List comprehensions

Create a list of square numbers between 1 to 9

In [1]:
```
number = 3  # variable
number *  number   # expresiion computation
```

Out[1]:
```
9
```

In [3]:
```
# python list data structure
# syntax [expression(value) loops  conditions]
square_list = [number * number for number in range(1,10,2)]
square_list
```

Out[3]:
```
[1, 9, 25, 49, 81]
```

In [ ]:
```
# loops
# repeatation
# range function in-build
# range(start, end+1 , step)
# 1,3,5, 7,9
```

In [4]:
```
# # data type
# integer  ---> 19
# string ----> 'mehul wankhede'
# float ---> 10.333
# boolean ----> True / False 0/1

# variable ----> # container which store a specific datatype value
```

In [5]:
```
number = 3
type(number)
```

Out[5]:
```
int
```

In [6]:
```
name = 'mehul wankhede'
type(name)
```

Out[6]:
```
str
```

In [7]:
```
marks = 76.88
type(marks)
```

Out[7]:
```
float
```

In [8]:
```
is_adult = True
type(is_adult)
```

Out[8]:
```
bool
```

write a list comprehension to list the stirng lengths

In [9]:
```
input_list = ['mehul','nilesh','rahul','khushi','Faizan']
# length of string  number of charectors in the string element
```

```
#[ 5, 6, 5, 6, 6]

# syntax [expression(value) loops  conditions]
output_list = [len(string_value) for string_value in input_list]
output_list
```

Out[9]:
```
[5, 6, 5, 6, 6]
```

In [10]:
```
for string_value in input_list:
    print(string_value)
```

```
mehul
nilesh
rahul
khushi
Faizan
```

Generate square number from 0 to 10 number should be divisible by 3

In [ ]:
```
```

In [19]:
```
# [1,2,3,4,5,6,7,8,9,10]

# [3, 6, 9]
# [9, 36, 81]
# syntax [expression(value) loops  conditions]
square_numbers = [number * number for number in range(0,11) if number%3 == 0]
square_numbers
```

Out[19]:
```
[0, 9, 36, 81]
```

Generate cube number from 0 to 10 number should be divisible by 2

In [39]:
```
# syntax [expression(value) loops  conditions]
#  number * number * number
#

cube_list = [number * number * number for number in range(0,11) if number%2 == 0]
cube_list
```

Out[39]:
```
[0, 8, 64, 216, 512, 1000]
```

In [ ]:
```
```

In [ ]:
```
```

In [ ]:
```
```

Generate square number from 0 to 10 number should be divisible by 5

In [29]:
```
# syntax [expression(value) loops  conditions]
square_numbers = [number * number for number in range(0,11) if number%5 == 0]
square_numbers
```

Out[29]:
```
[0, 25, 100]
```

In [37]:
```
5 % 5 == 0
```

Out[37]:
```
True
```

In [ ]:
```

```

In [28]:
```
#operator
21 % 3  # division modulus
23 + 2  # plus
23 -2   # minus
23 // 2  # division
# 2 ** 3  2 raise 3
2 * 3 # multiplication
```

Out[28]:
```
0
```

In [16]:
```
# syntax [expression(value) loops  conditions]
# if -else

number = 5
# coditional operators
# > greater than
# < less than
# == equals to
# != not equals to

if number > 0:# True
    print(f'number {number} is positive')
else:# False
    print(f'number {number} is negative')
```

```
number 5 is positive
```

In [17]:
```
number > 0
```

Out[17]:
```
True
```

In [18]:
```
number
```

Out[18]:
```
5
```

write a list comprehension to list the stirng lengths add only which has atleast 4 charectors

In [38]:
```
input_list = ['mehul','nilesh','rah','khhi','Faizan', 'mehul wankhede', 'mi']

# syntax [expression(value) loops  conditions
output_list = [len(string_value) for string_value in input_list if len(string_value) >=
output_list
```

Out[38]:
```
[5, 6, 4, 6, 14]
```

Functions

In [40]:
```
# block of code which performs specific task
```

```
# group of statesments
# in-build ---> defined by language itself
# user defined ---> created by developer or user
# anonymised functions
```

Anonymised functions

- a functions which does not have any name to it ### Lambda Functions

In [42]:
```
multiple_of_two = lambda x: x*2
```

In [43]:
```
multiple_of_two(8)
```

Out[43]:
```
16
```

In [44]:
```
multiple_of_two(3)
```

Out[44]:
```
6
```

In [45]:
```
# lambda function which square the given number
square_number_lambda = lambda x: x*x
```

In [46]:
```
# syntax [expression(value) loops  conditions]
square_numbers = [square_number_lambda(number) for number in range(0,11)]
square_numbers
```

Out[46]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [47]:
```
cube_number_lambda = lambda x: x*x*x
```

In [48]:
```
cube_numbers = [cube_number_lambda(number) for number in range(0,11)]
cube_numbers
```

Out[48]:
```
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

In [49]:
```
add_two_number = lambda x,y: x+y
```

In [50]:
```
add_two_number(10,20)
```

Out[50]:
```
30
```

In [53]:
```
two_number_mul = lambda x,y: x*y if x > 10 else x+y
```

In [55]:
```
two_number_mul(12,3)
```

Out[55]:
```
36
```

In [64]:
```
two_number_mul = lambda x: x*x if x %3 ==0 else ''
```

In [65]:
```
two_number_mul(9)
```

Out[65]:
```
81
```

In [ ]:
```

```