += 2
          x
Out[139…: 14
```

```
In [141…: x += 2
          x
Out[141…: 16
```

```
In [143…: x += 2
          x
Out[143…: 18
```

```
In [145…: x
Out[145…: 18
```

```
In [147…: x -= 2
          x
Out[147…: 16
```

```
In [149…: x -= 2
```

```
x
```

Out[149… 14

In [151…
```
x -= 2
x
```

Out[151… 12

In [153…
```
x
```

Out[153… 12

In [155…
```
x *= 2
x
```

Out[155… 24

In [157…
```
x *= 2
x
```

Out[157… 48

In [159…
```
x /= 2
x
```

Out[159… 24.0

In [161…
```
x /= 2
x
```

Out[161… 12.0

unary operator

In [165…
```
n = 7
n
```

Out[165… 7

In [167…
```
m = -n
m
```

Out[167… -7

python datatypes

1.INT

In [171…
```
i=45
print(i)
print(type(i))
```

```
45
<class 'int'>
```

In [173…

```
i1,i2 = 10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[173], line 1
----> 1 i1,i2 = 10

TypeError: cannot unpack non-iterable int object
```

In [177…

```
i1, i2 = 10,20
print(i1)
print(i2)
print(i1 + i2)
print(i1 - i2)
print(i1 * i2)
print(i1 / i2)
```

```
10
20
30
-10
200
0.5
```

FLOAT

In [180…

```
petrol = 110.56
petrol
```

Out[180…

```
110.56
```

In [182…

```
type(petrol)
```

Out[182…

```
float
```

string

In [187…

```
s = naresh it
s
```

```
  Cell In[187], line 1
    s = naresh it
               ^
SyntaxError: invalid syntax
```

In [189…

```
s = 'naresh it'
s
```

Out[189…

```
'naresh it'
```

In [200…
```
type(s)
```
Out[200…
```
str
```
In [202…
```
s1 = "nareshit"
s1
```
Out[202…
```
'nareshit'
```
In [204…
```
s2= '''naresh it technology
        datascience, ai student -- 6 month i will change your brain'''
s2
```
Out[204…
```
'naresh it technology \n        datascience, ai student -- 6 month i will chang
e your brain'
```
In [206…
```
s
```
Out[206…
```
'naresh it'
```
In [208…
```
print(s[0])
print(s[-1])
print(s[3])
```
```
n
t
e
```
In [210…
```
s
```
Out[210…
```
'naresh it'
```
In [212…
```
s[:]
```
Out[212…
```
'naresh it'
```
In [214…
```
s[2:5]
```
Out[214…
```
'res'
```

boolean

In [217…
```
true
```
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[217], line 1
----> 1 true

NameError: name 'true' is not defined
```
In [219…
```
True
```
Out[219…
```
True
```

In [221…
```
false
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[221], line 1
----> 1 false

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

In [223…
```
False
```

Out[223…
```
False
```

In [225…
```
b = True
b1 =False
```

In [227…
```
b
```

Out[227…
```
True
```

In [229…
```
b1
```

Out[229…
```
False
```

In [231…
```
b + b1
```

Out[231…
```
1
```

In [233…
```
print(b-b1)
print(b*b1)
print(b1/b)
print(b1//b)
```

```
1
0
0.0
0
```

In [237…
```
type(b1)
```

Out[237…
```
bool
```

complex

In [240…
```
c1 = 10 + 20j
c1
```

Out[240…
```
(10+20j)
```

In [242…
```
type(c1)
```

Out[242…
```
complex
```

In [244…
```
c1
```

```
Out[244… (10+20j)
```

```
In [246… c1.real
```

```
Out[246… 10.0
```

```
In [248… c1.imag
```

```
Out[248… 20.0
```

```
In [250… c2 = 3 + 5j
         c2
```

```
Out[250… (3+5j)
```

```
In [252… print(c1)
         print(c2)
```

```
(10+20j)
(3+5j)
```

```
In [254… c1 + c2
```

```
Out[254… (13+25j)
```

TYPE CASTING OR TYPE CONVERSION

```
In [257… int(3.14)
```

```
Out[257… 3
```

```
In [259… int(3.4,5.7)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[259], line 1
----> 1 int(3.4,5.7)

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

```
In [261… int(True)
```

```
Out[261… 1
```

```
In [263… int(True,False)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[263], line 1
----> 1 int(True,False)

TypeError: int() can't convert non-string with explicit base
```

```
In [265… int(False)
```

Out[265…
```
0
```

In [274…
```
print(int(3.4))
print(int(True))
print(int('10'))
```

```
3
1
10
```

In [276…
```
print(int(10+20j))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[276], line 1
----> 1 print(int(10+20j))

TypeError: int() argument must be a string, a bytes-like object or a real number, 
not 'complex'
```

In [278…
```
int('ten')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[278], line 1
----> 1 int('ten')

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

python variable

In [281…
```
va = 9
va
```

Out[281…
```
9
```

In [283…
```
id(va)
```

Out[283…
```
140724733946552
```

In [285…
```
1nit = 18
1nit
```

```
  Cell In[285], line 1
    1nit = 18
    ^
SyntaxError: invalid decimal literal
```

In [287…
```
nit1 = 18
nit1
```

Out[287…
```
18
```

In [291…
```
nit2 = 19
nit2
```

Out[291…

```
19
```

In [293…

```
v$ = 90
v$
```

```
  Cell In[293], line 1
    v$ = 90
     ^
SyntaxError: invalid syntax
```

In [295…

```
v_ =90
v_
```

Out[295…

```
90
```

In [297…

```
import keyword
keyword.kwlist
```

Out[297…

```
['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 [299…

```
len(keyword.kwlist)
```

Out[299…

```
35
```

In [301…
```
for = 67
for
```

```
  Cell In[301], line 1
    for = 67
        ^
SyntaxError: invalid syntax
```

In [311…
```
For = 67
For
```

Out[311…
```
67
```

In [313…
```
def = 90
def
```

```
  Cell In[313], line 1
    def = 90
        ^
SyntaxError: invalid syntax
```

In [315…
```
Def = 78
Def
```

Out[315…
```
78
```

In [317…
```
3a =89
```

```
  Cell In[317], line 1
    3a =89
    ^
SyntaxError: invalid decimal literal
```

In [319…
```
True = 8
```

```
  Cell In[319], line 1
    True = 8
    ^
SyntaxError: cannot assign to True
```

In [321…
```
true=8
true
```

Out[321…
```
8
```

In [323…
```
a = 5
b = 6
c = 7

a
b
c
```

Out[323…
```
7
```

In [325…
```
a = 5
b = 6
c = 7
```

```
print(a)
print(b)
print(c)
```

```
5
6
7
```

In [327…

```
import sys
sys.version
```

Out[327…

```
'3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.192
9 64 bit (AMD64)]'
```

In [329…

```
import_1 = 89
import_1
```

Out[329…

```
89
```

STRING

In [27]:

```
# single line comment
letter = 'p'
print(letter)
print(len(letter))
greeting = 'Hello, World!'
print(greeting)
print(len(greeting))
sentence = " I hope you are enjoying 30 days of python challenge"
print(sentence)
```

```
p
1
Hello, World!
13
 I hope you are enjoying 30 days of python challenge
```

In [29]:

```
# multiline string
multiline_string = '''I  am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python.'''
print(multiline_string)
```

```
I  am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python.
```

In [31]:

```
# another way of doing same thing
multiline_string = """I  am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python."""
print(multiline_string)
```

```
I  am a teacher and enjoy teaching.
I didn't find anything as rewarding as empowering people.
That is why I created 30 days of python.
```

In [33]:
```
# string concatenation
first_name = 'Anuja'
last_name = 'Pawar'
space= '  '
full_name = first_name + space + last_name
print(full_name)
```

```
Anuja  Pawar
```

In [35]:
```
#checking length of a string using len() builtin function
print(len(first_name))
print(len(last_name))
print(len(first_name)> len(last_name))
print(len(full_name))
```

```
5
5
False
12
```

In [37]:
```
# unpacking characters
language = 'python'
a,b,c,d,e,f = language      #unpacking sequence characters into variables
print(a)
print(b)
print(c)
print(d)
print(e)
print(f)
```

```
p
y
t
h
o
n
```

In [44]:
```
# accessing characters in strings by index
language = 'Python'
first_letter = language [0]
print(first_letter)
second_letter = language[1]
print(second_letter)
last_index = len(language) -1
last_letter = language[last_index]
print(last_letter)
```

```
P
y
n
```

In [46]:
```
# if we want to start from right end we can use negative indexing
language = 'Python'
last_letter = language[-1]
print(last_letter)
second_last = language[-2]
print(second_last)
```

```
n
o
```

In [48]:
```
#slicing

language = 'Python'
first_three = language[0:3]
last_three = language[3:6]
print(first_three)
print(last_three)
```

```
Pyt
hon
```

In [50]:
```
# another way
last_three = language[-3:]
print(last_three)
```

```
hon
```

In [52]:
```
#skipping character while splitting in python string
language = 'Python'
pto = language[0:6:2]
print(pto)
```

```
Pto
```

In [54]:
```
#escape sequence
print('I hope everyone enjoying the python challenge.\nDo you?')
print('Days\tTopics\tExercises')
print('Day 1\t3\t5')
print('Day 2\t3\t5')
print('Day 3\t3\t5')
print('Day 4\t3\t5')
print('This is a back slash symbol (\\)')
print('In every programming language it starts with \"Hello, World!\"')
```

```
I hope everyone enjoying the python challenge.
Do you?
Days    Topics  Exercises
Day 1   3       5
Day 2   3       5
Day 3   3       5
Day 4   3       5
This is a back slash symbol (\)
In every programming language it starts with "Hello, World!"
```

In [58]:
```
##string methods
challenge = 'thirty days of python'
print(challenge.capitalize())
```

```
Thirty days of python
```

In [60]:
```
#count()

challenge = 'thirty days of python'
print(challenge.count('y'))
print(challenge.count('y',7,14))
print(challenge.count('th'))
```

```
3
1
2
```

In [74]:
```
# endswitch()

challenge = 'thirty days of python'
print(challenge.endswith('on'))
print(challenge.endswith('tion'))
```

```
True
False
```

In [86]:
```
# expandtabs(): Replaces tab character with spaces, default tab size is 8. It ta

challenge = 'thirty\tdays\tof\tpython'
print(challenge.expandtabs())
print(challenge.expandtabs(10))
```

```
thirty  days    of      python
thirty    days      of        python
```

In [88]:
```
# find(): Returns the index of first occurrence of substring

challenge = 'thirty days of python'
print(challenge.find('y'))
print(challenge.find('th'))
```

```
5
0
```

In [90]:
```
# format()      formats string into nicer output
first_name = 'Anuja'
last_name = 'Pawar'
job = 'student'
country = 'India'
sentence = 'I am {} {}. I am a {}. I live in {}.'.format(first_name, last_name,
print(sentence)

radius = 10
pi = 3.14
area = pi # radius ## 2
result = 'The area of circle with {} is {}'.format(str(radius), str(area))
print(result) # The area of circle with 10 is 314.0
```

```
I am Anuja Pawar. I am a student. I live in India.
The area of circle with 10 is 3.14
```

In [92]:
```
# index(): Returns the index of substring
challenge = 'thirty days of python'
print(challenge.find('y'))
print(challenge.find('th'))
```

```
5
0
```

In [94]:
```
# isalnum(): Checks alphanumeric character

challenge = 'ThirtyDaysPython'
print(challenge.isalnum())

challenge = '30DaysPython'
```

```
print(challenge.isalnum())

challenge = 'thirty days of python'
print(challenge.isalnum())

challenge = 'thirty days of python 2019'
print(challenge.isalnum())
```

```
True
True
False
False
```

In [96]:

```
# isalpha(): Checks if all characters are alphabets

challenge = 'thirty days of python'
print(challenge.isalpha())
num = '123'
print(num.isalpha())
```

```
False
False
```

In [98]:

```
# isdecimal(): Checks Decimal Characters

challenge = 'thirty days of python'
print(challenge.find('y'))
print(challenge.find('th'))
```

```
5
0
```

In [102…

```
# isdigit(): Checks Digit Characters

challenge = 'Thirty'
print(challenge.isdigit())
challenge = '30'
print(challenge.isdigit())
```

```
False
True
```

In [104…

```
# isdecimal():Checks decimal character

num = '10'
print(num.isdecimal())
num = '10.5'
print(num.isdecimal())
```

```
True
False
```

In [106…

```
# isidentifier():Checks for valid identifier means it check if a string is a val

challenge = '30DaysOfPython'
print(challenge.isidentifier())
challenge = 'thirty_days_of_python'
print(challenge.isidentifier())
```

```
False
True
```

In [108…

```
# islower():Checks if all alphabets in a string are lowercase

challenge = 'thirty days of python'
print(challenge.islower())
challenge = 'Thirty days of python'
print(challenge.islower())
```

```
True
False
```

In [110…

```
# isupper(): returns if all characters are uppercase characters

challenge = 'thirty days of python'
print(challenge.isupper())
challenge = 'THIRTY DAYS OF PYTHON'
print(challenge.isupper())
```

```
False
True
```

In [112…

```
# isnumeric():Checks numeric characters

num = '10'
print(num.isnumeric())
print('ten'.isnumeric())
```

```
True
False
```

In [122…

```
# join(): Returns a concatenated string

web_tech = ['HTML', 'CSS', 'JavaScript', 'React']
result = '#, '.join(web_tech)
print(result)
```

```
HTML#, CSS#, JavaScript#, React
```

In [126…

```
# strip(): Removes both leading and trailing characters

challenge = ' thirty days of python '
print(challenge.strip('y'))
```

```
 thirty days of python
```

In [128…

```
# replace(): Replaces substring inside

challenge = 'thirty days of python'
print(challenge.replace('python', 'coding'))
```

```
thirty days of coding
```

In [130…

```
# split():Splits String from Left

challenge = 'thirty days of python'
print(challenge.split())
```

```
['thirty', 'days', 'of', 'python']
```

In [132…

```
# title(): Returns a Title Cased String

challenge = 'thirty days of python'
print(challenge.title())
```

```
Thirty Days Of Python
```

In [134…

```python
# swapcase(): Checks if String Starts with the Specified String

challenge = 'thirty days of python'
print(challenge.swapcase())
challenge = 'Thirty Days Of Python'
print(challenge.swapcase())
```

```
THIRTY DAYS OF PYTHON
tHIRTY dAYS oF pYTHON
```

In [136…

```python
# startswith(): Checks if String Starts with the Specified String

challenge = 'thirty days of python'
print(challenge.startswith('thirty'))
challenge = '30 days of python'
print(challenge.startswith('thirty'))
```

```
True
False
```

OPERATOR

In [2]:

```python
# arithmetic operator in python
# integer

print('Addition: ',1+2)
print('substraction: ',2-1)
print('multiplication: ' ,2*3)
print('Division:' ,4/2)
```

```
Addition:  3
substraction:  1
multiplication:  6
Division: 2.0
```

gives floating number

In [5]:

```python
print('Division: ', 6 / 2)
print('Division: ', 7 / 2)
print('Division without the remainder: ', 7 // 2)
```

```
Division:  3.0
Division:  3.5
Division without the remainder:  3
```

floating number or without the remaining

In [7]:

```python
print('Modulus: ', 3 % 2)
print ('Division without the remainder: ', 7 // 3)
print('Exponential: ', 3 ** 2)
```

```
Modulus:  1
Division without the remainder:  2
Exponential:  9
```

In [13]:

```python
# Floating numbers
print('Floating Number,PI=', 3.14)
print('Floating Number, gravity=', 9.81)
```

```
Floating Number,PI= 3.14
Floating Number, gravity= 9.81
```

In [15]:
```
# Complex numbers
print('Complex number: ', 1 + 1j)
print('Multiplying complex number: ',(1 + 1j) * (1-1j))
```

```
Complex number:  (1+1j)
Multiplying complex number:  (2+0j)
```

In [17]:
```
# Declaring the variable at the top first

a = 3
b = 2
```

In [21]:
```
# Arithmetic operations and assigning the result to a variable
total = a + b
diff = a - b
product = a * b
division = a / b
remainder = a % b
floor_division = a // b
exponential = a ** b
```

In [23]:
```
print(total)
print('a + b = ', total)
print('a - b = ', diff)
print('a * b = ', product)
print('a / b = ', division)
print('a % b = ', remainder)
print('a // b = ', floor_division)
print('a ** b = ', exponential)
```

```
5
a + b =  5
a - b =  1
a * b =  6
a / b =  1.5
a % b =  1
a // b =  1
a ** b =  9
```

In [25]:
```
# Declaring values and organizing them together
num_one = 3
num_two = 4
```

In [27]:
```
# Arithmetic operations
total = num_one + num_two
diff = num_two - num_one
product = num_one * num_two
div = num_two / num_two
remainder = num_two % num_one
```

In [29]:
```
# Printing values with label
print('total: ', total)
print('difference: ', diff)
print('product: ', product)
print('division: ', div)
print('remainder: ', remainder)
```

```
total:  7
difference:  1
product:  12
division:  1.0
remainder:  1
```

In [33]:
```
# Calculating area of a circle
radius = 10
area_of_circle = 3.14 * radius ** 2
print('Area of a circle:', area_of_circle)
```

```
Area of a circle: 314.0
```

In [35]:
```
# Calculating area of a rectangle
length = 10
width = 20
area_of_rectangle = length * width
print('Area of rectangle:', area_of_rectangle)
```

```
Area of rectangle: 200
```

In [37]:
```
# Calculating a weight of an object
mass = 75
gravity = 9.81
weight = mass * gravity
print(weight, 'N')
```

```
735.75 N
```

In [41]:
```
# logical operator

print(3 > 2)
print(3 >= 2)
print(3 < 2)
print(2 < 3)
print(2 <= 3)
print(3 == 2)
print(3 != 2)
print(len('mango') == len('avocado'))
print(len('mango') != len('avocado'))
print(len('mango') < len('avocado'))
print(len('milk') != len('meat'))
print(len('milk') == len('meat'))
print(len('tomato') == len('potato'))
print(len('python') > len('dragon'))
```

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

In [43]:
```
# Boolean comparison
print('True == True: ', True == True)
print('True == False: ', True == False)
print('False == False:', False == False)
print('True and True: ', True and True)
print('True or False:', True or False)
```

```
True == True:  True
True == False:  False
False == False: True
True and True:  True
True or False: True
```

In [47]:
```
# Another way comparison
print('1 is 1', 1 is 1)
print('1 is not 2', 1 is not 2)
print('A in Asabeneh', 'A' in 'Asabeneh')
print('B in Asabeneh', 'B' in 'Asabeneh')
print('coding' in 'coding for all')
print('a in an:', 'a' in 'an')
print('4 is 2 ** 2:', 4 is 2 ** 2)
```

```
1 is 1 True
1 is not 2 True
A in Asabeneh True
B in Asabeneh False
True
a in an: True
4 is 2 ** 2: True
```

```
<>:2: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?
<>:3: SyntaxWarning: "is not" with 'int' literal. Did you mean "!="?
<>:8: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?
<>:2: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?
<>:3: SyntaxWarning: "is not" with 'int' literal. Did you mean "!="?
<>:8: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?
C:\Users\ANUJA\AppData\Local\Temp\ipykernel_4692\4207187824.py:2: SyntaxWarning:
"is" with 'int' literal. Did you mean "=="?
  print('1 is 1', 1 is 1)
C:\Users\ANUJA\AppData\Local\Temp\ipykernel_4692\4207187824.py:3: SyntaxWarning:
"is not" with 'int' literal. Did you mean "!="?
  print('1 is not 2', 1 is not 2)
C:\Users\ANUJA\AppData\Local\Temp\ipykernel_4692\4207187824.py:8: SyntaxWarning:
"is" with 'int' literal. Did you mean "=="?
  print('4 is 2 ** 2:', 4 is 2 ** 2)
```

In [49]:
```
print(3 > 2 and 4 > 3)
print(3 > 2 and 4 < 3)
print(3 < 2 and 4 < 3)
print(3 > 2 or 4 > 3)
print(3 > 2 or 4 < 3)
print(3 < 2 or 4 < 3)
print(not 3 > 2)
print(not True)
print(not False)
print(not not True)
print(not not False)
```

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

```
In [ ]:
```