In [3]:
```
spam = 1  # comment or note
text = "# This is not a comment because it's inside quotes"
```

In [ ]:
```

```

Python as calculator

In [4]:
```
2+2
```

Out[4]: 4

In [5]:
```
50-5*6/4
```

Out[5]: 42.5

In [6]:
```
(50-5*6)/4
```

Out[6]: 5.0

In [7]:
```
8/5
```

Out[7]: 1.6

In [ ]:
```

```

Float

In [8]:
```
17/3    # classic division returns a float
```

Out[8]: 5.666666666666667

In [9]:
```
17//3   # floor division discards the factional part
```

Out[9]: 5

In [10]:
```
17 % 3   # the % operator returns the remainder of the division
```

Out[10]: 2

In [11]:
```
5*3+2  # floored quotient * divisor + remaainder
```

Out[11]: 17

In [ ]:
```

```

Calculate powers

In [12]:
```
5**2  # 5 powwr of 2
```

Out[12]: 25

In [13]:
```
2**7  # 2 power of 7
```

Out[13]: 128

In [ ]:

In [14]:
```
width = 20
height = 5*9

area = width*height
area
```

Out[14]: 900

In [ ]:

In [15]:
```
n  # variables is not assigned any value
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[15], line 1
----> 1 n

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

In [ ]:

In [ ]:
```
4*3.73-1   # mixed operands give float
```

In [ ]:

In [ ]:
```
tax = 12.5/100
```

In [ ]:
```
price = 100.50
```

In [ ]:
```
price * tax
```

In [ ]:
```
_
```

In [ ]:
```
price + _
```

In [ ]:
```
round(_, 2)
```

In [ ]:

Text

In [ ]:
```
'Spam eggs'  #single quotes string
```

In [ ]:
```
"Paris rabbit goy your back:)! Yay!"  # double quote string with special characters
```

In [ ]:
```
'1975'    # digits and numrals enclosed in quotes are also strings
```

In [ ]:

Escaping quotes

In [ ]:
```
'doesn\'t'   # use \' to escape the single quote
```

In [ ]:
```
"doesn't"   # or we can use double quotes
```

In [ ]:
```
'"yes," they said'
```

In [ ]:
```
'\"yes,"they said'
```

In [ ]:
```
'"Isn\'t," they said'
```

In [ ]:

Escaping line

In [ ]:
```
s = 'First line.\nSecond line.'
s

# Without print function, \n is considered as a part of the string
```

In [ ]:
```
print(s)
```

In [ ]:

In [ ]:
```
print('C:\some\name')    # \ in string is followed by n, so it interprets as \n =>
```

In [ ]:
```
print(r'C:some\name')    # adding r before the quote solves this issue
```

In [ ]:

In [ ]:
```
# Multiple  line strings
```

In [ ]:
```
print('''Usage:thingy[OPTIONS]
-h
-H hostname''')

# can use '''  ''' quotes for multiline string
```

In [ ]:
```
print("""\
      Usage:thingy[OPTIONS]
      -h
      -H hostname
      """)

# or we can use  """\   """
```

Joining strings

In [ ]:
```
# 3 times 'un' followed by 'ium'

3*'un' + 'ium'
```

In [ ]:
```
# Multiple strings in quotes next to each other are automatically joined together

'p' 'y' 'thon'
```

In [ ]:
```
'put several strings within parentheses'  'to have them joined together'
```

In [ ]:
```
text = ('put several strings within parentheses '
        'to have them joined together.')
text
```

In [ ]:
```

```

In [ ]:
```
prefix = 'Py'
prefix'thon'

# joining wont work for a string and variable together
```

In [ ]:
```
# for variable and string we need to use + operator

prefix = 'py'
prefix + 'thon'
```

In [ ]:
```

```

In [ ]:
```
word = 'Python'
print(word[0])
print(word[5])
```

In [ ]:
```
word[-1]
```

In [ ]:
```
word[-3]
```

In [ ]:
```
word[0:2]  # slicing
```

In [ ]:
```
word[2:5]
```

In [ ]:
```
word[:2]
```

In [ ]:
```
word[2:]
```

In [ ]:
```
word[-2:]
```

In [ ]:
```
word[:2] + word[2:]
```

In [ ]:
```
word[:4] + word[4:]
```

In [ ]:
```

```

In [ ]:
```
word[42]  # 6 characters in word string
```

In [ ]:
```
word[4:42]
```

In [ ]:
```
word[42:]   # out of range slicing
```

In [ ]:
```

```

In [ ]:
```
word[0] = 'j'

# python strings cannot be changed , they are immutable
```

In [ ]:
```
# but they can be added partially or fully to make a new string

'j'+word[1:]
```

In [ ]:
```
word[3:5] + 'la'
```

In [ ]:
```

```

In [ ]:
```
s = 'supercalifrgilisticexpialidocius'   # length of string
```

In [ ]:
```
len(s)
```

In [ ]:
```

```

Lists

In [ ]:
```
squares = [1,4,9,16,25]
```

In [ ]:
```
squares
```

In [ ]:
```
squares[0]
```

In [ ]:
```
squares[-3:]
```

In [ ]:
```
squares[--3:]
```

In [ ]:
```
squares = squares + [36,49,64,80,100]
squares
```

In [ ]:
```
squares[8] = 81
squares
```

In [ ]:
```
squares.append(121)
```

In [ ]:
```
squares
```

In [ ]:
```

```

variable copy

In [ ]:
```
rgb = ["red","green","blue"]
rgb
```

In [ ]:
```
rgby = rgb
rgby
```

In [ ]:
```
id(rgb) == id(rgby)
```

In [ ]:
```
rgby.append('yellow')
```

In [ ]:
```
rgby
```

In [ ]:
```

```

In [ ]:
```
correct_rgby = rgby[:]
```

In [ ]:
```
correct_rgby[-1] = 'yelow'
```

In [ ]:
```
correct_rgby
```

In [ ]:
```

```

assignment to slices

In [ ]:
```
letters = ['a','b','c','d','e','f','g']
```

In [ ]:
```
letters
```

In [ ]:
```
letters[2:5]
```

In [ ]:
```
letters[2:5] = ['C','D','E']   #replace items using slicing
```

In [ ]:
```
letters
```

In [ ]:
```
letters[2:5] = []    # removing items using slicing
```

In [ ]:
```
letters
```

In [ ]:
```
letters[:] = []   # claering all items
```

In [ ]:
```
letters
```

In [ ]:
```

```

In [ ]:
```
letters = ['a','b','c','d']
```

In [ ]:
```
letters
```

In [ ]:
```
len(letters)
```

In [ ]:
```

```

nestlists

In [ ]:
```
a = ['a','b','c']
n = [1,2,3]
x = [a,n]
```

In [ ]:
```
x
```

In [ ]:
```
x[0]
```

In [ ]:
```
x[1][2]
```

In [ ]:
```

```

fibonacci series code

In [22]:
```
# fibonacci series - the sum of two preceding elements defines the next element
```

```
a, b = 0, 1
while a<1000:
    print(a)
    a,b = b,a+b
```

```
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
```

In [ ]:

In [20]:
```
i = 256*256
print('The value of i is:',i)
```

```
The value of i is: 65536
```

In [ ]:

In [21]:
```
a,b = 0,1
while a<1000:
    print(a, end=',')
    a,b = b,a+b
```

```
0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,
```

In [ ]:

In [23]:
```
-3**2     # ** has higher precedence than - : BODMAS rule
```

Out[23]: -9

In [25]:
```
(-3)**2   # to avoid wrong calculation
```

Out[25]: 9

In [ ]:

In [45]:
```
'Nit\nAmeerpet'
```

Out[45]: 'Nit\nAmeerpet'

In [41]:
```
print('Nit\nAmeerpet')
```

```
Nit
Ameerpet
```

In [ ]:
```

```