The topics covered in Python:

- **Programming Fundamentals** - **Python Basics** **- Python Fundamentals** **- Data Structures** **- Object Oriented Programming with Python** **- Functional Programming with Python** **- Lambdas** **- Decorators** **- Generators** **- Testing in Python** **- Debugging** **- Error Handling** **- Regular Expressions** **- Comprehensions** **- Modules**	• enumerate

- enumerate :
  for i,char in enumerate('HELLO'):
  print(I,char)

Output:

0	H
1	E
2	L
3	L
4	O

- Condition – if -elif-else

```
if(condition):
    print()
elsif(condition):
    printf()
else
    printf()
```

- Loop – for

```
for item in range(6):
    print("")
```

Loop – while

```
i=0
while(condition):
    print()
    i++
```

- List : mylist = ["apple", "banana", "cherry"]
- Tuple : mytuple = ("apple", "banana", "cherry")
- Set : myset = {"apple", "banana", "cherry"}
- Dictionary : thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 }
- Function:

```
def my_function(*args,**kwargs):
    print("Hello from a function")
my_function([1,2,3],x=a+b)
```

- Walrus operator

```
Letter = 'Nanduuuuu'
    if ((n:=len(Letter))>4):
     print(f'letter having {n} words')
```

- Map : - map(function, iterable)

```
def double(x):
    return x * 2
numbers = [1, 2, 3, 4, 5]
doubled_numbers = map(double, numbers)
print(list(doubled_numbers)) # Output: [2, 4, 6, 8, 10]
```

```
numbers = [1, 2, 3, 4, 5]
squared_numbers = map(lambda x: x**2, numbers)
print(list(squared_numbers)) # Output: [1, 4, 9, 16, 25]
```

- Filter :- filter(function, iterable)

```
def is_even(x):
   return x % 2 == 0
numbers = [1, 2, 3, 4, 5]
even_numbers = filter(is_even, numbers)
print(list(even_numbers)) # Output: [2, 4]
```

```
numbers = [1, 2, 3, 4, 5]

even_numbers = filter(lambda x: x % 2 == 0, numbers)

print(list(even_numbers)) # Output: [2, 4]
```

- Reduce :- reduce(function, iterable)

```
from functools import reduce

def multiply(x, y):
   return x * y

numbers = [1, 2, 3, 4, 5]
product = reduce(multiply, numbers)
print(product) # Output: 120
```

```
from functools import reduce


numbers = [1, 2, 3, 4, 5]

product = reduce(lambda x, y: x * y, numbers)

print(product) # Output: 120
```

- **Lambda**

```
x = lambda a: a + 10

print(x(5))
```

```
from functools import reduce
my_list = [1,2,3]
print("lambda function (by map):  ",list(map(lambda item : item*2 , my_list)))
print("lambda function (by filter):  ",list(filter(lambda item : item%2!=0 , my_list)))
print("lambda function (by reducer):  ",reduce(lambda acce, item:acce+item, my_list))
```

```
def apply_function_to_list(numbers, func):
   return list(map(func, numbers))

numbers = [1, 2, 3, 4, 5]
squared_numbers = apply_function_to_list(numbers, lambda x: x**2)
print(squared_numbers) # Output : [1, 4, 9, 16, 25]
```

- List / set / Dictionary Comprehension ()/{}/{:}

```
my_data_list=[char for char in 'hello']
my_data_list2 =[num for  num in range(0,100)]
my_data_list3=(num**2 for num in range (0,100))
my_data_list4 = [num**2 for num in range (0,100) if num %2==0]

print('\nList Comprehension (Character): ',my_data_list)
print('\nList Comprehension (range): ',my_data_list2)
print('\nList Comprehension (square root): ',my_data_list3)
print('\nList Comprehension(even from square root): ',my_data_list4)
```

- Decorators

```
def my_decorators(func):
    def wrap_func():
        print("********************")
        func()
        print("********************")
    return wrap_func

@my_decorators
def hello():
    print('helloooo')

hello()
```

```
#output :

********************

helloooo

********************
```

- Error handling using try-except-else-finally :

```
while True:
    try:
        age = int(input("Enter Your Age : "))
        10/age
        print(age)
    except ValueError:
        print("Please Enter a number ")
    except ZeroDivisionError:
        print("Please enter a number Other than 0")
    else:
        print("Thankyou")
        break
    finally:
        print("ok done")
```

- Debugging using pdb

```
import pdb
def add(num1,num2):
    pdb.set_trace()
    return num+num2

add(1,"shdj")
```

- **Generators**

```
def generator_function(num):
    for i in range(num):
        yield i*2
g=generator_function(10)
print(next(g))
next(g)
print(next(g))
```

```
# From this loop Generator's Can able to Execute one at a time. that's what generator's do.

# next function in print used to call output

# iter or __iter__ function used to call next until stopIteration
```

Object-oriented programming (OOP)

- **Abstraction :** is a fundamental concept in object-oriented programming that allows developers to hide complex implementation details from the users of a class or function.

```
from abc import ABC, abstractmethod

class Animal(ABC):
    @abstractmethod
    def make_sound(self):
        pass

class Dog(Animal):
    def make_sound(self):
        print("Woof!")

class Cat(Animal):
    def make_sound(self):
        print("Meow!")

animals = [Dog(), Cat()]

for animal in animals:
    animal.make_sound()
```

Output:

Woof!

Meow!

- **Inheritance** is a fundamental concept in object-oriented programming that allows developers to create new classes based on existing ones, inheriting the properties and behaviors of the parent class. This helps to reduce code duplication and increase code reusability.

```
class Animal:
    def __init__(self, name, species):
        self.name = name
        self.species = species

    def make_sound(self):
        print("This animal makes a sound.")

class Dog(Animal):
    def make_sound(self):
        print("Woof!")

class Cat(Animal):
    def make_sound(self):
        print("Meow!")

dog = Dog("Fido", "Dog")
cat = Cat("Fluffy", "Cat")

print(dog.name)  # Output: Fido
print(cat.species)  # Output: Cat

dog.make_sound()  # Output: Woof!
cat.make_sound()  # Output: Meow!
```

- **Polymorphism** is a fundamental concept in object-oriented programming that allows developers to use the same method or function in different ways, depending on the context. This helps to increase code flexibility and reduce code redundancy.

```
class Animal:
    def make_sound(self):
        pass

class Dog(Animal):
    def make_sound(self):
        print("Woof!")

class Cat(Animal):
    def make_sound(self):
        print("Meow!")

def animal_sounds(animal):
    animal.make_sound()

dog = Dog()
cat = Cat()

animal_sounds(dog)  # Output: Woof!
animal_sounds(cat)  # Output: Meow!
```

- **Encapsulation** is a fundamental concept in object-oriented programming that refers to the idea of grouping related data and functions into a single unit, and controlling access to that unit from outside.

```
class BankAccount:
    def __init__(self, account_number, balance):
        self.__account_number = account_number
        self._balance = balance

    def deposit(self, amount):
        self._balance += amount

    def withdraw(self, amount):
        if amount <= self._balance:
            self._balance -= amount
        else:
            print("Insufficient balance.")

    def get_balance(self):
        return self._balance
```

next→

```
account = BankAccount("12345", 1000)

# Accessing public method to deposit amount
account.deposit(500)
print(account.get_balance())  # Output: 1500

# Accessing public method to withdraw amount
account.withdraw(2000)
                    # Output: Insufficient balance.

# Accessing protected variable to print balance
print(account._balance)  # Output: 1500

# Trying to access private variable - will result in
AttributeError
print(account.__account_number)
```

-----End -----

- Class

```
class MyClass:
    def __init__(self, arg1, arg2):
        self.arg1 = arg1
        self.arg2 = arg2

    def my_method(self):
        # Do something

my_instance = MyClass("value1", "value2")
my_instance.my_method()
```

- Modules – importing fuctions from different python files(.py) or import using pip from web.

Modules.py

```
def divi(num1 ,num2):
    return num1/num2
```

Testing_modules.py

```
from Modules import divi    )
import shopping.more_shopping.shopping_cart

print(divi(10,2))

print(shopping.more_shopping.shopping_cart.shop(100))
```

- **Regular Expressions** is used to validation, search, check a piece or a group of string. eg : for email & Password checking for verification.

```
import re

pattern=re.compile('this')
string="search this inside of this string"
a=pattern.search(string)
print(a.span())
print(a.start())
print(a.group())
        #group is useful when you do multiple search of single word.

b=pattern.findall(string)       #disply all the strings you search for
c=pattern.fullmatch(string)     #check both full strings are equal
d=pattern.match(string)         #Display only strings upto match index
print(b)
```

```
#.....Email Validation....
import re

validation =re.compile(r"(^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)")
password= re.compile(r"[a-zA-Z0-9@#_]{8,}[0-9]")
passchecker="Nandu@1234"
gmail_id= "ng@g.com"
z=validation.search(gmail_id)
y=password.fullmatch(passchecker)
print(y)
if validation.search(gmail_id):print(z)
else: print("incorrect email id, Try again")
```

- Input / Output

```
#Better way to work with files in Python
''''
with open('text.txt',mode='r') as my_file2:
    print(my_file2.read())                # ()..... read a file ouside
'''
with open('sad.txt',mode='w')as my_file3:
    txt2=my_file3.write("new one")
    print(txt2)   #().....write also  Create new file

with open('app/newfile.txt',mode='w')as my_file4:
    txt3=my_file4.write("my Name is nandu")
    print(txt3)   #().....adding a new file to just created folder ..
# NB:  ./app/ means current folder ,  ../app/ means one folder back from current folder

with open('text.txt',mode='r+') as my_file2:
    text= my_file2.write(":)")
    print(text)               ## ()..... write into a file ouside
# r - read, r+ - read write , w- write,a - append

print('**********(1)********** ')
#-----Input a file and output itself
my_file=open('text.txt')
print(my_file.read())
my_file.seek(0)
print(my_file.read())
my_file.seek(0)
print(my_file.read())          #.....(1)read
```

```
print('**********(2)*************')
print(my_file.readline())  #.............(1)readline

print(my_file.readlines()) #.............(1)readlines

# Not working (2&3)
#my_file.close() #------------close it

#..............Excercise.................
from translate import Translator
translator=Translator(to_lang="kor")
try:
    with open('./app/newfile.txt',mode='r') as excer:
        textchanger= excer.read()
        trans=translator.translate(textchanger)
        print(trans)
        with open('./app/newtransilatefile.txt', mode='w') as excer2:
            excer2.write(trans)
except FileNotFoundError as err:
    print("your file path")
```

- Testing

sampleExcercise.py

```
import random
def find_random_num(guess,answer):
    if 0 < guess < 6:
        if guess==answer:
            print("You are a genious")
            return True
    else:
        print("enter a  value between 1 to 5")


if __name__ =='__main__':
    answer = random.randint(1,5)
    print(type(answer))
    while True:
        try:
            guess = int(input('Guess a Number: '))
            if (find_random_num(guess,answer)):
                break
        except ValueError:
                print("please enter a number")
                continue
```

Testing.py

```
import unittest
import sampleExcercise

class TestMain(unittest.TestCase):
    #we are inheriting what unittest gives to this class.

    def test_do_stuff_samevalue(self):
        result=sampleExcercise.find_random_num(5, 5)
        self.assertTrue(result)

    def test_do_stuff_largevalue(self):
        result=sampleExcercise.find_random_num(5, 11)
        self.assertFalse(result)

    def test_do_stuff_typechange(self):
        result=sampleExcercise.find_random_num(5, "5")
        self.assertFalse(result)

if __name__ =='__main__':
    unittest.main()
```

Image Processing – with Pillow Module in Python3

```
from PIL import Image , ImageFilter
img=Image.open(r"C:\Users\nandu\Desktop\Python\ImagePlayground\Pokedex
\pikachu.jpg")
#print(img)
#print(img.format)
#print(img.size)
#print(img.mode)
#print(dir(img))  #what all the things which Image has given to us.

#blur the image
filter_image =img.filter(ImageFilter.BLUR)
filter_image.save("blur.png",'png')

filter_image =img.filter(ImageFilter.SHARPEN)
filter_image.save("SHARPEN.png",'png')

filter_image =img.filter(ImageFilter.SMOOTH)
filter_image.save("SMOOTH.png",'png')

filter_image =img.convert("L")
filter_image.save("Grey.png",'png')
resze=filter_image.resize((300,300)) #this is a tuple use this inside
()
crooked=filter_image.rotate(90)
box=(100,100,400,400)
crop_region=filter_image.crop(box)
crop_region.save("cropped.png",'png')
#crooked.show()
img2=Image.open(r"C:\Users\nandu\Desktop\Python\ImagePlayground\Pokede
x\astro.jpg")
tumbnails=img2.resize((400,400))
tumbnails.save("tumbnailrezized.png",'png')

#if you don't want to squash the image and get its best aspect ratio,
# like eg. a profile pic for Facebook twitter like project use the
method below.
img2.thumbnail((400,400))
img2.save("thumnail2.png",'png')
```

JPG TO PNG Converter

Q. 'Pokedex' folder contains images with extension jpg, so you need to create another folder and save all the Converted pngimage to new folder with name 'new'.

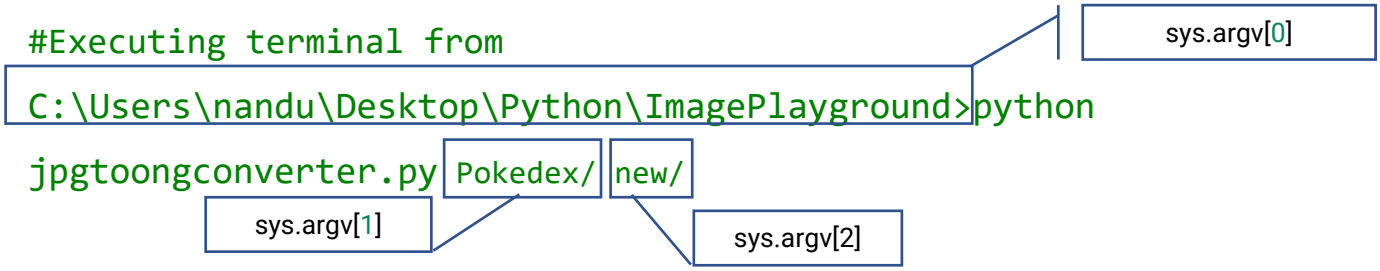
```
import sys
import os
from PIL import Image

#Executing terminal from
C:\Users\nandu\Desktop\Python\ImagePlayground>python
jpgtoongconverter.py Pokedex/ new/
#grab first and second argument Using sys
image_folder=sys.argv[1]
output_foder=sys.argv[2]

#check if new folder created or existed if not create one using os
if not os.path.exists(output_foder):
    os.makedirs(output_foder)

#loop through pokedex using os
for fileName in os.listdir(image_folder):

#Convert images to PNG using PIL
    img=Image.open(f'{image_folder}{fileName}')
#Save to the new Folder using PIL
    clearName=os.path.splitext(fileName)[0]  #{'Pickachu', '.jpg'}
    img.save(f'{output_foder}{clearName}.png','png')
    print("all done")
```



PDF Processing Using PyPDF2

Exercise : 1

```
import PyPDF2      #this a PyPDF2 v1.26 , syntax may be different from current

with open('dummy.pdf','rb') as file:  #rb means read binary
   reader=PyPDF2.PdfFileReader(file)
   print(reader.numPages)      #get number of pages
   page_file=reader.getPage(0)        #now page_file will savve first index page
   page_file.rotateCounterClockwise(90) #now page_file will roate saved page
   writer=PyPDF2.PdfFileWriter() #we are going to write something
   writer.addPage(page_file)      #yes we are writing on Page_file
   with open('tilt.pdf','wb') as new_file: #with within the with for saving
      writer.write(new_file) # roated page_file is created as tilt.pdf
```

Exercise: 2

```
import PyPDF2      #this a PyPDF2 v1.26 , syntax may be different from current
import sys


input=sys.argv[1:]  #gets the arguments from the terminal before
execution
def pdf_merger(pdf_list):
    merger=PyPDF2.PdfFileMerger()
    for pdf in pdf_list:
        merger.append(pdf)
    merger.write("super.pdf")        #creating a new merged pdf without
with open

pdf_merger(input)
```

Terminal

```
PS C:\Users\nandu\Desktop\Python\pdfPlayground>  python pdf.py dummy.pdf tilt.pdf twopage.pdf
```

*PdfFileMerger() – will merge pages one by one

*mergePage() – will merge page a page into another page.. eg watermark on every page

Exercise : 3

```
import PyPDF2       #this a PyPDF2 v1.26 , syntax may be different from current



template = PyPDF2.PdfFileReader(open(r'C:\Users\nandu\Desktop\Python\pdfPlayground\super.pdf','rb'))
watermark = PyPDF2.PdfFileReader(open(r'C:\Users\nandu\Desktop\Python\pdfPlayground\wtr.pdf','rb'))
output=PyPDF2.PdfFileWriter()


for i in range(template.getNumPages()):
    page_file=template.getPage(i)
    page_file.mergePage(watermark.getPage(0))
    output.addPage(page_file)
with open('watermarkouput.pdf','wb') as file:
    output.write(file)
```