FUNCTIONS

Built-in functions:

1. **abs( )** -> finds the absolute value
2. **all( )** -> checks if the given list/tuple/set is iterable, returns true/false
3. **dir( )** -> returns all the methods that can be performed
4. **divmod( )** -> prints quotient and remainder as a tuple
5. **enumerate( )** -> returns value associated with each index

   syntax: enumerate(iterable, start=0), here 0 is the starting index, which can be set to a different number

6. **filter( )** -> list(filter(function,iterable))
7. **isinstance( )** -> isinstance(object, classinfo), returns true/false
8. **map( )** -> list(map(function,iterable))
9. **reduce( )** -> reduce(function,iterable)
   required to import:
   from functools import reduce

Recursion function: a function calls other function within its scope

e.g. #factorial of a number:

```
def factorial(arg):
    return 1 if arg == 1 else (arg * factorial(arg - 1))
```

Lambda/Anonymous function: function defined without a name, used with filter( ), map( ), reduce( )

Syntax: lambda argument:output

e.g. lambda x:x*2

Function Arguments:

1. **Default arguments:**
   Syntax: functionName(non_default_argument, default_argument)
   e.g: greet(name, msg='Good Morning')
2. **Keyword Arguments:** allows to pass variable length of arguments to a function
   Syntax: def functionName(**kwargs):
   To call: functionName(argument1='value1', argument2='value2')
3. **Arbitary arguments:**
   Syntax: def functionName(*argument)
   To call: functionName('value1','value2','value3'…,'valueN')

Importing a Module:

1. import module_name
   Call -> module_name.function_name(arg1,arg2)
2. **Import with renaming:**
   import module_name as xyz
   Call -> xyz.function_name(arg)
3. **From…..import statement:**
   from module_name import function_name
   e.g. from datetime import datetime
   call -> datetime.now()
4. **Import all name:**
   from module_name import *
5. **Importing module from a package:**
   import package.subpackage.module

File Operations:

1. **Open a file:**
   f = open("file_name.txt")
2. **Read & write:**

   f = open("file_name.txt", 'r')

   f = open("file_name.txt", 'w')

3. **Closing a file:**

```
try:
        f = open("filename.txt")
finally:
        f.close()
```

4. **Write to file:**
   f = open("filename.txt",'operation'), operation -> w/x/a
   f.write("xyz")
5. **Read from a file:**
   f = open("filename.txt",'r')
   a. f.read()
   b. f.read(index)
   c. f.seek() -> changes current file cursor position
   d. f.tell() -> returns current file cursor position
   e. f.readline() -> to read individual lines
6. **Renaming and Deleting files:**
   import os
   a. os.rename("filename.txt","new_filename.txt") -> to rename a file
   b. os.remove("filename.txt") -> to remove a file

Python Directory and File Management:

1. **Get current directory:**
   import os
   os.getcmd()
2. **Change directory:**
   os.chdir("path")
3. **List directories and files:**
   os.listdir(os.getcwd())
4. **Making new directory:**
   os.mkdir('directory_name'), need to specify path or it will be created in the current directory
5. **Removing empty directory:**
   os.rmdir("directory_name")
6. **Removing non-empty directory:**
   import shutil
   shutil.rmtree('directory_name')

Exception Handling:

1. try:
   except:

requires:

```
import sys

sys.exc_info()[0]
```

2. Raising error:
   e.g. raise MemoryError("This is memory error")
3. try……finally

Debugging:

```
import pbd
```

use, pbd.set_trace() -> at breakpoint

few functions to debug:

a. c : continue
b. q : quit
c. h : help
d. list
e. p : print
f. p locals()
g. g global()