List Comprehension

> A Python list comprehension consists of brackets containing the expression, which is executed for each element along with the for loop to iterate over each element in the Python list.

In [30]:

```
n = [20,21,22]
n2 = []
for i in n:
    #print(i*2)
    n2.append(i*2)
print(n2)
```

```
[40, 42, 44]
```

In [31]:

```
n = [20,21,22]
double = [i*2 for i in n]
print(double)
```

```
[40, 42, 44]
```

In [32]:

```
numbers = [1, 2, 3, 4, 5]
square = [i**2 for i in numbers]
print(square)
```

```
[1, 4, 9, 16, 25]
```

In [32]:

```
# Print all even numbers from 0 to 10 using List Comprehension
lst = [i for i in range(11) if i%2==0]
print(lst)
```

```
[0, 2, 4, 6, 8, 10]
```

In [33]:
```python
"""
[
    [0,1,2],
    [0,1,2],
    [0,1,2],
]
"""


matrix = [[j for j in range(3)] for i in range(3)]
print(matrix)
```

```
[[0, 1, 2], [0, 1, 2], [0, 1, 2]]
```

In [34]:
```python
import pprint
pp = pprint.PrettyPrinter(width=20)
pp.pprint(matrix)
```

```
[[0, 1, 2],
 [0, 1, 2],
 [0, 1, 2]]
```

List Comprehensions vs For Loop

In [35]:
```python
# Using Loop
lst = []
for char in 'TIU is the best University':
    lst.append(char)
print(lst)
```

```
['T', 'I', 'U', ' ', 'i', 's', ' ', 't', 'h', 'e', ' ', 'b', 'e', 's', 't', ' ', 'U', 'n', 'i', 'v', 'e', 'r', 's',
'i', 't', 'y']
```

In [36]:
```python
# Using List Compehension
lst = [char for char in 'TIU is the best University']
print(lst)
```

```
['T', 'I', 'U', ' ', 'i', 's', ' ', 't', 'h', 'e', ' ', 'b', 'e', 's', 't', ' ', 'U', 'n', 'i', 'v', 'e', 'r', 's',
'i', 't', 'y']
```

In [37]:
```python
# Using Lambda Function
# [1,2,3,4,5] ---> [10,20,30,40,50]
number = list(map(lambda x:x*10,  [x for x in range(1,6) ]))
print(number)
```

```
[10, 20, 30, 40, 50]
```

Python Exception Handling

In [33]:
```python
num1 = int(input("Please enter the first number: "))
num2 = int(input("Please enter the second number: "))
quotient = num1 / num2
print (f"So {num1} / {num2} = {quotient}")
print ("End of the program...")
```

```
Please enter the first number: 26
Please enter the second number: 13
So 26 / 13 = 2.0
End of the program...
```

In [34]:
```python
num1 = int(input("Please enter the first number: "))
num2 = int(input("Please enter the second number: "))
quotient = num1 / num2
print (f"So {num1} / {num2} = {quotient}")
print ("End of the program...")
```

```
Please enter the first number: 19
Please enter the second number: 0

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_14524\3362782824.py in <module>
      1 num1 = int(input("Please enter the first number: "))
      2 num2 = int(input("Please enter the second number: "))
----> 3 quotient = num1 / num2
      4 print (f"So {num1} / {num2} = {quotient}")
      5 print ("End of the program...")

ZeroDivisionError: division by zero
```

In [38]:

```python
try:
    num1 = int(input("Please enter the first number: "))
    num2 = int(input("Please enter the second number: "))
    quotient = num1 / num2
    print (f"So {num1} / {num2} = {quotient}")

except ZeroDivisionError as zde:
    print ("ZeroDivisionError: Division by ZERO is Illegal...!!!")
    print ("ZeroDivisionError: So the error type is", type(zde))
    print ("ZeroDivisionError: So the error message is", zde)

except ValueError as ve:
    print ("ValueError: Invalid input has been provided...!!!")
    print ("ValueError: So the error type is", type(ve))
    print ("ValueError: So the error message is", ve)
print ("End of the program...")
```

```
Please enter the first number: 5
Please enter the second number: 2
So 5 / 2 = 2.5
End of the program...
```

In [40]:

```python
try:
    num1 = int(input("Please enter the first number: "))
    num2 = int(input("Please enter the second number: "))
    quotient = num1 / num2
    print (f"So {num1} / {num2} = {quotient}")
except Exception as ex:
    print ("Exception: Some other exception has occurred...!!!")
    print ("Exception: So the error type is", type(ex))
    print ("Exception: So the error message is", ex)
print ("End of the program...")
```

```
Please enter the first number: 2
Please enter the second number: one
Exception: Some other exception has occurred...!!!
Exception: So the error type is <class 'ValueError'>
Exception: So the error message is invalid literal for int() with base 10: 'one'
End of the program...
```

In [ ]:

```python
try:
    num1 = int(input("Please enter the first number: "))
    num2 = int(input("Please enter the second number: "))
    quotient = num1 / num2
    print (f"So {num1} / {num2} = {quotient}")
# except ZeroDivisionError as zde:
#     print ("ZeroDivisionError: Division by ZERO is Illegal...!!!")
#     print ("ZeroDivisionError: So the error type is", type(zde))
#     print ("ZeroDivisionError: So the error message is", zde)
except ValueError as ve:
    print ("ValueError: Invalid input has been provided...!!!")
    print ("ValueError: So the error type is", type(ve))
    print ("ValueError: So the error message is", ve)

except Exception as ex:
    print ("Exception: Some other exception has occurred...!!!")
    print ("Exception: So the error type is", type(ex))
    print ("Exception: So the error message is", ex)
print ("End of the program...")
```

In [42]:

```python
try:
    num1 = int(input("Please enter the first number: "))
    num2 = int(input("Please enter the second number: "))
    quotient = num1 / num2
    print (f"So {num1} / {num2} = {quotient}")
except ValueError as ve:
    print ("ValueError: Invalid input has been provided...!!!")
    print ("ValueError: So the error type is", type(ve))
    print ("ValueError: So the error message is", ve)
except ZeroDivisionError as zde:
    print ("ZeroDivisionError: Division by ZERO is Illegal...!!!")
    print ("ZeroDivisionError: So the error type is", type(zde))
    print ("ZeroDivisionError: So the error message is", zde)
except Exception as ex:
    print ("Exception: Some other exception has occurred...!!!")
    print ("Exception: So the error type is", type(ex))
    print ("Exception: So the error message is", ex)
else:
    print ("Else: This is the Else block executing...")
    print ("Else: Had a smooth execution...")
print ("End of the program...")
```

```
Please enter the first number: 10
Please enter the second number: 0
ZeroDivisionError: Division by ZERO is Illegal...!!!
ZeroDivisionError: So the error type is <class 'ZeroDivisionError'>
ZeroDivisionError: So the error message is division by zero
End of the program...
```

In [44]:

```python
try:
    num1 = int(input("Please enter the first number: "))
    num2 = int(input("Please enter the second number: "))
    quotient = num1 / num2
    print (f"So {num1} / {num2} = {quotient}")
except ValueError as ve:
    print ("ValueError: Invalid input has been provided...!!!")
    print ("ValueError: So the error type is", type(ve))
    print ("ValueError: So the error message is", ve)
except ZeroDivisionError as zde:
    print ("ZeroDivisionError: Division by ZERO is Illegal...!!!")
    print ("ZeroDivisionError: So the error type is", type(zde))
    print ("ZeroDivisionError: So the error message is", zde)
except Exception as ex:
    print ("Exception: Some other exception has occurred...!!!")
    print ("Exception: So the error type is", type(ex))
    print ("Exception: So the error message is", ex)
else:
    print ("Else: This is the Else block executing...")
    print ("Else: Had a smooth execution...")
finally:
    print ("Finally: This is Finally block executing...")
    print ("Finally: This block executes always...")
print ("End of the program...")
```

```
Please enter the first number: 10
Please enter the second number: 5
So 10 / 5 = 2.0
Else: This is the Else block executing...
Else: Had a smooth execution...
Finally: This is Finally block executing...
Finally: This block executes always...
End of the program...
```

In [46]:

```python
try:
    num1 = int(input("Please enter the first number within range (-100 to +100): "))
    num2 = int(input("Please enter the second number within range (-100 to +100): "))
    if (num1 < -100 or num2 < -100):
        raise NameError("Below-100")
    if (num1 > 100 or num2 > 100):
        raise NameError("Above100")
    quotient = num1 / num2
    print (f"So {num1} / {num2} = {quotient}")

except ValueError as ve:
    print ("ValueError: Invalid input has been provided...!!!")
    print ("ValueError: So the error type is", type(ve))
    print ("ValueError: So the error message is", ve)

except ZeroDivisionError as zde:
    print ("ZeroDivisionError: Division by ZERO is Illegal...!!!")
    print ("ZeroDivisionError: So the error type is", type(zde))
    print ("ZeroDivisionError: So the error message is", zde)

except NameError as ne:
    print ("NameError: Input value is out of range...")
    if (str(ne) == "Below-100"):
        print ("NameError: Input value is LESS THAN -100...")
    elif (str(ne) == "Above100"):
        print ("NameError: Input value is GREATER THAN 100...")

except Exception as ex:
    print ("Exception: Some other exception has occurred...!!!")
    print ("Exception: So the error type is", type(ex))
    print ("Exception: So the error message is", ex)

else:
    print ("Else: This is the Else block executing...")
    print ("Else: Had a smooth execution...")
finally:
    print ("Finally: This is Finally block executing...")
    print ("Finally: This block executes always...")
print ("End of the program...")
```

```
Please enter the first number within range (-100 to +100): 7
Please enter the second number within range (-100 to +100): 2
So 7 / 2 = 3.5
Else: This is the Else block executing...
Else: Had a smooth execution...
Finally: This is Finally block executing...
Finally: This block executes always...
End of the program...
```

Minor Project

> Email Id Validator (Is it a Valid Email or Not)

In [18]:

```python
# t@g.co ----> 6
# 2@g.co ---Restricted

email = input("Enter Your Email : ")
k,j,d=0,0,0
if len(email)>=6:
    if email[0].isalpha():
        if ("@" in email) and (email.count("@")==1):
            if (email[-4]==".") ^ (email[-3]=="."):
                for i in email:
                    if i==i.isspace():
                        k=1
                    elif i.isalpha():
                        if i==i.upper():
                            j=1
                    elif i.isdigit():
                        continue
                    elif i=="_" or i=="." or i=="@":
                        continue
                    else:
                        d=1
                if k==1 or j==1 or d==1:
                    print("Wrong Email.....")
                else:
                    print("Valid Email...")

            else:
                print("Wrong Email....")
        else:
            print("Wrong Email....")

    else:
        print("Wrong Email....")

else:
    print("Wrong Email....")
```

```
Enter Your Email : surabhi.mondal22234@gmail.com
Valid Email...
```

Regular Expression

In [19]:

```
# Rules
# a-z
#0-9
# . _ 1 time
# @   1 time
# After . 2,3 alpha req

import re
email = "^[a-z]+[\._]?[a-z0-9]+[@]\w+[.]\w{2,3}$"
user_email = input("Enter your Email: ")

if re.search(email,user_email):
    print("Valid Email...")
else:
    print("Invalid Email...")
```

```
Enter your Email: anwesha.b018@gmail.com
Valid Email...
```

Object Oriented Programming in Python

Procedural languages lack in encapsulation, difficult to manage when code size is >= 10 KLOC. Here variables are unprotected, no automatic memory management by deleting dereferenced variables.

In [20]:
```python
class MyFirstClass:
    pass
ob1 = MyFirstClass()
print(ob1,type(ob1))
```

```
<__main__.MyFirstClass object at 0x0000023088CFBB80> <class '__main__.MyFirstClass'>
```

In [23]:
```python
class MyFirstClass:
    '''This is a document string.
    This is a multi-line text...'''
ob1 = MyFirstClass()
print(ob1.__doc__)
print(MyFirstClass.__doc__)
```

```
This is a document string.
    This is a multi-line text...
This is a document string.
    This is a multi-line text...
```

> “self” is used to access and manipulate the instance variables and methods within a class. Without “self,” it would be impossible to differentiate between instance variables and class variables or methods.

In [27]:
```python
class MyFirstClass:
    """This is a document string..."""
    class_var1 = 100   # class or static variable
    class_var2 = 200
    def __init__(self,data1, data2): # Constructor method  positional parameters
        print("Executing the constructor method...")   # self is called an object binding variable
        print("self:", self, type(self))
        self.inst_var1 = data1   # instance variable
        self.inst_var2 = data2

    def display(self):
        print("Executing the display method...")
        print(f"Class variable values are {MyFirstClass.class_var1} and {MyFirstClass.class_var2}...")
        print(f"Class varibale values are {self.class_var1} and {self.class_var2}... ")
        print(f"Instane variable values are {self.inst_var1} and {self.inst_var2}...")

ob1 = MyFirstClass(111,222) # positional arguments
print(ob1.__doc__)
print(MyFirstClass.__doc__)
ob1.display()
```

```
Executing the constructor method...
self: <__main__.MyFirstClass object at 0x0000023088CF8E20> <class '__main__.MyFirstClass'>
This is a document string...
This is a document string...
Executing the display method...
Class variable values are 100 and 200...
Class varibale values are 100 and 200...
Instane variable values afe 111 and 222...
```

In [ ]:

In [ ]: