In [3]:
```
# functions
def get_sum(lst):
    """
    This function returns the sum of all elements of a list
    """
    all = 0
    for num in lst:
        all += num
        print(all)
    return all

sum1 = get_sum([1,2,3,4])
print(sum1)
print(get_sum.__doc__)
```

```
1
3
6
10
10

    This function returns the sum of all elements of a list
    
```

In [ ]:
```
# assignment - 1
# why we can't call the local variable outside the function
```

Assessment

In [15]:
```
# hcf of two numbwrs function
def hcf(a,b):
    while b:
        a,b = b,a%b
    return a

print(hcf(4,2))
```

```
2
```

In [21]:
```
# inbuilt functions
# abs()
print(abs(-100))
# all()
print(all([0]))
# divmod
print(divmod(13,2))
# enumerate
for idx,num in enumerate(range(10),start = 1):
    print("index - {0} has value {1}".format(idx,num))
```

```
100
False
(6, 1)
index - 1 has value 0
index - 2 has value 1
index - 3 has value 2
index - 4 has value 3
index - 5 has value 4
index - 6 has value 5
index - 7 has value 6
index - 8 has value 7
index - 9 has value 8
index - 10 has value 9
```

In [30]:
```python
# filter function
# first argumment is function to apply and second argument is iterable
# this is an example of higher order function
# function tacking other function as argument
def check_positive(num):
    return not (num * -1 > 0)


lst = [10,-20,30,40,-50]
positive_numbers = list(filter(check_positive,lst))
print(positive_numbers)

# using lambda function
positive_no = list(filter(lambda x:x>=0,lst))
print(positive_no)
```

```
[10, 30, 40]
[10, 30, 40]
```

In [32]:
```python
# isinstance()
# it is used to check whether the object is of a particular class
# first argument value then class name

print(isinstance([],tuple))
```

```
False
```

In [34]:
```python
# map function
# apply a function to all elements in the iterable
lst = [2,3,4,5,6,7]
# using lambda function
square_lst = list(map(lambda x:x**2,lst))
print(square_lst)
```

```
[4, 9, 16, 25, 36, 49]
```

In [37]:
```python
from functools import reduce
# reduce function
# it is used to apply any computation to all values in an iterable
# return a single value

#  sum of all elements using reduce
lst = [1,2,3,4,5]
# it work like accumulator
sum_elements = reduce(lambda x,y:x+y,lst)
print(sum_elements)
```

```
product_elements = reduce(lambda x,y:x*y,lst)
print(product_elements)
```

```
15
120
```

In [ ]:
```
# assignment 2
# where to use map(),filter() and reduce()
```

In [44]:
```
# function argumnets
# default arguments
def say(x,y = 1):
    print("x = {x} and y = {y}".format(x = x,y = y))
say(10,20)

def sum_numbers(x = 0 ,y = 0,z = 0):
    print(x+y+z)

sum_numbers(10,20,30)
sum_numbers(20,30)
sum_numbers(10,20)
sum_numbers(10)
sum_numbers()
```

```
x = 10 and y = 20
60
50
30
10
0
```

In [50]:
```
# keyword arguments
# **kwargs the parameter passed as name assigned

def user_details(**kwargs):
    print("name = {name},age = {age},address = {add}".format(name = kwargs['name']

user_details(name = "adharsh",age = "23",address = "kerala")
```

```
name = adharsh,age = 23,address = kerala
```

In [53]:
```
# arbitrary keyword
def details(*names):
    for name in names:
        print(name)
details('adharsh','23','btech')
```

```
adharsh
23
btech
```

In [ ]:
```
# class,module,packages
# creatind class
class Student:
    pass
```

In [1]:
```
# creating object and update value
class Hello:
    hi = "greeting"
```

```
test1 = Hello()
test2 = Hello()
test1.hi = "hi"
test2.hi = "hello"
print(Hello.hi,test1.hi,test2.hi)
```

```
greeting hi hello
```

In [8]:
```
# __init__() function
class Member:
    def __init__(self,name,age,position = 'developer'):
        self.name = name
        self.age = age
        self.position = position

    def hi(self,age,name):
        self.name = name
        self.age = age
    def test_fun(self):
        print("hello {0} how are you?".format(self.name))

test1 = Member('adharsh',23,'trainee')
print(test1.name)
print(test1.age)
print(test1.position)
test2 = Member('hi',20)
print(test2.name)
print(test2.position)
test1.test_fun()
```

```
adharsh
23
trainee
hi
developer
hello adharsh how are you?
```

In [10]:
```
# self usage we can use any thing but must use first position
class Person:
    def __init__(obj,name,age):
        obj.name = name
        obj.age = age
    def fun(obj):
        print(f"hello {obj.name}")

obj1 = Person("xyz",23)
obj1.fun()
```

```
hello xyz
```

In [11]:
```
# inheritance

class xyz:
    print("xyz is parent class")
class abc(xyz):
    print("child of xyz")
```

```
xyz is parent class
child of xyz
```

In [16]:
```
class Sports:
    def __init__(self,model,color,cost):
        self.__model = model
        self.__cost = cost
        self.__color = color

    def get_model(self):
        return self.__model
    def set_model(self,model):
        self.__model = model
    def get_cost(self):
        return self.__cost
    def set_cost(self,cost):
        self.__cost = cost
    def get_color(self):
        return self.__color
    def set_color(self,color):
        self.__color = color

class Shoes(Sports):
    def __init__(self,model,color,cost,brand):
        super().__init__(model,color,cost)
        self.__brand = brand
    def get_brand(self):
        return self.__brand
    def shoes_info(self):
        print(f"{self.get_model()} {self.get_color()} {self.get_cost()}")


sh = Shoes("Shoes","Black","2000","xyz")
print(sh.get_color())
print(sh.get_model())
print(sh.get_cost())
print(sh.get_brand())
sh.shoes_info()
```

```
Black
Shoes
2000
xyz
Shoes Black 2000
```

In [ ]:
```
# exception handling
# two type of errors runtime error and compile time error
# runtime errors can be handled through exception handling
# compile time errors are mainly syntax error
```

In [1]:
```
1/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[1], line 1
----> 1 1/0

ZeroDivisionError: division by zero
```

In [2]:
```
open("tst.txt")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[2], line 1
----> 1 open("tst.txt")

File C:\Program Files\Python312\Lib\site-packages\IPython\core\interactiveshell.py:
324, in _modified_open(file, *args, **kwargs)
    317 if file in {0, 1, 2}:
    318     raise ValueError(
    319         f"IPython won't let you open fd={file} by default "
    320         "as it is likely to crash IPython. If you know what you are doing,
"
    321         "you can use builtins' open."
    322     )
--> 324 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'tst.txt'
```

In [6]:
```
# python builtin exceptions
print(dir(__builtins__))
```

```
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BaseExcep
tionGroup', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'C
hildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedE
rror', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'Encod
ingWarning', 'EnvironmentError', 'Exception', 'ExceptionGroup', 'False', 'FileExist
sError', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExi
t', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'I
nterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupErro
r', 'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryErro
r', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDe
precationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionError', 'Refe
renceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIterati
on', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit',
'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecod
eError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWar
ning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError',
'__IPYTHON__', '__build_class__', '__debug__', '__doc__', '__import__', '__loader_
_', '__name__', '__package__', '__spec__', 'abs', 'aiter', 'all', 'anext', 'any',
'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'cla
ssmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir',
'display', 'divmod', 'enumerate', 'eval', 'exec', 'execfile', 'filter', 'float', 'f
ormat', 'frozenset', 'get_ipython', 'getattr', 'globals', 'hasattr', 'hash', 'hel
p', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'licens
e', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct',
'open', 'ord', 'pow', 'print', 'property', 'range', 'repr', 'reversed', 'round', 'r
unfile', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'supe
r', 'tuple', 'type', 'vars', 'zip']
```

In [7]:
```
# catching the exceptions in python
# sys.exc_info() will give the exception details
import sys
lst = ['b',2,0]

for entry in lst:
    try:
        print("value is {}".format(entry))
        k = 1 / int(entry)
        print("reciprocal of {0} is {1}".format(entry,k))
    except:
```

```
        print("error occured",sys.exc_info())
        print("-------------------------------------------------------------------------
        print("next item")
```

```
value is b
error occured (<class 'ValueError'>, ValueError("invalid literal for int() with bas
e 10: 'b'"), <traceback object at 0x0000023693B9B240>)
--------------------------------------------------------------------------------
---------------------------
next item
value is 2
reciprocal of 2 is 0.5
value is 0
error occured (<class 'ZeroDivisionError'>, ZeroDivisionError('division by zero'),
<traceback object at 0x0000023693B9B180>)
--------------------------------------------------------------------------------
---------------------------
next item
```

In [14]:
```
import sys
def div_number(a,b):
    try:
        return a/b
    except :
        print(sys.exc_info())

print(sum_number(2,3))
print(sum_number(1,0))
```

```
0.6666666666666666
(<class 'ZeroDivisionError'>, ZeroDivisionError('division by zero'), <traceback obj
ect at 0x0000023693AB6D40>)
None
```

In [16]:
```
class LengthException(Exception):
    pass

try:
    name = input("enter user name")
    if len(name)<8:
        raise LengthException("the user name must be atleast 8 characters")
except Exception as e:
    print(e)
```

```
the user name must be atleast 8 characters
```

In [17]:
```
# catching a specific exception
lst = ['b',2,0]

for entry in lst:
    try:
        print("value is {}".format(entry))
        k = 1 / int(entry)
        print("reciprocal of {0} is {1}".format(entry,k))
    except ValueError:
        print("Value Error")
    except ZeroDivisionError:
        print("ZeroDivivisonError")
    except:
```

```
        print("other error")
```

```
value is b
Value Error
value is 2
reciprocal of 2 is 0.5
value is 0
ZeroDivivisonError
```

In [20]:
```
# Raisisng exception
try:
    num = int(input("enter the number"))
    if num <= 0:
        raise ValueError("ERROR : Entered negative Number")
except ValueError as e:
    print(e)
```

```
ERROR : Entered negative Number
```

In [21]:
```
try:
    num = int(input("enter the number"))
    if num <= 0:
        raise MemoryError("ERROR : RAM Storage can't be negative")
except MemoryError as e:
    print(e)
```

```
ERROR : RAM Storage can't be negative
```

In [23]:
```
raise KeyboardInterrupt("you pressed a key")
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[23], line 1
----> 1 raise KeyboardInterrupt("you pressed a key")

KeyboardInterrupt: you pressed a key
```

In [30]:
```
# finally
# execute every time no matter code crash or not

try:
    with open("test.txt") as f:
        print(f)
finally:

    print("finally executed")
```

```
finally executed
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[30], line 5
      1 # finally
      2 # execute every time no matter code crash or not
      4 try:
----> 5     with open("test.txt") as f:
      6         print(f)
      7 finally:

File C:\Program Files\Python312\Lib\site-packages\IPython\core\interactiveshell.py:
324, in _modified_open(file, *args, **kwargs)
    317 if file in {0, 1, 2}:
    318     raise ValueError(
    319         f"IPython won't let you open fd={file} by default "
    320         "as it is likely to crash IPython. If you know what you are doing,
"
    321         "you can use builtins' open."
    322     )
--> 324 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'test.txt'
```

In [34]:
```python
# loging module
import logging
# basic configaration
logging.basicConfig(level = logging.DEBUG,
                    filename = 'app.log',
                    format = '%(asctime)s - %(levelname)s - %(message)s'
                   )

logging.debug("This is debug message")
logging.info("App started")
logging.warning("This is warning")
logging.error("This is error")
logging.critical("Critical issue!!")
```

```
INFO:root:App started
WARNING:root:This is warning
ERROR:root:This is error
CRITICAL:root:Critical issue!!
```

In [ ]:
```python
# try-catch examples
# prime number
import sys
def is_prime(num):
    try:
        if num < 2:
            return False
        for i in range(2,int(num**0.5)+1):
            if num % i == 0:
                return False
        return True
    except :
        print(sys.exc_info())
```

In [ ]:
```python
# CALCULATOR
import sys
flag = True
while flag:
```

```
    print("*****************************CALCULATOR*******************************")
    print()
    try:
        num1 = int(input("Enter the first number"))
        num2 = int(input("Enter the second number"))
        op = input("enter any operation(+,-,*,/)")
        match op:
            case '+':result = num1 + num2
            case '-' : result = num1 - num2
            case '*' : result = num1 * num2
            case '/' : result = num1 / num2
        print("------------------------------------------------------------")
        print(" your first input = {0} \n your second input = {1} \n operation = {
        print("------------------------------------------------------------")

    except:
        print("ERROR:" , sys.exc_info())

    print("*******************************************************************************
```

```
*****************************CALCULATOR*******************************

ERROR: (<class 'ZeroDivisionError'>, ZeroDivisionError('division by zero'), <traceb
ack object at 0x00000248FF268700>)
*******************************************************************************
*****************************CALCULATOR*******************************

------------------------------------------------------------
 your first input = 2
 your second input = 0
 operation = +
 result = 2
------------------------------------------------------------
*******************************************************************************
*****************************CALCULATOR*******************************

ERROR: (<class 'KeyboardInterrupt'>, KeyboardInterrupt('Interrupted by user'), <tra
ceback object at 0x00000248FF26A340>)
*******************************************************************************
*****************************CALCULATOR*******************************
```

In [1]:
```
1_000 + True
```

Out[1]: 1001

In [ ]:
```

```