Python Notes

**PART 1**

**Math Operators:**

\*\* Exponent 2 \*\* 3 8

% Modulus/remainder 22 % 8 6

// Integer division/floored quotient 22 // 8 2

/ Division 22 / 8 2.75

\* Multiplication 3 \* 5 15

- Subtraction 5 - 2 3

+ Addition 2 + 2 4

**String Concatenation and Replication**

>>> "Joy"+"Sagnik"

'JoySagnik'

>>> "joy" \* 5

'joyjoyjoyjoyjoy'

**Storing Values in Variables**

>>> a= 12

>>> a

12

>>> b=22

>>> a+b

34

>>> c=a+b

>>> a=2

>>> c+2

36

>>> a="good"

>>> b="bye"

>>> a+b

'goodbye'

**Comments:**

#This is sample comment

**Input function:**

a=input()

print('It is good to meet you, ' + a)

**Length:**

print(len(a)) //Length

print('I am ' + str(23)+ " years old")

int() float()

**if/else :**

#This is coment

print('Enter your name:')

name=input()

a=5

print("Enter b:")

b=int(input())

c=a-b

if(name=="Sagnik" and c>0):

print("The result "+ str(c) +" is positive" )

elif(name=="Sagnik" and c<0):

print("The result "+ str(c) +" is negative" )

else:

print("Invalid User")

**While loop with Break and Continue:**

#This is coment

while(True):

print("Enter your name:")

name=input()

if name != "Sagnik":

continue

print("Enter your age:")

age=input()

if age != '23':

continue

else:

break

print("Success")

i=0

while i<5:

print(name)

i=i+1

**for Loops and the range() Function**

print('My name is:')

sum=0

for i in range(5):

print('Sagnik Roy in five times (' +str(i) + ')')

sum=sum+i

print('The sum is: '+ str(sum))

#This is coment

for i in range(0,10,2):

print(i)

**Importing Modules**

import random

for i in range(5):

print(random.randint(99,999))

import random

import sys

for i in range(5):

print(random.randint(99,999))

while True:

print('Type exit to exit')

response=input()

if (response=='exit'):

sys.exit()

print('You have entered'+response+',')

**Functions**

def hello():

print('Hi there')

print('How are you?')

hello()

**def Statements with Parameters**

def hello(name):

print('Hi' + name)

print('How are you?')

hello('Sagnik')

**Return Values and return Statements**

import random

def choose(number):

if(number==0):

return 'The number is zero, You are good programmer'

elif(number==1):

return 'The number is One, You are bad programmer'

elif(number==2):

return 'The number is Two, You are good programmer'

elif(number==3):

return 'The number is Three, You are bad programmer'

hold=choose(random.randint(0,3))

print(hold)

**None value**

>>> spam = print('Hello!')

Hello!

>>> None == spam

True

**Keyword Arguments and print()**

>>> print('cat','dog')

cat dog

**Local and Global Scope**

def bacon():

eggs=0

def spam():

eggs=10

bacon()

print(eggs)

spam()

**Output: 10**

def bacon():

eggs="Bacon Local"

print(eggs)

def spam():

eggs="Spam Local"

bacon()

print(eggs)

eggs="Global"

spam()

print(eggs)

**The global Statement**

def spam():

global eggs

eggs="Spam Local"

eggs="Global"

spam()

print(eggs)

**Exception Handling**

def divided(number):

try:

a=48//number

print(a)

except ZeroDivisionError:

print('Invalid argument')

divided(5)

divided(6)

divided(0)

divided(1)

**Lists**

>>> spam=['cat','bat','crow']

>>> spam

['cat', 'bat', 'crow']

>>> spam[1]

'bat

>>> spam=[['cat','bat','cockroach'],[1,2,3,4,5,6]]

>>> spam[0]

['cat', 'bat', 'cockroach']

>>> spam[1]

[1, 2, 3, 4, 5, 6]

>>> spam[0][2]

'cockroach'

**Negative Indexes**

>>> spam=[['cat','bat','cockroach'],[1,2,3,4,5,6]]

>>> spam[0]

['cat', 'bat', 'cockroach']

>>> spam[1]

[1, 2, 3, 4, 5, 6]

>>> spam[0][2]

'cockroach'

**Getting Sublists with Slices**

>>> spam

['cat', 'bat', 'rat', 'gate']

>>> spam[1:3]

['bat', 'rat']

>>> spam[2:4]

['rat', 'gate']

>>> spam[:2]

['cat', 'bat']

**Getting a List’s Length with len()**

>>> len(spam)

4

**Changing Values in a List with Indexes**

>>> spam[1]='Sagnik'

>>> spam

['cat', 'Sagnik', 'rat', 'gate']

**List Concatenation and List Replication**

>>> spam \* 3

['cat', 'Sagnik', 'rat', 'gate', 'cat', 'Sagnik', 'rat', 'gate', 'cat', 'Sagnik', 'rat', 'gate']

>>> spam

['cat', 'Sagnik', 'rat', 'gate']

>>> spam=spam+['bat']

>>> spam

['cat', 'Sagnik', 'rat', 'gate', 'bat']

**Removing Values from Lists with del Statements**

>>> spam

['cat', 'Sagnik', 'rat', 'gate', 'bat']

>>> spam

['cat', 'Sagnik', 'rat', 'gate', 'bat']

>>> del spam[1]

>>> spam

['cat', 'rat', 'gate', 'bat']

**Working with Lists**

names=[]

while True:

print("Enter the name of the list : (" + str(len(names)+1) + ") :")

d=input()

if(d==''):

break

names=names+[d]

print("The list elements are :")

for i in names:

print(' ' + i)

**The in and not in Operators**

>>> names

['Sagnik', 'Joy', 'Sam', 'Bonhi', 'Sudarsan', 'Apurba', 'Anuj']

>>> 'sagnik' in names

False

>>> 'Sagnik' in names

True

>>> 'Apurbo' in names

False

>>> 'Joy' not in sag

Traceback (most recent call last):

File "<pyshell#53>", line 1, in <module>

'Joy' not in sag

NameError: name 'sag' is not defined

>>> 'Joy' not in names

False

>>> 'Sammy' not in names

True

example :

names=[]

while True:

print("Enter the name of the list : (" + str(len(names)+1) + ") :")

d=input()

if(d==''):

break

if d in names:

print(d+" already in Nams")

continue

names=names+[d]

print("The list elements are :")

for i in names:

print(' ' + i)

**Augmented Assignment Operators**

bat=2

>>> bat+=2

>>>4

>>> a="Hello "

>>> a+= "World"

>>> a

'Hello World'

**Method:**

>>> names

['Sagnik', 'Antara', 'Aditi', 'Manisha', 'Nibedita', 'Moumita', 'Mahua', 'Shreya', 'Bonhi']

>>> names.index('Bonhi')

8

>>> names.append('Subhra')

>>> names

['Sagnik', 'Antara', 'Aditi', 'Manisha', 'Nibedita', 'Moumita', 'Mahua', 'Shreya', 'Bonhi', 'Subhra']

>>> names.insert(5,"Susmita")

>>> names

['Sagnik', 'Antara', 'Aditi', 'Manisha', 'Nibedita', 'Susmita', 'Moumita', 'Mahua', 'Shreya', 'Bonhi', 'Subhra']

>>> names.remove('Susmita')

>>> names

['Sagnik', 'Antara', 'Aditi', 'Manisha', 'Nibedita', 'Moumita', 'Mahua', 'Shreya', 'Bonhi', 'Subhra']

>>> names.insert(6,'Susmita')

>>> names

['Sagnik', 'Antara', 'Aditi', 'Manisha', 'Nibedita', 'Moumita', 'Susmita', 'Mahua', 'Shreya', 'Bonhi', 'Subhra']

>>> names.sort()

>>> names

['Aditi', 'Antara', 'Bonhi', 'Mahua', 'Manisha', 'Moumita', 'Nibedita', 'Sagnik', 'Shreya', 'Subhra', 'Susmita']

>>> len(names)

11

>>> names.sort(reverse=True)

>>> names

['Susmita', 'Subhra', 'Shreya', 'Sagnik', 'Nibedita', 'Moumita', 'Manisha', 'Mahua', 'Bonhi', 'Antara', 'Aditi']

**List-like Types: Strings and Tuples**

>>> names="Sagnik"

>>> names

'Sagnik'

>>> names[0]

'S'

>>> names[1:4]

'agn'

>>> names[1::2]

'ank'

>>> names[::-1]

'kingaS'

>>> names[1:-1]

'agni'

>>> 'S' in names

True

>>> 'S' not in names

False

>>> 'z' in names

False

>>> "z" not in names

True

**Mutable and Immutable Data Types**

>>> names

'Sagnik is my name'

>>> newNames=names[11:]+" is "+names[0:7]

>>> newNames

'y name is Sagnik '

>>> newNames=names[10:]+" is "+names[0:7]

>>> newNames

'my name is Sagnik '

**Tuple**

>> namet=('Sagnik',24,5.7)

>>> type(namet)

<class 'tuple'>

(24, 5.7)

>>> namet[::-1]

(5.7, 24, 'Sagnik')

**Converting Types with the list() and tuple() Functions**

>>> name

[1, 2, 3, 4]

>>> name=tuple(name)

>>> name

(1, 2, 3, 4)

>>> name=list(name)

>>> name

[1, 2, 3, 4]

**Copy a list**

>>> name

[1, 2, 3, 4]

>>> import copy

>>> name1=copy.copy(name)

>>> name1

[1, 2, 3, 4]

>>> name1[1]="Sagnik"

>>> name1

[1, 'Sagnik', 3, 4]

**Dictionaries and Structuring Data**

>>> name={'name':'Sagnik','Height':5.6,'Weight':82}

>>> name

{'name': 'Sagnik', 'Height': 5.6, 'Weight': 82}

>>> len(name)

3

>>> type(name)

<class 'dict'>

>>> name['Height']

5.6

>>> print("My name is:"+name['name'])

My name is:Sagnik

>>> name={22:'Sagnik',52:55}

>>> name

{22: 'Sagnik', 52: 55}

>>> name[52]

55

Example:

birthdays={'sagnik':'July 16','antara':'April 16'}

while True:

print('Enter the name of which birthday you want:')

name=input()

if(name==''):

break

if name in birthdays:

print('The birthday of '+name +' is:' +birthdays[name])

else:

print(name +' is not in Birthday list:')

print('Please update it: (For Name=' + name +') ')

bday=input()

birthdays[name]=bday

print('Success.')

**The keys(), values(), and items() Methods**

birthdays={'sagnik': 'July 16', 'antara': 'April 16', 'Sagnik': 'July 16', 'Bonhi': '13th April', 'Manisha': 'Saraswati Puja Date'}

for v in birthdays.values():

print(v)

for k in birthdays.keys():

print(k)

for i in birthdays.items():

print(i)

for k,v in birthdays.items():

print("The birthday of "+ k +" is "+ v)

**The setdefault() Method**

>>> birthdays

{'sagnik': 'July 16', 'antara': 'April 16', 'Sagnik': 'July 16', 'Bonhi': '13th April', 'Manisha': 'Saraswati Puja Date'}

>>> birthdays.setdefault('angsuman','Decembeer 25')

'Decembeer 25'

>>> birthdays

{'sagnik': 'July 16', 'antara': 'April 16', 'Sagnik': 'July 16', 'Bonhi': '13th April', 'Manisha': 'Saraswati Puja Date', 'angsuman': 'Decembeer 25'}

>>> birthdays.setdefault('angsuman','December 25')

'Decembeer 25'

>>> birthdays

{'sagnik': 'July 16', 'antara': 'April 16', 'Sagnik': 'July 16', 'Bonhi': '13th April', 'Manisha': 'Saraswati Puja Date', 'angsuman': 'Decembeer 25'}

Ex.

message='It was a bright cold day in April, and the clocks were striking thirteen.'

count={}

for character in message:

count.setdefault(character,0)

count[character]=count[character]+1

**Pretty Printing(sort the dictionary)**

import pprint

message='It was a bright cold day in April, and the clocks were striking thirteen.'

count={}

for character in message:

count.setdefault(character,0)

count[character]=count[character]+1

pprint.pprint(count)

**Ma n i p u l a t i n g S t r i n g s**

\' Single quote

\" Double quote

\t Tab

\n Newline (line break)

\\ Backslash

**Multiline Strings with Triple Quotes**

>>> print('''Hello

i am sagnik

I am fine ''')

Hello

i am sagnik

I am fine

**Multiline Comments**

""" This

is

a multiline comment """

print("Hello Guys")

**The upper(), lower(), isupper(), and islower() String Methods**

>>> spam

'Hello World!'

>>> spam.upper()

'HELLO WORLD!'

>>> spam.lower()

'hello world!'

>>> spam.isupper()

False

>>> spam=spam.upper()

>>> spam

'HELLO WORLD!'

>>> spam.isupper()

True

eg.

print('Hello! How are you?')

spam=input()

if spam.lower()=='great':

print('Very good!!')

else:

print('Do some interesting!')

**The isX String Methods**

isalpha() returns True if the string consists only of letters and is not blank.

isalnum() returns True if the string consists only of letters and numbers

and is not blank.

isdecimal() returns True if the string consists only of numeric characters

and is not blank.

isspace() returns True if the string consists only of spaces, tabs, and new-

lines and is not blank.

istitle() returns True if the string consists only of words that begin with

an uppercase letter followed by only lowercase letters.

Ex:

while True:

print('Enter your age')

age=input()

if age.isdecimal()==True:

break

print('Enter correct age:')

while True:

print('Enter a password(Combination with alphabets and numbers)')

pwd=input()

if pwd.isalnum()==True:

break

print('Enter correct password.')

**The startswith() and endswith() String Methods**

>>> spam

'Hello World!'

>>> spam.startswith("Hello")

True

>>> spam.endswith("!")

True

**The join() and split() String Methods**

>>> spam=['cat','rat','bat']

>>> ','.join(spam)

'cat,rat,bat'

>>> spam1="Hello world this is Sagnik Roy"

>>> spam1.split()

['Hello', 'world', 'this', 'is', 'Sagnik', 'Roy']

>>> spam1.split(',')

['Hello world this is Sagnik Roy']

>>> spam1

'Hello world this is Sagnik Roy'

>>> spam1.split('and')

['Hello world this is Sagnik Roy']

>>> spam1.split('l')

['He', '', 'o wor', 'd this is Sagnik Roy']

>>> 'and'.join(spam)

'catandratandbat'

eg,

>>> spam='''Hello

this is Sagnik Roy

I am using python '''

>>> spam.split('\n')

['Hello', 'this is Sagnik Roy', 'I am using python ']

**Justifying Text with rjust(), ljust(), and center()**

>>> spam='Hello'

>>> spam.rjust(20)

' Hello'

>>> spam.rjust(20,'#')

'###############Hello'

>>> spam.ljust(20)

'Hello '

>>> spam.ljust(20,"+")

'Hello+++++++++++++++'

>>> spam.center(20)

' Hello '

>>> spam.center(50,'=')

'======================Hello======================='

**Removing Whitespace with strip(), rstrip(), and lstrip()**

>>> spam='======================Hello======================='

>>> spam.strip('=')

'Hello'

>>> spam.lstrip('=')

'Hello======================='

>>> spam.rstrip('=')

'======================Hello'

**PART 2**

**Regular Expressions**

**Not Regular Expression Example**

def isPhoneNumber(text):

if len(text) != 12:

return False

for i in range(0,3):

if not text[i].isdecimal():

return False

if not text[3]=='-':

return False

for i in range(4,7):

if not text[i].isdecimal():

return False

if not text[7]=='-':

return False

for i in range(8,12):

if not text[i].isdecimal():

return False

return True

print('Enter a phone number (Example= 222-222-1234) :')

number=input()

print(isPhoneNumber(number))

message = 'Call me at 415-555-1011 tomorrow. 415-555-9999 is my office.'

for i in range(len(message)):

chunk=message[i:i+12]

if isPhoneNumber(chunk):

print("Phone Number found:" + chunk)

**Finding Patterns of Text with Regular Expressions**

>>> import re

>>> phoneNumberRegx=re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')

>>> mo=phoneNumberRegx.search('Call me at 415-555-1011 tomorrow.')

>>> print(mo.group())

415-555-1011

**Grouping with Parentheses**

>>> phoneNumberRegx=re.compile(r'(\d\d\d)-(\d\d\d-\d\d\d)')

>>> mo=phoneNumberRegx.search('180-214-1458')

>>> mo.group()

'180-214-145'

>>> mo.group(1)

'180'

>>> mo.group(2)

'214-145'

>>> mo.groups()

('180', '214-145')

>>> mo.group(0)

'180-214-145'

>>> areaCode,mainNumber=mo.groups()

>>> areaCode

'180'

>>> mainNumber

'214-145'

**Matching Multiple Groups with the Pipe**

>>> nameRegx=re.compile(r'Sagnik|Bonhi')

>>> mo=nameRegx.search('Bonhi and Sagnik')

>>> mo.group()

'Bonhi'

>>> nameRegx=re.compile(r'Super(Bat|Man|Cat)')

>>> mo2=nameRegx.search("He is a SuperMan")

>>> mo2.group()

'SuperMan'

>>> mo3=nameRegx.search("He is having SuperBat")

>>> mo3.group()

'SuperBat'

**Optional Matching with the Question Mark**

>>> import re

>>> batRegx=re.compile(r'bat(wo)?man')

>>> mo=batRegx.search('I need a batman')

>>> mo.group()

'batman'

>>> mo1=batRegx.search('I need a batwoman')

>>> mo1.group()

'batwoman'

>>> phoneRegx=re.compile(r'(\d\d\d-)?\d\d\d-\d\d\d\d')

>>> mo=phoneRegx.search("I haveto call with 189-8899")

>>> mo.group()

'189-8899'

>>> mo.groups()

(None,)

>>> mo1=phoneRegx.search("I have a call from 188-598-4521")

>>> mo1.group()

'188-598-4521'

**Matching Zero or More with the Star**

>>> batRegx=re.compile(r'bat(wo)\*man')

>>> mo=batRegx.search('I need a batman')

>>> mo.group()

'batman'

>>> mo1=batRegx.search('I need a batwowowowowoman')

>>> mo1.group()

'batwowowowowoman'

**Matching One or More with the Plus**

>>> batRegx=re.compile(r'bat(wo)+man')

>>> mo1=batRegx.search('I need a batwowowowowoman')

>>> mo1.group()

'batwowowowowoman'

>>> mo=batRegx.search('I need a Batman')

>>> mo

>>> mo.group()

Traceback (most recent call last):

File "<pyshell#27>", line 1, in <module>

mo.group()

AttributeError: 'NoneType' object has no attribute 'group'

**Matching Specific Repetitions with Curly Brackets**

(Ha){3}

(Ha)(Ha)(Ha)

(Ha){3,5}

((Ha)(Ha)(Ha))|((Ha)(Ha)(Ha)(Ha))|((Ha)(Ha)(Ha)(Ha)(Ha))

>>> haRegx=re.compile(r'(ha){3}')

>>> mo=haRegx.search("He smile like hahaha")

>>> mo.group()

'hahaha'

>>> mo1=haRegx.search("He smile like haha")

>>> mo1==None

True

**Greedy and Nongreedy Matching**

>>> greedyHaRegx=re.compile(r'(ha){3,5}')

>>> mo=greedyHaRegx.search('He smole like hahahahaha')

>>> mo.group()

'hahahahaha'

>>> nongreedyHaRegx=re.compile(r'(ha){3,5}?')

>>> mo1=nongreedyHaRegx.search('He smile like hahahahaha')

>>> mo1.group()

'hahaha'

**The findall() Method**

>>> phoneNumberRegx=re.compile(r'\d\d-\d\d\d\d\d\d\d\d\d\d')

>>> phoneNumberRegx.findall("My phone number 91-9475245371 and my mother's number 91-9614151551")

['91-9475245371', '91-9614151551']

>>> phoneNumberRegx=re.compile(r'(\d\d)?-(\d\d\d\d\d\d\d\d\d\d)')

>>> phoneNumberRegx.findall("My phone number 91-9475245371 and my mother's number 91-9614151551")

[('91', '9475245371'), ('91', '9614151551')]

**Character Classes**

\d Any numeric digit from 0 to 9.

\D Any character that is not a numeric digit from 0 to 9.

\w Any letter, numeric digit, or the underscore character.

(Think of this as matching “word” characters.)

\W Any character that is not a letter, numeric digit, or the

underscore character.

\s Any space, tab, or newline character. (Think of this as

matching “space” characters.)

\S Any character that is not a space, tab, or newline.

eg.

>>> xmasRegx=re.compile(r'\d+\s\w+')

>>> xmasRegx.findall('12 drummers, 11 pipers, 10 lords, 9 ladies, 8 maids, 7 swans, 6 geese, 5 rings, 4 birds, 3 hens, 2 doves, 1 partridge')

['12 drummers', '11 pipers', '10 lords', '9 ladies', '8 maids', '7 swans', '6 geese', '5 rings', '4 birds', '3 hens', '2 doves', '1 partridge']

**Making Your Own Character Classes**

>>> vowelRegx=re.compile(r'[aeiouAEIOU]')

>>> vowelRegx.findall('RoboCop eats baby food. BABY FOOD.')

['o', 'o', 'o', 'e', 'a', 'a', 'o', 'o', 'A', 'O', 'O']

>>> constRegx=re.compile(r'[^aeiouAEIOU]')

>>> constRegx.findall('RoboCop eats baby food. BABY FOOD.')

['R', 'b', 'C', 'p', ' ', 't', 's', ' ', 'b', 'b', 'y', ' ', 'f', 'd', '.', ' ', 'B', 'B', 'Y', ' ', 'F', 'D', '.']

**The Wildcard Character**

atRegx=re.compile(r’.at’)

>>> atRegx.findall('The cat in the hat sat on the flat mat.')

['cat', 'hat', 'sat', 'lat', 'mat']

**Matching Everything with Dot-Star**

nameRegx=re.compile(r'First Name:(.\*) Last Name:(.\*)')

>>> mo=nameRegx.search('First Name:Sagnik Last Name:Roy')

>>> mo.group()

'First Name:Sagnik Last Name:Roy'

>>> mo.group(1)

'Sagnik'

>>> mo.group(2)

'Roy'

>>> bracketNonGreedyRegx=re.compile(r'<.\*?>')

>>> mo=bracketNonGreedyRegx.search('<I am Sagnik> Doing Python >')

>>> mo.group()

'<I am Sagnik>'

>>> bracketGreedyRegx=re.compile(r'<.\*>')

>>> mo2=bracketGreedyRegx.search('<I am Sagnik> Doing Python >')

>>> mo2.group()

'<I am Sagnik> Doing Python >'

**Case-Insensitive Matching**

>>> robocopRegx=re.compile(r'robocop',re.I)

>>> mo=robocopRegx.search('Robocop')

>>> mo.group()

'Robocop'

>>> mo1=robocopRegx.search('RoboCop')

>>> mo1.group()

'RoboCop'

**Dotall**

>>> abc=""" Hello

This Is Sagnik Roy

I am using Python """

>>> abc

' Hello\nThis Is Sagnik Roy\nI am using Python '

>>> dotallRegx=re.compile(r'.\*',re.DOTALL)

>>> mo=dotallRegx.search(abc)

>>> mo.group()

' Hello\nThis Is Sagnik Roy\nI am using Python '

**R e ad i n g a n d Writing Files**

**Joining Statement**

>>> import os

>>> os.path.join('usr','bin','spam')

'usr/bin/spam'

>>> myfiles=['python.py','document.docx','sprad.xls']

>>> for files in myfiles:

os.path.join('~/Documnent',files)

'~/Documnent/python.py'

'~/Documnent/document.docx'

'~/Documnent/sprad.xls'

**The Current Working Directory**

>>> os.getcwd()

'/home/jarvis'

>>> os.chdir('/home/jarvis/Education')

>>> os.getcwd()

'/home/jarvis/Education'

**Absolute vs. Relative Paths**

../ = Parent Folder

./ = This Directory

**Creating New Folders with os.makedirs()**

>>> os.getcwd()

'/home/jarvis/Education'

>>> os.chdir('../Documents')

>>> os.getcwd()

'/home/jarvis/Documents'

>>> os.makedirs('./Example')

**Handle Relative and Absolute Path**

>>> os.path.abspath('.')

'C:\\Python34'

>>> os.path.abspath('.\\Scripts')

'C:\\Python34\\Scripts'

>>> os.path.isabs('.')

False

>>> os.path.isabs(os.path.abspath('.'))

True

>>> os.getcwd()

'/'

>>> os.chdir('/home/jarvis/Education/Django')

>>> os.path.relpath('./','/home/jarvis/Documents/Example')

'../../Education/Django'

>>> path='/home/jarvis/Education/hello.py'

>>> os.path.basename(path)

'hello.py'

>>> os.path.dirname(path)

'/home/jarvis/Education'

>>> os.path.split(path)

('/home/jarvis/Education', 'hello.py')

>>> path.split(os.path.sep)

['', 'home', 'jarvis', 'Education', 'hello.py']

**Finding File Sizes and Folder Contents**

>>> os.path.getsize('C:\\Windows\\System32\\calc.exe')

776192

>>> os.listdir('C:\\Windows\\System32')

['0409', '12520437.cpx', '12520850.cpx', '5U877.ax', 'aaclient.dll']

**Checking Path Validity**

>>> os.path.exists('/home/jarvis/Documents')

True

>>> os.path.isdir('/home/jarvis/Education')

True

>>> os.path.isfile('/home/jarvis/Education/hello.py')

True

**The File Reading/Writing Process**

o**pen(),read(),write(),close()**

>>> hellofile=open('example.txt','w')

>>> hellofile.write('Hello World \n')

13

>>> hellofile.close()

>>> hellofile=open('example.txt','a')

>>> hellofile.write("This is Sagnik Roy")

18

>>> hellofile.close()

>>> hellofile=open('example.txt')

>>> hellocontent=hellofile.read()

>>> hellocontent

'Hello World \nThis is Sagnik Roy'

>>> hellofile=open('example.txt')

>>> hellofile.readlines()

['Hello World \n', 'This is Sagnik Roy']

**Saving Variables with the pprint.pformat() Function**

>>> import pprint

>>> student=[{'name':'Sagnik','dept':'IT'},{'name':'Bonhi','dept':'CSE'}]

>>> pprint.pformat(student)

"[{'dept': 'IT', 'name': 'Sagnik'}, {'dept': 'CSE', 'name': 'Bonhi'}]"

>>> file=open('example.txt','w')

>>> file.write('Students:='+ pprint.pformat(student) +'\n')

79

>>> file.close()

>>> file=open('example.txt')

>>> file.readlines()

["Students:=[{'dept': 'IT', 'name': 'Sagnik'}, {'dept': 'CSE', 'name': 'Bonhi'}]\n"]

>>> import example

>>> example.Students

[{'dept': 'IT', 'name': 'Sagnik'}, {'dept': 'CSE', 'name': 'Bonhi'}]

>>> example.Students[1]['dept']

**The shutil Module**

The shutil (or shell utilities) module has functions to let you copy, move,

rename, and delete files in your Python programs. To use the shutil func-

tions, you will first need to use import shutil .

**Copy**

>>> import os

>>> os.getcwd()

'/home/jarvis'

>>> import shutil

>>> shutil.copy('/home/jarvis/Documents/example.py','./Education')

'./Education/example.py'

>>> shutil.copytree('./Documents/Example','./Education/untitled folder/abc')

'./Education/untitled folder/abc'

**Moving and Renaming Files and Folders**

>>> shutil.move('./Documents/example.py','./Education')

'./Education/example.py'

>>> shutil.move('./Education/example.py','./Education/a.py')

'./Education/a.py'

>>> shutil.move('./Education/a.py','./Education/a.txt')

'./Education/a.txt'

>>> shutil.copy('./Education/a.txt','./Education/a1.txt')

'./Education/a1.txt'

>>> shutil.move('./Education/a1.txt','./Education/example')

'./Education/example'

>>> shutil.move('./Education/a.txt','./Documents')

'./Documents/a.txt'

**Permanently Deleting Files and Folders**

Calling os.unlink(path) will delete the file at path .

Calling os.rmdir(path) will delete the folder at path . This folder must be

empty of any files or folders.

Calling shutil.rmtree(path) will remove the folder at path , and all files

and folders it contains will also be deleted.

>>> for files in os.listdir('./'):

if files.endswith('.txt'):

print(files)

os.unlink(files)

a3.txt

a1.txt

a2.txt

>>> os.rmdir('untitled folder')

**os.walk()**

import os

os.chdir('/home/jarvis/Documents')

for folderName,subFolderName,fileName in os.walk('Example'):

print("You are in :"+folderName)

for sflName in subFolderName:

print('The sub folder of: '+folderName+' is '+sflName)

for fName in fileName:

print('The file containing of : '+ folderName +' is '+fName)

**Compressing Files with the zipfile Module**

>>> import zipfile

>>> import os

>>> demozip=zipfile.ZipFile('/home/jarvis/Documents/a.zip')

>>> demozip

<zipfile.ZipFile filename='/home/jarvis/Documents/a.zip' mode='r'>

>>> demozip.namelist()

['Documents/', 'Documents/Al Sweigart - Automate the Boring Stuff with Python\_ Practical Programming for Total Beginners (2015, No Starch Press).pdf', 'Documents/\_\_pycache\_\_/', 'Documents/\_\_pycache\_\_/example.cpython-36.pyc', 'Documents/a/', 'Documents/a.txt', 'Documents/a/.fr-g4iPrw/', 'Documents/a/b/', 'Documents/a/b/d/', 'Documents/a/b/e/', 'Documents/a/b/e/asasa.txt', 'Documents/a/c/', 'Documents/a/c/f/', 'Documents/a/c/f/dsree.txt', 'Documents/a/c/g/', 'Documents/a/xyz.txt']

>>> info=demozip.getinfo('Documents/Al Sweigart - Automate the Boring Stuff with Python\_ Practical Programming for Total Beginners (2015, No Starch Press).pdf')

>>> info.file\_size

17449893

>>> info.compress\_size

14679815

**Extract**

>>> demozip=zipfile.ZipFile('/home/jarvis/Documents/a.zip')

>>> demozip.extractall()

>>> demozip.extract('Documents/Al Sweigart - Automate the Boring Stuff with Python\_ Practical Programming for Total Beginners (2015, No Starch Press).pdf','/home/jarvis/Pictures')

'/home/jarvis/Pictures/Documents/Al Sweigart - Automate the Boring Stuff with Python\_ Practical Programming for Total Beginners (2015, No Starch Press).pdf'

>>> exampleZip.close()

**Creating and Adding to ZIP Files**

>>> import zipfile

>>> demofile=zipfile.ZipFile('/home/jarvis/Documents/a.zip','w')

>>> demofile.write('./python.pdf',compress\_type=zipfile.ZIP\_DEFLATED)

>>> demofile.close()

**Debugging**

>>> raise Exception("Exception Happend!!")

Traceback (most recent call last):

File "<pyshell#0>", line 1, in <module>

raise Exception("Exception Happend!!")

Exception: Exception Happend!!

eg,

def boxPrint(symbol,height,width):

if len(symbol)!=1:

raise Exception('Symbol Must be Single Character')

if height<=2:

raise Exception('Height must be greater than 2')

if width<=2:

raise Exception('Width must be greater than 2')

print(symbol \* width)

for i in range(height-2):

print(symbol+' '\*(width-2)+symbol)

print(symbol \* width)

tuple=(('\*',4,10),('@',20,30),('+',1,5),('zz',5,6))

for sym,hgt,wth in tuple:

try:

boxPrint(sym,hgt,wth)

except Exception as err:

print("An Exception hapend"+str(err))

**Getting the Traceback as a String**

def spam():

bacon()

def bacon():

raise Exception("This is a error!!")

spam()

**output:**

Traceback (most recent call last):

File "/home/jarvis/Education/example.py", line 5, in <module>

spam()

File "/home/jarvis/Education/example.py", line 2, in spam

bacon()

File "/home/jarvis/Education/example.py", line 4, in bacon

raise Exception("This is a error!!")

Exception: This is a error!!

**ex.**

import traceback

try:

raise Exception('An Exception has happend')

except:

file=open('errorInfo.txt','w')

file.write(traceback.format\_exc())

file.close()

print('The error mssage have save in errorinfo.txt file')

**Assertions**

>>> door='open'

>>> assert door=='open','It needs to open'

>>> door='close'

>>> assert door=='open','It needs to open'

Traceback (most recent call last):

File "<pyshell#15>", line 1, in <module>

assert door=='open','It needs to open'

AssertionError: It needs to open

**Using an Assertion in a Traffic Light Simulation**

market\_2nd = {'ns': 'green', 'ew': 'red'}

mission\_16th = {'ns': 'red', 'ew': 'green'}

def traffficLights(stoplight):

for key in stoplight.keys():

if stoplight[key]=='red':

stoplight[key]=='yellow'

elif stoplight[key]=='yellow':

stoplight[key]=='green'

elif stoplight[key]=='green':

stoplight[key]=='red'

traffficLights(market\_2nd)

**Logging**

import logging

logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s -%(levelname)s- %(message)s')

EX.

import logging

logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s -%(levelname)s- %(message)s')

logging.debug('Start the prorgram')

def factorial(n):

logging.debug('Start the factorial of (%s)' % (n))

result=1

for i in range(1,n+1):

result\*=i

logging.debug('i is: '+str(i)+' result is: '+str(result))

logging.debug('End of factorial (%s)'%n)

print(factorial(5))

logging.debug('End of program')

>>> import logging

>>> logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s -

%(levelname)s - %(message)s')

>>> logging.debug('Some debugging details.')

2015-05-18 19:04:26,901 - DEBUG - Some debugging details.

>>> logging.info('The logging module is working.')

2015-05-18 19:04:35,569 - INFO - The logging module is working.

>>> logging.warning('An error message is about to be logged.')

2015-05-18 19:04:56,843 - WARNING - An error message is about to be logged.

>>> logging.error('An error has occurred.')

2015-05-18 19:05:07,737 - ERROR - An error has occurred.

>>> logging.critical('The program is unable to recover!')

2015-05-18 19:05:45,794 - CRITICAL - The program is unable to recover!

**Disabling Logging**

import logging

logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s -%(levelname)s- %(message)s')

logging.critical('Crtical Error! Critical Error')

logging.disable(logging.CRITICAL)

logging.critical("This is critical Error")

logging.error('This is Error!!!!!')

**IDLE’s Debugger**

**Go**

Clicking the Go button will cause the program to execute normally until it

terminates or reaches a breakpoint. (Breakpoints are described later in this

chapter.) If you are done debugging and want the program to continue nor-

mally, click the Go button.

**Step**

Clicking the Step button will cause the debugger to execute the next line

of code and then pause again. The Debug Control window’s list of global

and local variables will be updated if their values change. If the next line of

code is a function call, the debugger will “step into” that function and jump

to the first line of code of that function.

**Over**

Clicking the Over button will execute the next line of code, similar to the

Step button. However, if the next line of code is a function call, the Over

button will “step over” the code in the function. The function’s code will be

executed at full speed, and the debugger will pause as soon as the function

call returns. For example, if the next line of code is a print() call, you don’treally care about code inside the built-in print() function; you just want the

string you pass it printed to the screen. For this reason, using the Over but-

ton is more common than the Step button.

**Out**

Clicking the Out button will cause the debugger to execute lines of code

at full speed until it returns from the current function. If you have stepped

into a function call with the Step button and now simply want to keep exe-

cuting instructions until you get back out, click the Out button to “step out”

of the current function call.

**Quit**

If you want to stop debugging entirely and not bother to continue executing

the rest of the program, click the Quit button. The Quit button will imme-

diately terminate the program. If you want to run your program normally

again, select Debug4Debugger again to disable the debugger.

**WEB SCRAPPING**

**webbrowser** Comes with Python and opens a browser to a specific page.

**Requests** Downloads files and web pages from the Internet.

**Beautiful** **Soup** Parses HTML, the format that web pages are written in.

**Selenium** Launches and controls a web browser. Selenium is able to

fill in forms and simulate mouse clicks in this browser.

Import webbrowser

webbrowser.open('<http://www.google.com/>')

**Handle the Command Line Arguments**

import sys

import webbrowser

if(len(sys.argv)>1):

res=''.join(sys.argv[1:])

webbrowser.open('<https://www.google.com/maps/'+res>)

**Downloading fles from Web using Requests**

>>> import requests

u >>> res = requests.get('http://www.gutenberg.org/cache/epub/1112/pg1112.txt')

>>> type(res)

<class 'requests.models.Response'>

v >>> res.status\_code == requests.codes.ok

True

>>> len(res.text)

178981

>>> print(res.text[:250])

**Errors:**

res=requests.get('http://inventwithpython.com/page\_that\_does\_not\_exist')

>>> res.status\_code

404

>>> res.raise\_for\_status

<bound method Response.raise\_for\_status of <Response [404]>>

>>> res.raise\_for\_status()

Traceback (most recent call last):

File "<pyshell#9>", line 1, in <module>

res.raise\_for\_status()

File "/usr/lib/python3/dist-packages/requests/models.py", line 935, in raise\_for\_status

raise HTTPError(http\_error\_msg, response=self)

requests.exceptions.HTTPError: 404 Client Error: Not Found for url: <http://inventwithpython.com/page_that_does_not_exist>

Example:

import requests

try:

res=requests.get('http://inventwithpython.com/page\_that\_does\_not\_exist')

res.raise\_for\_status()

except Exception as ex:

print("The error code is: "+ str(ex))

**Saving Downloaded Files to the Hard Drive**

import requests

res=requests.get('http://www.gutenberg.org/cache/epub/1112/pg1112.txt')

res.raise\_for\_status()

file=open('downloadtext.txt','wb')

for chunk in res.iter\_content(100000):

file.write(chunk)

file.close()

**Creating Beautiful Soup Object from HTML**

>>> import requests, bs4

>>> res = requests.get('http://nostarch.com')

>>> res.raise\_for\_status()

>>> noStarchSoup = bs4.BeautifulSoup(res.text)

>>> type(noStarchSoup)

<class 'bs4.BeautifulSoup'>

>>> exampleFile = open('example.html')

>>> exampleSoup = bs4.BeautifulSoup(exampleFile)

>>> type(exampleSoup)

<class 'bs4.BeautifulSoup'>

**Finding an Element with the select() Method**

soup.select('div') All elements named <div>

soup.select('#author') The element with an id attribute of author

soup.select('.notice') All elements that use a CSS class attri-

bute named notice

soup.select('div span') All elements named <span> that are within

an element named <div>

soup.select('div > span') All elements named <span> that are

directly within an element named <div> ,

with no other element in between

soup.select('input[name]') All elements named <input> that have a

name attribute with any value

soup.select('input[type="button"]') All elements named <input> that have an

attribute named type with value button

Eg,

>>> import bs4

>>> exampleFile = open('example.html')

>>> exampleSoup = bs4.BeautifulSoup(exampleFile.read())

>>> elems = exampleSoup.select('#author')

>>> type(elems)

<class 'list'>

>>> len(elems)

1

>>> type(elems[0])

<class 'bs4.element.Tag'>

>>> elems[0].getText()

'Al Sweigart'

>>> str(elems[0])

'<span id="author">Al Sweigart</span>'

>>> elems[0].attrs

{'id': 'author'}

and.

>>> pElems = exampleSoup.select('p')

>>> str(pElems[0])

'<p>Download my <strong>Python</strong> book from <a href="http://

inventwithpython.com">my website</a>.</p>'

>>> pElems[0].getText()

'Download my Python book from my website.'

>>> str(pElems[1])

'<p class="slogan">Learn Python the easy way!</p>'

>>> pElems[1].getText()

'Learn Python the easy way!'

>>> str(pElems[2])

'<p>By <span id="author">Al Sweigart</span></p>'

>>> pElems[2].getText()

'By Al Sweigart

**Getting Data from an Element’s Attributes**

>>> import bs4

>>> soup = bs4.BeautifulSoup(open('example.html'))

>>> spanElem = soup.select('span')[0]

>>> str(spanElem)

'<span id="author">Al Sweigart</span>'

>>> spanElem.get('id')

'author'

>>> spanElem.get('some\_nonexistent\_addr') == None

True

>>> spanElem.attrs

{'id': 'author'}

**Working with Excel Spreadsheets**

**Opening Excel Documents with OpenPyXL**

>>> import openpyxl

>>> wb=openpyxl.load\_workbook('./Education/example/example.xlsx')

>>> type(wb)

<class 'openpyxl.workbook.workbook.Workbook'>

**Getting Sheets from the Workbook**

>>> wb.get\_sheet\_names()

['Sheet1', 'Sheet2', 'Sheet3']

>>> wb.get\_active\_sheet()

<Worksheet "Sheet3">

>>> sheet=wb.get\_sheet\_by\_name('Sheet2')

>>> sheet

<Worksheet "Sheet2">

>>> sheet.title

'Sheet2'

>>> anothersheet=wb.get\_active\_sheet()

>>> anothersheet

<Worksheet "Sheet3">

**Getting Cells from the Sheets**

>>> sheet=wb.get\_sheet\_by\_name('Sheet1')

>>> sheet['A1']

<Cell 'Sheet1'.A1>

>>> sheet['B1'].value

datetime.datetime(2015, 5, 4, 0, 0)

>>> sheet['C1'].value

datetime.time(1, 34, 2)

>>> sheet['D1'].value

'PM'

>>>

>>> sheet['E1'].value

'Apples'

>>> c=sheet['E3']

>>> c.row

3

>>> c.column

'E'

>>> c.value

'Pears'

>>> 'Row'+str(c.row)+' ,Column'+c.column+" is "+str(c.value)

'Row3 ,ColumnE is Pears'

>>> sheet.cell(row=2,column=5)

<Cell 'Sheet1'.E2>

>>> sheet.cell(row=2,column=5).value

'Cherries'

>>> for i in range(1,8,2):

print(i,str(sheet.cell(row=i,column=5).value))

1 Apples

3 Pears

5 Apples

7 Strawberries

**Creating and Saving Excel Documents**

>>> wb.get\_sheet\_names()

['Sheet1', 'Sheet2', 'Sheet3']

>>> sheet=wb.get\_sheet\_by\_name('Sheet2')

>>> sheet.title

'Sheet2'

>>> sheet.title="emp"

>>> wb.get\_sheet\_names()

['Sheet1', 'emp', 'Sheet3']

>>wb.save(‘example.xlxs’)

**Creating and Removing Sheets**

>>> wb.get\_sheet\_names()

['Sheet1', 'emp', 'Sheet3', 'Chart']

>>> wb.create\_sheet(index=0,title='First Sheet')

<Worksheet "First Sheet">

>>> wb.create\_sheet(index=2,title='Middle Sheet')

<Worksheet "Middle Sheet">

>>> wb.get\_sheet\_names()

['First Sheet', 'Sheet1', 'Middle Sheet', 'emp', 'Sheet3', 'Chart']

>>> wb.remove\_sheet(wb.get\_sheet\_by\_name('First Sheet'))

>>> wb.remove\_sheet(wb.get\_sheet\_by\_name('Middle Sheet'))

>>> wb.remove\_sheet(wb.get\_sheet\_by\_name('Chart'))

>>> wb.get\_sheet\_names()

['Sheet1', 'emp', 'Sheet3']

**Writing Values to Cells**

>>> import openpyxl

>>> wb = openpyxl.Workbook()

>>> sheet = wb.get\_sheet\_by\_name('Sheet')

>>> sheet['A1'] = 'Hello world!'

>>> sheet['A1'].value

'Hello world!'

**Formulas**

import openpyxl

wb = openpyxl.Workbook()

sheet = wb.get\_active\_sheet()

sheet['A1'] = 200

sheet['A2'] = 300

sheet['A3'] = '=SUM(A1:A2)'

wb.save('writeFormula.xlsx')

**Adjusting Rows and Columns**

**Setting Row Height and Column Width**

import openpyxl

wb = openpyxl.Workbook()

sheet = wb.get\_active\_sheet()

sheet['A1'] = 'Tall row'

sheet['B2'] = 'Wide column'

sheet.row\_dimensions[1].height = 70

sheet.column\_dimensions['B'].width = 20

wb.save('dimensions.xlsx')

**Merging and Unmerging Cells**

sheet = wb.get\_active\_sheet()

sheet.merge\_cells('A1:D3')

sheet['A1'] = 'Twelve cells merged together.'

sheet.merge\_cells('C5:D5')

sheet['C5'] = 'Two merged cells.'

wb.save('merged.xlsx')

**Freeze Panes**

import openpyxl

wb = openpyxl.load\_workbook('produceSales.xlsx')

sheet = wb.get\_active\_sheet()

sheet.freeze\_panes = 'A2'

wb.save('freezeExample.xlsx')

sheet.freeze\_panes = 'A2' Row 1

sheet.freeze\_panes = 'B1' Column A

sheet.freeze\_panes = 'C1' Columns A and B

sheet.freeze\_panes = 'C2' Row 1 and columns A and B

sheet.freeze\_panes = 'A1' or

sheet.freeze\_panes = None No frozen panes

**Working with PDF and Word Documents**

**Extracting Text from PDFs**

import PyPDF2

>>> pdfFileObj = open('meetingminutes.pdf', 'rb')

>>> pdfReader = PyPDF2.PdfFileReader(pdfFileObj)

u >>> pdfReader.numPages

19

v >>> pageObj = pdfReader.getPage(0)

w >>> pageObj.extractText()

'OOFFFFIICCIIAALL BBOOAARRDD MMIINNUUTTEESS Meeting of March 7, 2015

\n

The Board of Elementary and Secondary Education shall provide leadership

and create policies for education that expand opportunities for children,

empower families and communities, and advance Louisiana in an increasingly

competitive global market. BOARD of ELEMENTARY and SECONDARY EDUCATION '

**Decrypting**

>>> pdfReader=PyPDF2.PdfFileReader(pdfFieObject)

>>> pdfReader

<PyPDF2.pdf.PdfFileReader object at 0x7f61b412f8d0>

>>> pdfReader.getIsEncrypted()

True

>>> pdfReader.decrypt('qwerty')

**Copying**

import PyPDF2

pdf1File = open('meetingminutes.pdf', 'rb')

pdf2File = open('meetingminutes2.pdf', 'rb')

pdf1Reader = PyPDF2.PdfFileReader(pdf1File)

pdf2Reader = PyPDF2.PdfFileReader(pdf2File)

pdfWriter = PyPDF2.PdfFileWriter()

x

y >>> for pageNum in range(pdf1Reader.numPages):

pageObj = pdf1Reader.getPage(pageNum)

pdfWriter.addPage(pageObj)

x

y >>> for pageNum in range(pdf2Reader.numPages):

pageObj = pdf2Reader.getPage(pageNum)

pdfWriter.addPage(pageObj)

z >>>

>>>

>>>

>>>

>>>

pdfOutputFile = open('combinedminutes.pdf', 'wb')

pdfWriter.write(pdfOutputFile)

pdfOutputFile.close()

pdf1File.close()

pdf2File.close()

**Rotating**

import PyPDF2

fpdfFile=open('./Python3.pdf','rb')

pdfReader=PyPDF2.PdfFileReader(fpdfFile)

page=pdfReader.getPage(0)

page.rotateClockwise(90)

pdfWriter=PyPDF2.PdfFileWriter()

pdfWriter.addPage(page)

outputFile=open('/home/jarvis/Education/Python.pdf','wb')

pdfWriter.write(outputFile)

outputFile.close()

**Mering**

import PyPDF2

pdf1File=open('./Python3.pdf','rb')

pdf2File=open('./Python2.pdf','rb')

pdf1Reader=PyPDF2.PdfFileReader(pdf1File)

pdf2Reader=PyPDF2.PdfFileReader(pdf2File)

pdf1page=pdf1Reader.getPage(1)

pdf2page=pdf2Reader.getPage(2)

pdf1page.mergePage(pdf2page)

pdfWriter=PyPDF2.PdfFileWriter()

pdfWriter.addPage(pdf1page)

outputFile=open('/home/jarvis/Education/Python.pdf','wb')

pdfWriter.write(outputFile)

outputFile.close()

**Encrypting**

import PyPDF2

pdf1File=open('./Python3.pdf','rb')

pdf1Reader=PyPDF2.PdfFileReader(pdf1File)

pdfWriter=PyPDF2.PdfFileWriter()

for pagenum in range(pdf1Reader.numPages):

page=pdf1Reader.getPage(pagenum)

pdfWriter.addPage(page)

print('\_\_\_\_\_Page Added\_\_\_\_')

pdfWriter.encrypt('qwerty')

outputFile=open('/home/jarvis/Education/Python.pdf','wb')

pdfWriter.write(outputFile)

outputFile.close()

**Word Documents**

**Reading Word Documents**

>>> import docx

u >>> doc = docx.Document('demo.docx')

v >>> len(doc.paragraphs)

7

w >>> doc.paragraphs[0].text

'Document Title'

x >>> doc.paragraphs[1].text

'A plain paragraph with some bold and some italic'

y >>> len(doc.paragraphs[1].runs)

4

z >>> doc.paragraphs[1].runs[0].text

'A plain paragraph with some '

{ >>> doc.paragraphs[1].runs[1].text

'bold'

| >>> doc.paragraphs[1].runs[2].text

' and some '

} >>> doc.paragraphs[1].runs[3].text

'italic

**Getting the Full Text from a .docx File**

#! python3

import docx

def getText(filename):

doc = docx.Document(filename)

fullText = []

for para in doc.paragraphs:

fullText.append(para.text)

return '\n'.join(fullText)

>>> doc = docx.Document('demo.docx')

>>> doc.paragraphs[0].text

'Document Title'

>>> doc.paragraphs[0].style

'Title'

>>> doc.paragraphs[0].style = 'Normal'

>>> doc.paragraphs[1].text

'A plain paragraph with some bold and some italic'

>>> (doc.paragraphs[1].runs[0].text, doc.paragraphs[1].runs[1].text, doc.

paragraphs[1].runs[2].text, doc.paragraphs[1].runs[3].text)

('A plain paragraph with some ', 'bold', ' and some ', 'italic')

>>> doc.paragraphs[1].runs[0].style = 'QuoteChar'

>>> doc.paragraphs[1].runs[1].underline = True

>>> doc.paragraphs[1].runs[3].underline = True

>>> doc.save('restyled.docx')

**Add Paragraph and Headings**

>>> import docx

>>> doc=docx.Document('./Education/example.docx')

>>> doc.add\_paragraph('This is Paragraph 1')

<docx.text.paragraph.Paragraph object at 0x7ff2a1686160>

>>> paragraph2=doc.add\_paragraph('This is Paragraph 2')

>>> paragraph3=doc.add\_paragraph('This is paragraph3')

>>> paragraph2.add\_run("This is adding to paragraph 2")

<docx.text.run.Run object at 0x7ff2a0081e80>

>>> doc.save('./Education/example.docx')

**Working with CSV Files a n d JSON Data**

CSV stands for “comma-separated values,”

JSON is short for JavaScript Object Notation.

**The csv Module**

>>> import csv

>>> examplefile=open('./Education/example.csv')

>>> exampleReader=csv.reader(examplefile)

>>> exampleData=list(exampleReader)

>>> exampleData

[['4/5/2015 13:34', 'Apples', '73'], ['4/5/2015 3:41', 'Cherries', '85'], ['4/6/2015 12:46', 'Pears', '14'], ['4/8/2015 8:59', 'Oranges', '52'], ['4/10/2015 2:07', 'Apples', '152'], ['4/10/2015 18:10', 'Bananas', '23'], ['4/10/2015 2:40', 'Strawberries', '98']]

>>> exampleData[0][1]

'Apples'

>>> exampleData[2][1]

'Pears'

**Reading Data from Reader Objects in a for Loop**

>>> examoleFile=open('./Education/example.csv')

>>> exampleReader=csv.reader(examoleFile)

>>> exampleReader

<\_csv.reader object at 0x7f1025050438>

>>> for row in exampleReader:

print('Row #'+str(exampleReader.line\_num)+' '+str(row))

Row #1 ['4/5/2015 13:34', 'Apples', '73']

Row #2 ['4/5/2015 3:41', 'Cherries', '85']

Row #3 ['4/6/2015 12:46', 'Pears', '14']

Row #4 ['4/8/2015 8:59', 'Oranges', '52']

Row #5 ['4/10/2015 2:07', 'Apples', '152']

Row #6 ['4/10/2015 18:10', 'Bananas', '23']

Row #7 ['4/10/2015 2:40', 'Strawberries', '98']

**Writer Objects**

>>> outputFile = open('./Education/example.csv', 'w', newline='')

>>> exampleWriter=csv.writer(outputFile)

>>> exampleWriter.writerow(['Sagnik','25','IT'])

14

>>> exampleWriter.writerow(['Bonhi, Chowdhury','21','CST'])

27

>>> outputFile.close()

**The delimiter and lineterminator Keyword Arguments**

>>> csvfile=open('./Education/example.csv','w',newline='')

>>> csvwriter=csv.writer(csvfile)

>>> cdvwriter=csv.writer(csvfile,delimiter='\t',lineterminator='\n\n')

>>> cdvwriter.writerow(['4/5/2015 13:34', 'Apples', '73'])

26

>>> cdvwriter.writerow(['4/5/2015 3:41', 'Cherries', '85'])

27

>>> cdvwriter.writerow(['4/6/2015 12:46', 'Pears', '14'])

25

>>> csvfile.close()

**JSON and APIs**

**The json Module**

**Reading JSON with the loads() Function**

>>> jsonData='{"name": "Zophie", "isCat": true, "miceCaught": 0,"felineIQ": null}'

>>> jsonload=json.loads(jsonData)

>>> jsonload

{'name': 'Zophie', 'isCat': True, 'miceCaught': 0, 'felineIQ': None}

>>> jsonload['name']

'Zophie'

**Writing JSON with the dumps() Function**

>>> pythonValues={'Name':'Sagnik','Age':25,'Department':'Information Technology'}

>>> jsonvalues=json.dumps(pythonValues)

>>> jsonvalues

'{"Name": "Sagnik", "Age": 25, "Department": "Information Technology"}'

**Keeping Time,**

**S c h e d u l i n g Tas k s ,**

**a n d La u n c h i n g P r o g r a m s**

>>> import time

>>> time.time

<built-in function time>

>>> time.time()

1547057070.0719242

**Example**

import time

def product():

prod=1

for i in range(10000):

prod=prod\*i

return prod

startTime=time.time()

result=product()

endTime=time.time()

print('The result is: %s' % result)

print('The time taken is: %s'% (endTime-startTime))

**time.sleep()**

import time

def product():

prod=1

for i in range(1,30):

prod=prod\*i

print(prod)

time.sleep(.1)

return prod

startTime=time.time()

result=product()

endTime=time.time()

time.sleep(2)

print('\n The result is: %s' % result)

time.sleep(2)

print('The time taken is: %s'% (endTime-startTime))

**Rounding Numbers**

>>> now=time.time()

>>> now

1547058092.8396122

>>> round(now)

1547058093

>>> round(now,4)

1547058092.8396

>>> round(now,2)

1547058092.84

**TheDatetime Module**

>>> import datetime

>>> datetime.datetime.now()

datetime.datetime(2015, 2, 27, 11, 10, 49, 55, 53)

>>> dt = datetime.datetime(2015, 10, 21, 16, 29, 0)

>>> dt.year, dt.month, dt.day

(2015, 10, 21)

y >>> dt.hour, dt.minute, dt.second

(16, 29, 0)

**The timedelta Data Type**

>>> delta=datetime.timedelta(days=10,hours=13,minutes=11,seconds=30,microseconds=5)

>>> delta

datetime.timedelta(10, 47490, 5)

>>> str(delta)

'10 days, 13:11:30.000005'

>>> delta.total\_seconds()

911490.000005

>>> now=datetime.datetime.now()

>>> now

datetime.datetime(2019, 1, 10, 0, 25, 55, 229898)

>>> after=datetime.timedelta(days=1000)

>>> now+after

datetime.datetime(2021, 10, 6, 0, 25, 55, 229898)

>>> now-after

datetime.datetime(2016, 4, 15, 0, 25, 55, 229898)

>>> now+(2\*after)

datetime.datetime(2024, 7, 2, 0, 25, 55, 229898)

**Pausing until a specific date**

import datetime

import time

halloween2016 = datetime.datetime(2016, 10, 31, 0, 0, 0)

while datetime.datetime.now() < halloween2016:

time.sleep(1)

**Converting datetime Objects into Strings**

>>> oct21st = datetime.datetime(2015, 10, 21, 16, 29, 0)

>>> oct21st.strftime('%Y/%m/%d %H:%M:%S')

'2015/10/21 16:29:00'

>>> oct21st.strftime('%I:%M %p')

'04:29 PM'

>>> oct21st.strftime("%B of '%y")

"October of '15"

strftime directive Meaning

%Y Year with century, as in '2014'

%y Year without century, '00' to '99' (1970 to 2069)

%m Month as a decimal number, '01' to '12'

%B Full month name, as in 'November'

%b Abbreviated month name, as in 'Nov'

%d Day of the month, '01' to '31'

%j Day of the year, '001' to '366'

%w Day of the week, '0' (Sunday) to '6' (Saturday)

%A Full weekday name, as in 'Monday'

%a Abbreviated weekday name, as in 'Mon'

%H Hour (24-hour clock), '00' to '23'

%I Hour (12-hour clock), '01' to '12'

%M Minute, '00' to '59'

%S Second, '00' to '59'

%p 'AM' or 'PM'

%% Literal '%' character

**Converting Strings into datetime Objects**

>>> datetime.datetime.strptime('October 21, 2015', '%B %d, %Y')

datetime.datetime(2015, 10, 21, 0, 0)

>>> datetime.datetime.strptime('2015/10/21 16:29:00', '%Y/%m/%d %H:%M:%S')

datetime.datetime(2015, 10, 21, 16, 29)

>>> datetime.datetime.strptime("October of '15", "%B of '%y")

datetime.datetime(2015, 10, 1, 0, 0)

>>> datetime.datetime.strptime("November of '63", "%B of '%y")

datetime.datetime(2063, 11, 1, 0, 0)

**Multithreading**

import time,datetime,threading

print('Start of a Program')

def takeNAp():

now=datetime.datetime.now()+datetime.timedelta(seconds=10)

while datetime.datetime.now() < now:

print('\_\_\_\_Waiting\_\_\_')

time.sleep(1)

print('Program Has Started')

threadobject=threading.Thread(target=takeNAp)

threadobject.start()

print('End of Program')

**Passing Arguments to the Thread’s Target Function**

**Seperation**

>>> print('cat','dog',sep='&')

cat&dog

cat&dog

>>> import threading

>>> threadobj=threading.Thread(target=print,args=['Cat','Dog','Cow'],kwargs={'sep':' & '})

>>> threadobj

<Thread(Thread-2, initial)>

>>> threadobj.start()

Cat

>>> & Dog & Cow

**Launching Other Programs from Python**

>>> import subprocess

>>> subprocess.Popen('/usr/bin/io.elementary.calculator')

<subprocess.Popen object at 0x7f2cfa2c9278>

>>> calcProc = subprocess.Popen('c:\\Windows\\System32\\calc.exe')

v >>> calcProc.poll() == None

True

w >>> calcProc.wait()

0

>>> calcProc.poll()

0

**Passing Command Line Arguments to Popen()**

>>> subprocess.Popen(['/usr/bin/io.elementary.code','./exampletext.txt'])

<subprocess.Popen object at 0x7f2cf54182b0>

**Running Other Python Scripts**

>>> subprocess.Popen(['/usr/bin/idle','./tutorial.py'])

<subprocess.Popen object at 0x7f2cf54186a0>

**Opening File in Default Application**

>>> open('./exampletext.txt','w')

<\_io.TextIOWrapper name='./exampletext.txt' mode='w' encoding='UTF-8'>

>>> filr=open('./exampletext.txt','w')

>>> filr.write('Hello World I am using Python!')

30

>>> filr.close()

>>> subprocess.Popen(['start','./exampletext.txt'],shell=True)

<subprocess.Popen object at 0x7f2cf542ba20>