Filter

```
lst = list(range(-10, 10))
def positive(num):
  #return num if num>0 else None #this will also work
  return num>0

print(lst)
print(filter(positive, lst))
print(list(filter(positive, lst)))
```

```
[-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
<filter object at 0x7f7a5c54e550>
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Map

```
# map function
lst = list(range(1, 11))
def square(num):
  return num**2
squares = list(map(square, lst))
print(lst)
print(squares)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Reduce

```
from functools import reduce
def add(x,y):
  return x+y
lst = list(range(1,11))
print(lst)
print(reduce(add, lst))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
55
```

Keyword Arguments

```
def greet(**kwargs):
  if kwargs:
    print("Hello {}, your dob is {}".format(kwargs["name"], kwargs["dob"]))
```

```
greet(name="Raghu", dob="24 Apr,1993")
```

```
    Hello Raghu, your dob is 24 Apr,1993
```

```
## Arbitrary Functionn
```

```
def greet(*names):
  """
  No of arguments is not known
  """
  for name in names:
    print("Hello {0}".format(name))
greet("Raghu", "Vivek", "Bhavi")
```

```
    Hello Raghu
    Hello Vivek
    Hello Bhavi
```

Lambda Function

```
sqr = lambda x : x**2
print(sqr(2))
```

```
    4
```

```
from functools import reduce
lst = list(range(1, 21))
print(lst)
print("Filtered List (only even numbers) : "+str(list(filter(lambda x : x%2==0, lst))))
print("Squared list : "+str(list(map(lambda x: x**2, lst))))
print("Reduced list (summation) : "+str(reduce(lambda x,y : x+y, lst)))
```

```
    [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
    Filtered List (only even numbers) : [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
    Squared list : [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289,
    Reduced list (summation) : 210
```

Modules

```
import math
print(math.pi)
```

```
    3.141592653589793
```

```
import datetime
datetime.datetime.now()
```

```
    datetime.datetime(2021, 5, 14, 7, 31, 29, 606602)
```

```
import math as m
print(m.pi)
```

```
    3.141592653589793
```

```
from datetime import datetime
print(datetime.now())
```

```
    2021-05-14 07:33:46.325412
```

```
from math import *
print(pi)
```

```
    3.141592653589793
```

File I/O

File Opening modes

1. `r` - open in readonly mode
2. `w` - open in write mode. Create new file if the file doesn't exist. If exists, overwrite the file
3. `x` - open only for write mode. If a file exists, it'll throw error
4. `a` - append mode
5. `t` - only for text (default mode)
6. `b` - binary mode (storing lists of data, matrix and so on)
7. `+` - open for updating (reading & updating)

Functionalities available

- `f = open(filename, mode)` - open file in the mentioned mode
- `f = open(filename, encoding='utf-8')` - open with our necessary encoding
- `f.write('String contents')` - write contents to the file
- `f.close()` - close the file (use it with try/catch/finally)
- `f.read()` or `f.read(int)` - reads the content of the file character by character from the current cursor position
- `f.seek(int)` - move the cursor to what location in file
- `f.tell()` - returns the current cursor location
- `f.readline()` - reads the file line by line
- `f.readlines()` - returns the list of lines

`import os` functionalities

- `os.getcwd()` - get current working directory
- `os.chdir(path)` - change the current working directory
- `os.rename(file1, file2)` - rename the **file1** to **file2**
- `os.rmdir(dir)` - remove the **empty directory**

- `os.listdir(dir)` - list the contents of the directory **dir**
- `os.mkdir(dir)` - creates a new directory
- `import shutil`
  `shutil.rmtree('test1')` - removes the **non-empty directory**

```
!echo "New Text File" > 'example.txt'
!cat 'example.txt'
```

```
New Text File
```

```
try:
  f = open('example.txt', 'w')
  f.write("This is the first file\n")
  f.write("contains two lines")
finally:
  f.close
```

```
!cat 'example.txt'
```

```
This is the first file
contains two lines
```

```
f = open('example.txt', 'r')
print(f.read())
f.close()
```

```
This is the first file
contains two lines
```

```
f = open('example.txt', 'r')
print(f.read(4)) #file pointer moved to index 4
print(f.read(10))
```

```
This
 is the fi
```

```
print(f.tell()) #tell the current cursor location
```

```
14
```

```
f.seek(0) #move the cursor location to 0th location
print(f.read(15))
f.close()
```

```
This is the fir
```

```
#read a file line by line
f = open('example.txt', 'r')
for line in f:
```

```
  print(line)
f.close()
```

```
    This is the first file

    contains two lines
```

```
#read a file line by line
f = open('example.txt', 'r')
print(f.readline())
print(f.readline())
f.close()
```

```
    This is the first file

    contains two lines
```

```
#read a file line by line
f = open('example.txt', 'r')
print(f.readlines())
f.close()
```

```
    ['This is the first file\n', 'contains two lines']
```

```
import os
os.rename('example.txt', 'sample.txt')
f = open('sample.txt')
print(f.readline())
f.close()
```

```
    This is the first file
```

```
os.remove('sample.txt')
!ls
```

```
    cat  sample_data
```

```
print(os.getcwd()) #get current working directory
!mkdir 'test2'
!mkdir 'test1'
```

```
    /content
```

```
!ls
```

```
    cat  sample_data  test1  test2
```

```
os.chdir("/content/test1")
print(os.getcwd())
```

```
    /content/test1
```

```
!echo 'file1' > 'file1'
!echo 'file2' > 'file2'
!echo 'file3' > 'file3'
!echo 'file4' > 'file4'
print(os.listdir(os.getcwd()))
```

```
['file4', 'file3', 'file1', 'file2']
```

```
os.mkdir('directory1')
os.mkdir('directory2')
print(os.listdir(os.getcwd()))
os.rmdir('directory1') #remove an empty directory
print(os.listdir(os.getcwd()))
```

```
['directory1', 'file4', 'file3', 'file1', 'file2', 'directory2']
['file4', 'file3', 'file1', 'file2', 'directory2']
```

```
print(os.getcwd())
os.chdir("../")
print(os.getcwd())
```

```
/content/test1
/content
```

```
os.rmdir("test1") #non empty directory
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-62-6e6fc1d9ea85> in <module>()
----> 1 os.rmdir("test1") #non empty directory

OSError: [Errno 39] Directory not empty: 'test1'
```

SEARCH STACK OVERFLOW

```
import shutil
shutil.rmtree('test1')
print(os.getcwd())
```

```
/content
```

Exception handling

```
import sys
lst = ['b', 0, 2, -1]
for item in lst:
  try:

    if item < 0:
      raise ValueError("item less than zero")
```

```
    val = 1/item
    print('Division of 1/', item, ' is ', val)
  except(TypeError):
    print("Type Error for ", item)
  except ValueError as e:
    print("Value Error with message {", e, '}')
  except:
    print("Exception ", sys.exc_info()[0], ' ', sys.exc_info()[1],' occurred')
  finally:
    print('**** NEXT ITEM ****')
```

```
Type Error for  b
**** NEXT ITEM ****
Exception  <class 'ZeroDivisionError'>   division by zero  occurred
**** NEXT ITEM ****
Division of 1/ 2  is  0.5
**** NEXT ITEM ****
Value Error with message { item less than zero }
**** NEXT ITEM ****
```

Debugging

Debugging inputs

- `c` - continue
- `q` - quit
- `h` - help
- `list` - show the stopping point
- `p {}` - prints variable
- `p locals()` - prints all the local variables
- `p globals()` - prints all the global variables

```
import pdb

def seq(n):
  for i in range(n):
    pdb.set_trace() #debug breakpoint
    print(i)
  return

seq(5)
```

```
> <ipython-input-80-011075416f7c>(6)seq()
-> print(i)
(Pdb) c
0
> <ipython-input-80-011075416f7c>(5)seq()
-> pdb.set_trace() #debug breakpoint
(Pdb) c
1
> <ipython-input-80-011075416f7c>(6)seq()
```

```
-> print(i)
(Pdb) c
2
> <ipython-input-80-011075416f7c>(5)seq()
-> pdb.set_trace() #debug breakpoint
(Pdb) c
3
> <ipython-input-80-011075416f7c>(6)seq()
-> print(i)
(Pdb) c
4
```