lesson 5

Functions:-

- Defining Functions
- Variable Scope
- Documentation
- lambda Expression
- iterators and Generators

☆ Defining Functions:-

ex: def cylinder_volume(height, radius):
    PI = 3.14159
    return height * PI * radius**2

☆ To call it
cylinder_volume(10,3)

☆ Variable Scope:-

- refers to which part of a program a variable can be referenced or used
- important to consider scope when variable in function. If a variable is created inside a function it can only be used within that function, Accessing that in outside that function is not possible

```
- def some_func():
      word = "hello"
  def another_func():
      word = 'good bye'
```
This works Fine

```
word = "hello"
def some_func():
    print(word)

some_func()
```
This works Fine

- Unbound local Error: since python doesn't allow functions to modify variable that are outside the function's scope

☆ Documentation:-

- used to make your code easier to understand

ex def DoDu(PoPu, land)
Comment ← """ Calc the PoPu """
        return pum / land

☆ Lambda Expressions:-

- used to create anonymous, that is functions that don't have a name, they are helpful for creating quick function that aren't needed later in your code
- This can be especially useful for higher order functions, or functions that take in other functions as arguments

ex def multiply(x,y):
    return x*y ⟹ multiply = lambda x,y: x*y

- used for short functions.

lesson 6

Scripting:-

- Python installations
- Running and Editing Scripts
- interacting with user input
- Handling Exceptions
- Reading and writing Files
- importing local standard, and third_party Modules
- Experimenting with an interpreter

✡ Errors and Exceptions:-

- Syntax errors: when python can't interpret our code, errors you are likely to get when you make a typo.
- Exceptions: when unexpected things happened during execution of a program even if the code is syntactically correct

✡ Handling Errors:-

- try statement
- Specifying Exceptions:

```
try:
    # some code
except (ValueError, keyboardInterrupt):
    # some code
```

✡ Accessing Error messages:

```
try:
    #some code
except ZeroDivisionError as e:
    #some code
    print("zero : {}".format(e))
```

✡ Reading and writing Files:-

- Reading File:-

```
=> f = open('m_path1.txt', 'r')
   file_data = f.read()
   f.close()
```

- write a File:-

```
=> f = open('mypath.txt', 'w')
   f.write("Hello there!")
   f.close()
```

- Too many open files

```
files = []
for i in range(100):
    files.append(open('some_file.txt', 'r'))
    print(i)
```

- with

```
with open('main.txt', 'r') as f:
    file_data = f.read
```

✡ importing Local Scripts:-

- importing use_full_func as uf

```
uf.add_five([1,2,3,4])
```

- using main Block
  - To avoid running executable

```
if __name__ == "__main__"
```

✡ The Standard Library

✡ Techniques For importing Modules

- From module_name import first_obj, second_obj
- import m.- as new_name
- From module name import * import everything

- To install Packages => pip install package_name.
- using a requirements.txt file
  => pip install -r requirements.txt