```
evens = list(filter(is_even, nums))
double = list(map(lambda n : n*2, evens))

sums = reduce(add_all, double)
sums
print(sums)
```

```
56
```

In [123…

```
from functools import reduce

nums = [3,2,6,8,4,6,2]

evens = list(filter(is_even, nums))
double = list(map(lambda n : n*2, evens))
sums = (reduce(lambda a,b : a + b, double))

print(evens)
print(double)
print(sums)
```

```
[2, 6, 8, 4, 6, 2]
[4, 12, 16, 8, 12, 4]
56
```

15th July 2025

Exception Handling in Python

- so far we are done with oops concent & fundamental of python
- lets go towards some other side of python called ERRORS
- ERRORs are 3 type of errors -- COMPLILE TIME ERROR || LOGICAL ERROR || RUNTIME ERROR

Compile time error/ Syntactical Errors e.g., missing(:) wrong spelling

- if we print with wrong spelling & we working with if and we forget to columns those mistakes are syntatictal error &

those error are called as compile time error

Logical Error means wrong output

- when we add 2 + 3 = 4
- code is compiled and also code gives the output but we got the wrong output then those type

of error called as LOGICAL ERROR

- code is complied and no syntactical error. code not gives logical error as well
- but some time user might give wrong input by user (10/2 - user entered 10 & 2 code will work)
- when the user enter 6/0 then entry made by the user is called RUN TIME ERROR

Run time error e.g. divide by zero

- user getting error at run time those error called as run time
- being a developer you need to be take care of all these type of errors
- may be error occures for databases connection, network connection, application which interaction with serverw
- even though you are getting error you execution not stopped

Statement is Normal and Critical Normal means will not give any error

In [2]:
```
a = 5
b = 2
print(a/b)
print('bye')
```

```
2.5
bye
```

In [3]:
```
c = 5
d = 0

print(d/c)

print('bye')
```

```
0.0
bye
```

In [4]:
```
c = 5
d = 0

print('bye')

print(c/d)
```

```
bye
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[4], line 6
      2 d = 0
      4 print('bye')
----> 6 print(c/d)

ZeroDivisionError: division by zero
```

```
In [5]: c = 5
        d = 0

        print(c/d)

        print('bye')
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[5], line 4
      1 c = 5
      2 d = 0
----> 4 print(c/d)
      6 print('bye')

ZeroDivisionError: division by zero
```

- in the above code there are 2 senario

1- user some time not understand what is mean by zero division error 2- we are not getting bye statement

- that means execution stopped in between & we dont want that
- to solve this type of problem we need to use special block usint try block or try

statement

- this is called critical statement that means not sure that code will work or not
- it says try to execute & since your says TRY to execute but what if it is error then i

will except the error using EXCEPTION

- TRY to execute the statement & if you got the error then except the EXCEPTION that means

handle the exception

- Except the block you will print the message

In [6]:
```
c = 3
d = 0

try:
    print(c/d) # if these code give error then except the exception and print the message

except Exception:
    print('hey you cannot divide Number by zero')

print('bye')
```

```
hey you cannot divide Number by zero
bye
```

In [7]:
```
c = 3
d = 0

try:
    print(d/c) # if these code give error then except the exception and print the message

except Exception:
    print('hey you cannot divide Number by zero')

print('bye')
```

```
0.0
bye
```

In [8]:
```
c = 3
d = 0

try:
    print(c/d)

except Exception as e :
    print('hey you cannot divide Number by zero:', e)

print('bye')
```

```
hey you cannot divide Number by zero: division by zero
bye
```

- In real time when you open the database connection you need to close the db connection
- When open the door of the fridge if someone calls you need to close the fridge door
- the question is which block open & close weather it is try or except
- if you open the resorse then you must close it

In [11]:
```
c = 3
d = 3

try:
    print('resource open') # resource can be anything this can be file this can be database
    print(c/d)
    print('resource closed')

except Exception as e :
    print('hey you cannot divide Number by zero',':', e)
```

```
resource open
1.0
resource closed
```

In [12]:
```
c = 3
d = 0

try:
    print('resource open') # resource can be anything this can be file this can be database
    print(c/d)
    print('resource closed')

except Exception as e :
    print('hey you cannot divide Number by zero',':', e)
# if you see the output then resource is open but resource is not closed hear at exception block
```

```
resource open
hey you cannot divide Number by zero : division by zero
```

In [13]:
```
# if i print the error message as well along with message
c = 3
d = 0
```

```
try:
    print('resource open') # resource can be anything this can be file this can be database
    print(c/d)

except Exception as e :
    print('hey you cannot divide Number by zero',':', e)
    print('resource closed')
# if you see the output then resource is open but resource is not closed hear at exception block
```

```
resource open
hey you cannot divide Number by zero : division by zero
resource closed
```

In [15]:

```
# if i print the error message as well along with message
c = 3
d = 3

try:
    print('resource open') # resource can be anything this can be file this can be database
    print(c/d)

except Exception as e :
    print('hey you cannot divide Number by zero',':', e)
    print('resource closed')

finally:
    print('resource closed')
# it does not matter about any block finally it will execute
```

```
resource open
1.0
resource closed
```

In [ ]: