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 :

Output:

0 H

1 E

2 L

3 L

4 O

for i,char in enumerate(‘HELLO’):

print(I,char)

* 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)

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

squared\_numbers = map(lambda x: x\*\*2, numbers)

print(list(squared\_numbers)) # Output: [1, 4, 9, 16, 25]

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]

* Filter :- filter(function, iterable)

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

even\_numbers = filter(lambda x: x % 2 == 0, numbers)

print(list(even\_numbers)) # Output: [2, 4]

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]

* Reduce :- reduce(function, iterable)

from functools import reduce

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

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

print(product) # Output: 120

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

* **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

#output :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

helloooo

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def my\_decorators(func):

def wrap\_func():

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

func()

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

return wrap\_func

@my\_decorators

def hello():

print('helloooo')

hello()

* 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**

# 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

def generator\_function(num):

for i in range(num):

yield i\*2

g=generator\_function(10)

print(next(g))

next(g)

print(next(g))

**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.

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 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🡪

* 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.

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))

Modules.py

def divi(num1 ,num2):

return num1/num2

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

#................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")

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)

* 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

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()

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

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\Pokedex\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

sys.argv[0]

#Executing terminal from

C:\Users\nandu\Desktop\Python\ImagePlayground>python jpgtoongconverter.py Pokedex/ new/

sys.argv[2]

sys.argv[1]

#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

Terminal

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

pdf\_merger(input)

\*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)