In [1]:
```
5 # without variable
```

Out[1]: 5

In [3]:
```
v = 5 #with variable
v
```

Out[3]: 5

In [5]:
```
5 = v
```

```
  Cell In[5], line 1
    5 = v
    ^
SyntaxError: cannot assign to literal here. Maybe you meant '==' instead of '='?
```

In [11]:
```
va = 34, 56
```

In [14]:
```
va
```

Out[14]: (34, 56)

In [18]:
```
va, var = 34, 56
```

In [21]:
```
print(va)
print(var)
```

```
34
56
```

In [23]:
```
import sys
sys.version
```

Out[23]: '3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)]'

In [25]:
```
A = 78
b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[25], line 2
      1 A = 78
----> 2 b

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

In [27]:
```
A
```

Out[27]: 78

In [29]:
```
nit = 21
NIT
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[29], line 2
      1 nit = 21
----> 2 NIT

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

In [31]:
```
nit
```

Out[31]: 21

In [33]:
```
1a = 67
1a
```

```
  Cell In[33], line 1
    1a = 67
    ^
SyntaxError: invalid decimal literal
```

In [35]:
```
a1 = 67
a1
```

Out[35]: 67

In [37]:
```
1@ = 89
```

```
  Cell In[37], line 1
    1@ = 89
       ^
SyntaxError: invalid syntax
```

In [39]:
```
x_train, x_test, y_train, y_test = 80, 20, 70,
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[39], line 1
----> 1 x_train, x_test, y_train, y_test = 80, 20, 70,

ValueError: not enough values to unpack (expected 4, got 3)
```

In [41]:
```
x_train, x_test, y_train, y_test = 80, 20, 70, 67, 100
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[41], line 1
----> 1 x_train, x_test, y_train, y_test = 80, 20, 70, 67, 100

ValueError: too many values to unpack (expected 4)
```

In [43]:
```
x_train, x_test, y_train, y_test = 80, 20, 70, 67

x_train
x_test
y_train
y_test
```

Out[43]: 67

In [45]:
```
x_train, x_test, y_train, y_test = 80, 20, 70, 67

print(x_train)
print(x_test)
print(y_train)
print(y_test)
```

```
80
20
70
67
```

In [47]:
```
if = 89
```

```
  Cell In[47], line 1
    if = 89
       ^
SyntaxError: invalid syntax
```

In [49]:
```
IF = 89
IF
```

Out[49]:
```
89
```

In [51]:
```
else = 90
```

```
  Cell In[51], line 1
    else = 90
    ^
SyntaxError: invalid syntax
```

In [53]:
```
eLSE = 78
eLSE
```

Out[53]:
```
78
```

In [55]:
```
import keyword
keyword.kwlist
```

Out[55]:
```
['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 [57]:
```
print(len(keyword.kwlist))
```

```
35
```

In [59]:
```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa = 90
aa
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[59], line 2
      1 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa = 90
----> 2 aa

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

In [61]:
```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Out[61]:
```
90
```

In [63]:
```
i = 2
i
```

Out[63]:
```
2
```

In [65]:
```
type(i)
```

Out[65]:
```
int
```

In [67]:
```
1i = 3
1i
```

```
  Cell In[67], line 1
    1i = 3
    ^
SyntaxError: invalid decimal literal
```

In [69]:
```
i1 = 3
i1
```

Out[69]:
```
3
```

# int completed

In [72]:
```
f = 110.5
f
```

Out[72]:
```
110.5
```

In [74]:
```
type(f)
```

Out[74]:
```
float
```

In [76]:
```
f1 = 1e0
f1
```

Out[76]:
```
1.0
```

In [78]:
```
f2 = 2e1
f2
```

Out[78]:
```
20.0
```

In [80]:
```
f3 = 3e2
f3
```

Out[80]:
```
300.0
```

# float completed

In [83]:
```
import keyword
keyword.kwlist
```

Out[83]:
```
['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 [85]:
```
b = TRUE
b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[85], line 1
----> 1 b = TRUE
      2 b

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

In [87]:
```
b = true
b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[87], line 1
----> 1 b = true
      2 b

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

In [89]:
```
b1 = True
b1
```

Out[89]: True

In [91]:
```
b2 = False
b2
```

Out[91]: False

In [93]:
```
type(b2)
```

Out[93]: bool

In [95]:
```
True + True + False - True
```

Out[95]: 1

In [97]:
```
True * False
```

Out[97]: 0

In [99]:
```
b1 + b2
```

Out[99]: 1

In [101…
```
b1 - b2
```

Out[101… 1

In [103…
```
b2 - b2
```

Out[103… 0

In [105…
```
b1 - b1
```

Out[105… 0

In [ ]:

In [ ]:

# Bool completed

In [108…
```
c = 10 + 20j
c
```

Out[108… (10+20j)

In [110…
```
type(c)
```

Out[110… complex

In [112…
```
c.real
```

Out[112… 10.0

In [114…
```
c.imag
```

Out[114… 20.0

In [116…
```
c
```

Out[116… (10+20j)

In [118…
```
c1 = 5 + 5j
c1
```

Out[118… (5+5j)

In [120…
```
c + c1
```

Out[120… (15+25j)

In [122…
```
c - c1
```

Out[122… (5+15j)

In [124…
```
c2 = 10 -2.5J
c2
```

Out[124… (10-2.5j)

In [126…
```
c3 = 2.0 + 34j
c3
```

Out[126… (2+34j)

# complex we complete

In [129…
```
s = 'generatiee ai & llm model great career ahead.also it is research domain.'
s
```

Out[129… 'generatiee ai & llm model great career ahead.also it is research domain.'

In [131…
```
s1= "generatiee ai & llm model great career ahead.also it is research domain."
s1
```

Out[131… 'generatiee ai & llm model great career ahead.also it is research domain.'

In [133…
```
s2= '''generatiee ai & llm model great career ahead.
            also it is research domain.'''
s2
```

Out[133… 'generatiee ai & llm model great career ahead.\n also it is research domain.'

# Basic string we completed

In [ ]:

# Type casting or Type conversion

In [139…
```
int(2.3) # float dataype to int
```

Out[139… 2

In [141…
```
int(2.7)
```

Out[141… 2

In [143…
```
int(2.3, 3.7)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[143], line 1
----> 1 int(2.3, 3.7)

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

In [145…
```
int(True)
```

Out[145… 1

In [147…
```
int('10')
```

Out[147… 10

In [149…
```
int('ten')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[149], line 1
----> 1 int('ten')

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

In [151…
```
int(1+2j)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[151], line 1
----> 1 int(1+2j)

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

In [153…
```
float(10)
```

Out[153… 10.0

In [155…
```
float(False)
```

Out[155… 0.0

In [157…
```
float('20')
```

Out[157…
```
20.0
```

In [159…
```
float('twenty')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[159], line 1
----> 1 float('twenty')

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

In [161…
```
float(10-20j)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[161], line 1
----> 1 float(10-20j)

TypeError: float() argument must be a string or a real number, not 'complex'
```

In [163…
```
print(bool(9))
print(bool(9.9))
print(bool('9'))
print(bool(9 + 9j))
print(bool(_))
print(bool())
```

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

In [165…
```
print(bool( ))
```

```
False
```

In [167…
```
print(bool(0))
```

```
False
```

# We can type cast from all other datatype to bool

In [170…
```
str(2)
```

Out[170…
```
'2'
```

In [172…
```
str(2.2)
```

Out[172…
```
'2.2'
```

In [174…
```
str(True)
```

Out[174… 'True'

In [176…
```
str(False)
```

Out[176… 'False'

In [178…
```
True + True
```

Out[178… 2

In [180…
```
str(10+20j)
```

Out[180… '(10+20j)'

In [182…
```
print(len('milk') != len('meato'))
```

```
True
```

In [186…
```
com = 'milk'
print(com[0])
print(com[1])
print(com[2])
print(com[3])
print(com[4])
```

```
m
i
l
k
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[186], line 6
      4 print(com[2])
      5 print(com[3])
----> 6 print(com[4])

IndexError: string index out of range
```

# List Datastructure

In [189…
```
i = 6.6
type(i)
```

Out[189… float

In [191…
```
l = []
l
```

Out[191… []

In [193…
```
type(l)
```

Out[193… list

In [195…
```
len()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[195], line 1
----> 1 len()

TypeError: len() takes exactly one argument (0 given)
```

In [197…
```
len(l)
```

Out[197…
```
0
```

In [201…
```
l.append(10)
```

In [203…
```
l
```

Out[203…
```
[10, 10]
```

In [205…
```
len(l)
```

Out[205…
```
2
```

In [207…
```
l.append(10,20,30,40)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[207], line 1
----> 1 l.append(10,20,30,40)

TypeError: list.append() takes exactly one argument (4 given)
```

In [209…
```
l.append(10)
l.append(20)
l.append(30)
l.append(40)
```

In [211…
```
l
```

Out[211…
```
[10, 10, 10, 20, 30, 40]
```

In [213…
```
l1 = []
```

In [215…
```
l1.append(70)
l1.append(2.3)
l1.append(True)
l1.append('1+2j')
l1.append([1,2,3])
```

In [217…
```
l1
```

Out[217…
```
[70, 2.3, True, '1+2j', [1, 2, 3]]
```

In [219…
```
print(l)
print(l1)
```

```
[10, 10, 10, 20, 30, 40]
[70, 2.3, True, '1+2j', [1, 2, 3]]
```

In [221…
```
print(id(l))
print(id(l1))
```

```
2198094726656
2198094660352
```

In [223…
```
print(len(l))
print(len(l1))
```

```
6
5
```

In [225…
```
l1
```

Out[225… [70, 2.3, True, '1+2j', [1, 2, 3]]

In [227…
```
l2 = l1.copy()
```

In [229…
```
l2
```

Out[229… [70, 2.3, True, '1+2j', [1, 2, 3]]

In [231…
```
l1 == l2
```

Out[231… True

In [233…
```
l
```

Out[233… [10, 10, 10, 20, 30, 40]

In [235…
```
l1 == l2
```

Out[235… True

In [237…
```
print(l1)
print(l2)
```

```
[70, 2.3, True, '1+2j', [1, 2, 3]]
[70, 2.3, True, '1+2j', [1, 2, 3]]
```

In [239…
```
print(id(l1)) == print(id(l2))
```

```
2198094660352
2198094740288
```

Out[239… True

In [241…
```
a = 5
b = 5
```

In [243…
```
print(id(a)) == print(id(b))
```

```
140717146319416
140717146319416
```

Out[243… True

In [245…
```
l
```

Out[245… [10, 10, 10, 20, 30, 40]

In [247…
```
l.remove(1000)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[247], line 1
----> 1 l.remove(1000)

ValueError: list.remove(x): x not in list
```

In [249…
```
l.remove(10)
```

In [251…
```
l
```

Out[251… [10, 10, 20, 30, 40]

In [253…
```
l.remove(10)
l
```

Out[253… [10, 20, 30, 40]

In [257…
```
l
```

Out[257… [10, 20, 30, 40]

# String indexing

In [261…
```
s7 = 'nareshit'
s7
```

Out[261… 'nareshit'

In [263…
```
s7[0]
```

Out[263… 'n'

In [265…
```
s7[1]
```

Out[265… 'a'

In [267…
```
s7[10]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[267], line 1
----> 1 s7[10]

IndexError: string index out of range
```

In [269…
```
s7
```

Out[269… 'nareshit'

In [271…
```
s7[-3]
```

Out[271… 'h'

In [273…
```
s7[-9]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[273], line 1
----> 1 s7[-9]

IndexError: string index out of range
```

In [275…
```
s7
```

Out[275… 'nareshit'

In [277…
```
for i in s7:
```

```
n
a
r
e
s
h
i
t
```

# Slicing

In [280…
```
s7
```

Out[280… 'nareshit'

In [282…
```
s8 = 'abcdefghi'
s8
```

Out[282… 'abcdefghi'

In [284…
```
s8[0:9]
```

Out[284… 'abcdefghi'

In [286…
```
s8[1:8]
```

Out[286… 'bcdefgh'

In [288…
```
s8
```

Out[288… 'abcdefghi'

In [290…
```
s8[1:-3]
```

Out[290… 'bcdef'

In [292…
```
s8
```

Out[292… 'abcdefghi'

In [294…
```
s8[1:-4]
```

Out[294… 'bcde'

In [296…
```
step_indexing = [1,2,3,4,5,6,7,8,9,10]
step_indexing
```

Out[296… [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

In [298…
```
step_indexing[0:10:4]
```

Out[298… [1, 5, 9]

In [300…
```
step_indexing[0:10:5]
```

Out[300… [1, 6]

In [302…
```
step_indexing
```

Out[302… [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

In [304…
```
step_indexing[:]
```

Out[304… [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

In [308…
```
l5 = l.reverse()
l5
```

In [310…
```
l
```

Out[310… [40, 30, 20, 10]

# Bit wise number system

In [313…
```
25
```

Out[313… 25

In [315…
```
bin(25)
```

Out[315… '0b11001'

In [317…
```
int(0b11001)
```

Out[317… 25

In [319…
```
bin(35)
```

Out[319… '0b100011'

In [321…
```
int(0b100011)
```

Out[321… 35

In [323…
```
oct(25)
```

Out[323… '0o31'

In [325…
```
int(0o31)
```

Out[325… 25

In [327…
```
bin(7)
```

Out[327… '0b111'

In [329…
```
hex(7)
```

Out[329… '0x7'

In [331…
```
0xa
```

Out[331… 10

In [333…
```
hex(256)
```

Out[333… '0x100'

In [335…
```
int(0x100)
```

Out[335… 256

In [337…
```
l
```

Out[337… [40, 30, 20, 10]

In [339…
```
l1
```

Out[339… [70, 2.3, True, '1+2j', [1, 2, 3]]

In [341…
```
l2
```

Out[341… [70, 2.3, True, '1+2j', [1, 2, 3]]

In [343…
```
l2.count(70)
```

Out[343… 1

In [345…
```
l2.append(70)
```

In [347…
```
l2
```

Out[347… [70, 2.3, True, '1+2j', [1, 2, 3], 70]

In [349…
```
l2.count(70)
```

Out[349… 2

In [351…
```
l2[:]
```
Out[351…
```
[70, 2.3, True, '1+2j', [1, 2, 3], 70]
```
In [353…
```
l2[:5]
```
Out[353…
```
[70, 2.3, True, '1+2j', [1, 2, 3]]
```
In [355…
```
l
```
Out[355…
```
[40, 30, 20, 10]
```
In [357…
```
l[5:]
```
Out[357…
```
[]
```
In [359…
```
l
```
Out[359…
```
[40, 30, 20, 10]
```
In [361…
```
l[:-1]
```
Out[361…
```
[40, 30, 20]
```
In [363…
```
l
```
Out[363…
```
[40, 30, 20, 10]
```
In [365…
```
l[::-1] # advnce slciing
```
Out[365…
```
[10, 20, 30, 40]
```
In [367…
```
l
```
Out[367…
```
[40, 30, 20, 10]
```
In [369…
```
l[::-2]
```
Out[369…
```
[10, 30]
```
In [371…
```
l
```
Out[371…
```
[40, 30, 20, 10]
```
In [373…
```
l.index(20)
```
Out[373…
```
2
```
In [375…
```
l2
```
Out[375…
```
[70, 2.3, True, '1+2j', [1, 2, 3], 70]
```
In [377…
```
id(l2)
```
Out[377…
```
2198094740288
```

In [380…
```
len(l2)
```
Out[380… 6

In [382…
```
l2.clear()
```

In [384…
```
l2
```
Out[384… []

In [386…
```
id(l2)
```
Out[386… 2198094740288

In [388…
```
del l2
```

In [390…
```
l1
```
Out[390… [70, 2.3, True, '1+2j', [1, 2, 3]]

In [392…
```
l1.pop()
```
Out[392… [1, 2, 3]

In [394…
```
l1
```
Out[394… [70, 2.3, True, '1+2j']

In [396…
```
l1
```
Out[396… [70, 2.3, True, '1+2j']

In [398…
```
l
```
Out[398… [40, 30, 20, 10]

In [404…
```
l2 = l.copy()
```

In [408…
```
l2
```
Out[408… [40, 30, 10]

In [416…
```
l2.pop(1)
```
Out[416… 30

In [418…
```
l2
```
Out[418… [40]

In [420…
```
l2.pop(-1)
l2
```
Out[420… []

In [422…
```
l2.insert(2, 25)
```

In [424…
```
l2
```

Out[424… [25]

In [426…
```
l
```

Out[426… [40, 30, 20, 10]

In [428…
```
l[0]
```

Out[428… 40

In [430…
```
l[0] = 400
```

In [432…
```
l
```

Out[432… [400, 30, 20, 10]

In [434…
```
c
```

Out[434… (10+20j)

In [436…
```
len(l1)
```

Out[436… 4

In [438…
```
l2
```

Out[438… [25]

In [440…
```
len(l2)
```

Out[440… 1

In [442…
```
l2.extend(l1)
```

In [444…
```
l2
```

Out[444… [25, 70, 2.3, True, '1+2j']

In [446…
```
len(l2)
```

Out[446… 5

In [448…
```
for i in l2:
    print(i)
```

```
25
70
2.3
True
1+2j
```

In [450…
```
for i in enumerate(l2):
    print(i)
```

```
(0, 25)
(1, 70)
(2, 2.3)
(3, True)
(4, '1+2j')
```

In [452…
```
l2
```

Out[452…
```
[25, 70, 2.3, True, '1+2j']
```

In [ ]:
```

```