# Python Notes

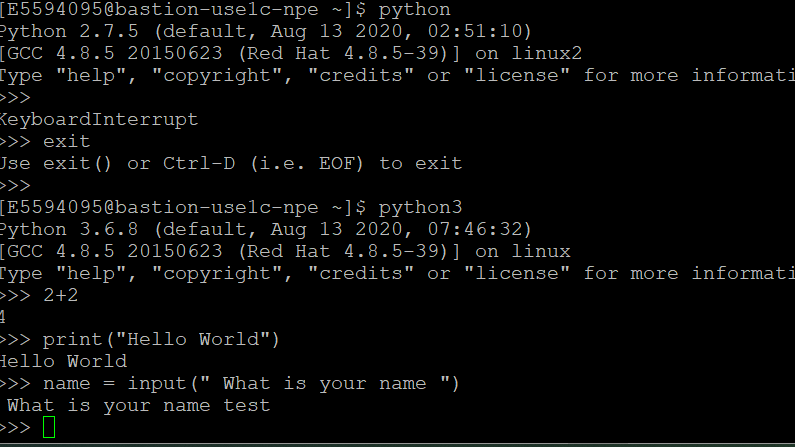
Python's interpreted mode in the command line interface

Python3

>>

Python, like JavaScript, is an interpreted language, in other words, it's not like Java or C or C++, the application does not need to be compiled and built into an executable before you can run it

Python interpreter simply takes each line of code as it comes across it and interprets and execute it



Define main

def main():

Check if main is present or not

print("Hello world")

if \_\_name\_\_ == "\_\_main\_\_" :

main()

Call main

## Varables and expressions:

Comments are declared with hash

# declare a varaibles

test1="0"

print (test1)

#Re-declaring of var works

test1="abc"

print (test1)

# error : diff type var cannot concanate

print ("test 1 is a var " + 123)

PS D:\Python study> & C:/Python34/python.exe "d:/Python study/variables\_start.py"

0

abc

Traceback (most recent call last):

File "d:/Python study/variables\_start.py", line 8, in <module>

print ("test 1 is a var " + 123)

TypeError: Can't convert 'int' object to str implicitly

PS D:\Python study>

# declare a varaibles

test1="0"

print (test1)

#Re-declaring of var works

test1="abc"

print (test1)

# error : diff type var cannot concanate

print ("test 1 is a var " + str(123))

## Function

#Define a basic func

def functest1():

    print("functiontest1")

def functest1():

    print("functiontest1")

functest1() #1

print(functest1())#2

print(functest1)#3

PS D:\Python study> & C:/Python34/python.exe "d:/Python study/Function\_start.py"

functiontest1 🡪 in this case function was called directly #1

functiontest1 🡪 in this case function was called within print function

None 🡪 in this case our function does not have return value , whereas in this case func is not even executed , in below it prints the value of func itself

<function functest1 at 0x023134B0>

PS D:\Python study>

Thus the above example signifies function itself is a object

## Condition

def  main():

    x,y= 10,100

    #Conditional flow uses if ,elif,else

    if (x<y):

        st =" x is less than y"

    print(st)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

PS D:\Python study> & C:/Python34/python.exe "d:/Python study/condition\_start.py"

x is less than y

PS D:\Python study>

def  main():

    x,y= 1000,100

    #Conditional flow uses if ,elif,else

    if (x<y):

        st =" x is less than y"

    print(st)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

PS D:\Python study> & C:/Python34/python.exe "d:/Python study/condition\_start.py"

Traceback (most recent call last):

File "d:/Python study/condition\_start.py", line 8, in <module>

main()

File "d:/Python study/condition\_start.py", line 6, in main

print(st)

UnboundLocalError: local variable 'st' referenced before assignment

Because st was referenced before it was assigned

def  main():

    x,y= 1000,100

    #Conditional flow uses if ,elif,else

    if (x<y):

        st =" x is less than y"

    else:

        st ="x is greter than y"

    print(st)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Some other languages have this notion of a compound conditional known as switch and case. And the answer is no. Python, in it's efforts to be simple, sticks to the if else construct, and uses elif as a substitute for writing a switch case block. So if you're familiar with other programming languages and wondering if there's the equivalent of that, then in Python, as of right now, there's not

def  main():

    x,y= 100,100

    #Conditional flow uses if ,elif,else

    if (x<y):

        st =" x is less than y"

    elif(x==y):

        st =" x is same as  y"

    else:

        st ="x is greater than y"

    print(st)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Python has a construct called a conditional statement. The conditional statement lets you write a common else if construct all in one line, and it's a more concise way of writing the comparison logic

def  main():

    x,y= 100,100

    #Conditional flow uses if ,elif,else

    if (x<y):

        st =" x is less than y"

    elif(x==y):

        st =" x is same as  y"

    else:

        st ="x is greter than y"

    print(st)

    st="x is in conditional statement " if (x<y) else "x is greater thn or same"

    print(st)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

## Loops

Repeating code over and over again, via a construct known as a loop

### While loop

To write a while loop, I use the keyword while, and then I give it a condition. So, I'll write while x is less than five and then my colon. And again you can see that the code is being indented under the colon there. And I'll write print x and then x equals x plus one. A while loop executes a block of code while a particular condition evaluates true. So, while x is less than five, we're going to print the value of x, and then we're going to increment x by one. Some languages, like C, provide a whole bunch of ways of doing loops. Python likes to keep things simple. It's only got two ways of doing loops, while and for.

def main():

    x=0

# define a while loop

    while (x<5):

        print(str(x) +"in a while")

        x=x+1

if \_\_name\_\_ =="\_\_main\_\_":

    main()

PS D:\Python study> & C:/Python34/python.exe "d:/Python study/Loops\_start.py"

0in a while

1in a while

2in a while

3in a while

4in a while

### For loop

Alright, so now let's try a for loop. So, I'll write for x in range five comma 10. So, for loops in Python work a little bit differently than you might be used to in other languages. For example, in JavaScript you have the concept of an index counter, that controls the number of iterations in a for loop. For example, you might have something like, for, and then i equals zero, and then i is less than some condition, right? And then i plus plus. What you have is this loop counter variable that's controlling the execution of the for loop.

So Python works a little bit differently. Python's for loops are what are called iterators. In this case, I want to have x loop over a range of numbers. I used the range function to give me that range of numbers. So, I've got a range going from five to 10, and I'm going to print x each time.

def main():

    x=0

# Define afor loop

    for x in range(5,10):

        print(x)

if \_\_name\_\_ =="\_\_main\_\_":

    main()

But for loops operate over sets of things, not just numbers.

def main():

    x=0

#use for loop over a collection

    days= ["mon","Tue", "wed", "thu","fri","sat","sun"]

    for d in days:

        print(d)

if \_\_name\_\_ =="\_\_main\_\_":

    main()

So, it's looping over the contents of the list

PS D:\Python study> & C:/Python34/python.exe "d:/Python study/Loops\_start.py"

mon

Tue

wed

thu

fri

sat

sun

PS D:\Python study>

### Break and continue

def main():

    x=0

#use Break and continue statement

    days= ["mon","Tue", "wed", "thu","fri","sat","sun"]

    for d in days:

        if (d=="wed"):break

        print(d)

if \_\_name\_\_ =="\_\_main\_\_":

    main()

PS D:\Python study> & C:/Python34/python.exe "d:/Python study/Loops\_start.py"

mon

Tue

PS D:\Python study>

The break statement is used to break the execution of a loop, if a condition is met. The break statement will cause this for loop to terminate and fall through to the next block of code. In this case that's the end of the function. There's nothing else to execute.

The continue statement skips the rest of the loop, for that particular iteration

Continue basically means skip the rest of the execution of this loop. So, just go back up to the top of the loop, and start with the next value. So, don't do this statement right here that's in the loop

    for x in range(5,10):

        if (x % 2 == 0): continue

        print(x)

The enumerate function will iterate over this collection like loop normally would, but in addition to returning the value of the item being looked at, it also returns a value that is the index of the item in question. This function is going to return two values. It's going to return the index of the member of the collection that we're looking at, as well as the actual member of the collection.

def main():

    x=0

#use for enumerate to get index

    days= ["mon","Tue", "wed", "thu","fri","sat","sun"]

    for i,d in enumerate(days):

        print(i ,d + "only for")

if \_\_name\_\_ =="\_\_main\_\_":

    main()

PS D:\Python study> & C:/Python34/python.exe "d:/Python study/Loops\_start.py"

0 mononly for

1 Tueonly for

2 wedonly for

3 thuonly for

4 frionly for

5 satonly for

6 sunonly for

PS D:\Python study>

## Classes

Classes are a really great way of encapsulating functionality that can be kept together and passed around as a complete module for use in other projects Classes are defined using the class keyword and they're given a name and if this class was based on a superclass that I was inheriting from, I would put the name of that other class

Inside a class, I can define functions or in object oriented terminology, they are called methods and they are part of this class.

Usually the first argument to any of the methods of a class, is the self-argument and the self-argument refers to the object itself. It's kind of like this keyword in JavaScript.

# Example of Classes

Define class and its methods, with self-argument

class testclass():

    def testclassmethod1 (self):

        print("testclassmethod1")

    def testclassmethod2 (self,somestr):

        print("testclassmethod2"+somestr)

Obj of class defined

def main():

Call method 1 without self arg which is taken care python runtime

    test = testclass()self

    test.testclassmethod1()

    test.testclassmethod2("strings jksadk")

Method 2 wid args

    print("Hello world main ")

if \_\_name\_\_ == "\_\_main\_\_" :

  main()

Class1 method 1 is called under class 2 method2

# Example of Classes

class testclass():

    def testclassmethod1 (self):

        print("testclassmethod1")

    def testclassmethod2 (self,somestr):

Class 1 method 1 is called in class2 method 1.

        print("testclassmethod2"+somestr)

class testclass2(testclass):

    def testclass2method1 (self):

        testclass.testclassmethod1(self)

        print("testclassmethod1")

    def testclass2method2 (self,somestr):

        print("testclassmethod2")

def main():

    test = testclass()

    test2 = testclass2()

    test2.testclass2method1()

    test2.testclass2method2("tet")

    test.testclassmethod1()

    test.testclassmethod2("strings jksadk")

    print("Hello world main ")

if \_\_name\_\_ == "\_\_main\_\_" :

  main()

method two is not calling the base class. It's not calling the function that's up here in myClass. It simply overrides the existing method two that's in the base class but does it's own thing.

## Importing and using modules

One of the most powerful features of the Python ecosystem is the large number of library modules of pre-build code that you can use in your programs. And the standard Python installation contains quite a few of these modules

# import the math module, which contains features for working with mathematics

import math

# the math module contains lots of pre-built functions

print("The square root of 16 is", math.sqrt(16))

# in addition to functions, some modules contain useful constants

print("Pi is:", math.pi)

# try some of the math functions for yourself here:

## The date, time, and datetime classes

from datetime import date

from datetime import time

from datetime import datetime

def main():

    today=date.today()

    print("today date" , today )

    print ("Date components: ", today.day , today.month , today.year )

    print ("weeday ", today.weekday())

    days = ["mon", "tues", "wed", "thurs","fri", "sat","sun"]

    print("day is ", days[today.weekday()])

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Python> & C:/Users/Dell/AppData/Local/Programs/Python/Python39/python.exe d:/Python/Python\_notes/date\_start.py

today date 2021-04-24

Date components: 24 4 2021

weeday 5

day is sat

PS D:\Python>

Import the date, time, and datetime classes. These are predefined pieces of functionality in the Python library that let me manipulate dates and times.

from datetime import date

from datetime import time

from datetime import datetime

def main():

    todaydatetime = datetime.now()

    print("todat time and date" , todaydatetime.today())

    print("time" , todaydatetime.time())

if \_\_name\_\_ == "\_\_main\_\_":

    main()

### Formatting time output

To format date information you use the stirftime function which is available as a method on the datetime object that is returned by the now function.

This function takes a string argument that contains one or more of these codes as placeholders. So they function as placeholders for date and time data

Times and dates can be formatted using a set of predefined string

%y/%Y - Year, %a/%A - weekday, %b/%B - month, %d - day of month

%c - locale's date and time, %x - locale's date, %X - locale's time

%I/%H - 12/24 Hour, %M - minute, %S - second, %p - locale's AM/PM

For example, to format a string with the full year number I'd use a string like this The current year is: and then %Y Just cut that and put it down here in the formatting section.

#Formatting time output

from datetime import datetime

def main():

    todaydatetime = datetime.now()

    print("todat time and date" , todaydatetime.today())

    print("time" , todaydatetime.time())

    print(todaydatetime.strftime("the year is :%y"))

    print(todaydatetime.strftime("the year is :%a ,%d ,%b, %y,%X ,%p"))

if \_\_name\_\_ == "\_\_main\_\_":

    main()

PS D:\Python> & C:/Users/Dell/AppData/Local/Programs/Python/Python39/python.exe d:/Python/Python\_notes/format\_start.py

todat time and date 2021-04-24 13:23:54.857509

time 13:23:54.857509

the year is :21

the year is :Sat ,24 ,Apr, 21,13:23:54 ,PM

PS D:\Python>

## Using timedelta objects

Given a particular date, you might want to calculate a date in the future or the past relative to that date. We can use the time delta class in Python

A timedelta is basically a span of time. It's not a particular date, it's not a particular time, and it’s a span of time.

#

# Example file for working with timedelta objects

#

from datetime import date

from datetime import time

from datetime import datetime

from datetime import timedelta

# construct a basic timedelta and print it

print(timedelta(days=365,hours=5,minutes=10))

# print today's date

timedate =datetime.now()

print("todays date "+str(timedate))

# print today's date one year from now

print("one year from now",timedate+ timedelta(days=365))

# create a timedelta that uses more than one argument

print("2 days from now",timedate+ timedelta(days=2,hours=5,minutes=10))

# calculate the date 1 week ago, formatted as a string

print("1 week from now",timedate - timedelta(days=7,hours=5,minutes=10))

# calculate the date 1 week ago, formatted as a string

weekago= timedate - timedelta(days=7,hours=5,minutes=10)

strweekago = weekago.strftime("%A %B %d %y")

print("1 week from now",strweekago)

#

# Example file for working with timedelta objects

#

from datetime import date

from datetime import time

from datetime import datetime

from datetime import timedelta

# construct a basic timedelta and print it

print(timedelta(days=365,hours=5,minutes=10))

# print today's date

timedate =datetime.now()

print("todays date "+str(timedate))

# print today's date one year from now

print("one year from now",timedate+ timedelta(days=365))

# create a timedelta that uses more than one argument

print("2 days from now",timedate+ timedelta(days=2,hours=5,minutes=10))

# calculate the date 1 week ago, formatted as a string

print("1 week from now",timedate - timedelta(days=7,hours=5,minutes=10))

weekago= timedate - timedelta(days=7,hours=5,minutes=10)

strweekago = weekago.strftime("%A %B %d %y")

print("1 week from now",strweekago)

### How many days until April Fools' Day?

aprilfd= datetime(timedate.year,4,1)

print(aprilfd)

# use date comparison to see if April Fool's has already gone for this year

# if it has, use the replace() function to get the date for next year

if (aprilfd < timedate):

    print( "april fool day has passed" )

    aprilfd = aprilfd.replace(aprilfd.year +1)

    print ("new april fool day", aprilfd)

else:

     print( "april fool day has  not passed" )

     print ("new april fool day", aprilfd)

# Now calculate the amount of time until April Fool's Day

daysleft= aprilfd -timedate

print(daysleft.days ,"days left ")

PS D:\Python> & C:/Users/Dell/AppData/Local/Programs/Python/Python39/python.exe d:/Python/Python\_notes/timedeltas\_start.py

365 days, 5:10:00

todays date 2021-04-24 14:37:49.308391

one year from now 2022-04-24 14:37:49.308391

2 days from now 2021-04-26 19:47:49.308391

1 week from now 2021-04-17 09:27:49.308391

1 week from now Saturday April 17 21

2021-04-01 00:00:00

april fool day has passed

new april fool day 2022-04-01 00:00:00

341 days left

## Working with calendars

Python's library provides a couple of useful utilities for working with calendars in both text and HTML formats

Text format:-

# import the calendar module

import calendar

# create a plain text calendar

textcal = calendar.TextCalendar(calendar.SUNDAY)

St = textcal.formatmonth(2021, 3, 4 ,0)

print(St)

PS D:\Python> & C:/Users/Dell/AppData/Local/Programs/Python/Python39/python.exe d:/Python/Python\_notes/calendars\_start.py

March 2021

Sun Mon Tue Wed Thu Fri Sat

1 2 3 4 5 6

7 8 9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26 27

28 29 30 31

HTML Calendar:-

# import the calendar module

import calendar

# create an HTML formatted calendar

Htmlcalc = calendar.HTMLCalendar(calendar.MONDAY)

ht = Htmlcalc.formatmonth(2021,8)

print(ht)

<table border="0" cellpadding="0" cellspacing="0" class="month">

<tr><th colspan="7" class="month">August 2021</th></tr>

<tr><th class="mon">Mon</th><th class="tue">Tue</th><th class="wed">Wed</th><th class="thu">Thu</th><th class="fri">Fri</th><th class="sat">Sat</th><th class="sun">Sun</th></tr>

<tr><td class="noday">&nbsp;</td><td class="noday">&nbsp;</td><td class="noday">&nbsp;</td><td class="noday">&nbsp;</td><td class="noday">&nbsp;</td><td class="noday">&nbsp;</td><td class="sun">1</td></tr>

<tr><td class="mon">2</td><td class="tue">3</td><td class="wed">4</td><td class="thu">5</td><td class="fri">6</td><td class="sat">7</td><td class="sun">8</td></tr>

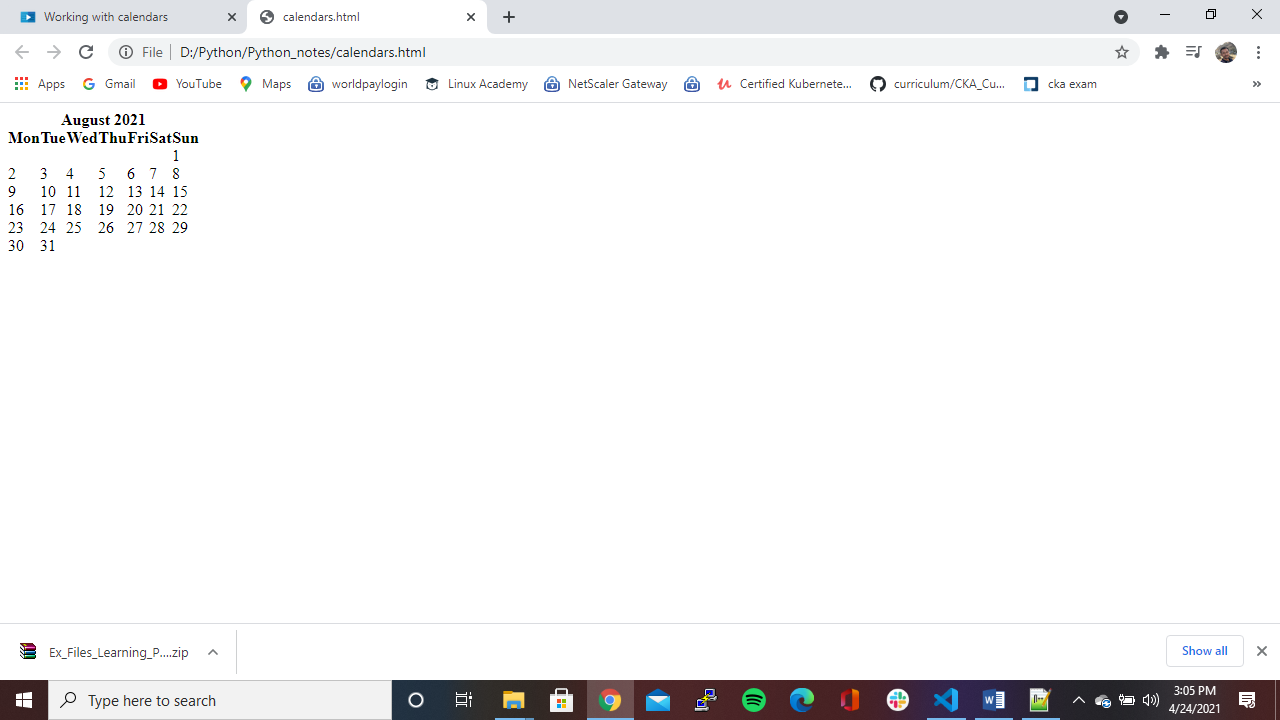
<tr><td class="mon">9</td><td class="tue">10</td><td class="wed">11</td><td class="thu">12</td><td class="fri">13</td><td class="sat">14</td><td class="sun">15</td></tr>

<tr><td class="mon">16</td><td class="tue">17</td><td class="wed">18</td><td class="thu">19</td><td class="fri">20</td><td class="sat">21</td><td class="sun">22</td></tr>

<tr><td class="mon">23</td><td class="tue">24</td><td class="wed">25</td><td class="thu">26</td><td class="fri">27</td><td class="sat">28</td><td class="sun">29</td></tr>

<tr><td class="mon">30</td><td class="tue">31</td><td class="noday">&nbsp;</td><td class="noday">&nbsp;</td><td class="noday">&nbsp;</td><td class="noday">&nbsp;</td><td class="noday">&nbsp;</td></tr>

</table>



# loop over the days of a month

# zeroes mean that the day of the week is in an overlapping month

for monthname in calendar.month\_name:

    print(monthname)

for dayname in calendar.day\_name:

    print(dayname)

# Calculate days based on a rule: For example, consider

# a team meeting on the first Friday of every month.

# To figure out what days that would be for each month,

# we can use this script:

print("teams meeting")

for m in range (1,13):

    cal =calendar.monthcalendar (2021, m)

    week1 =cal[0]

    week2 = cal[2]

    if week1[calendar.FRIDAY]!=0:

         meetday=week1[calendar.FRIDAY]

    else:

        meetday=week2[calendar.FRIDAY]

    print("%10s, %2d" %(calendar.month\_name[m], meetday))

## Files

### Reading and writing files

Python already knows about files inherently, so I don't need to import any classes to work with them

#

# Read and write files using the built-in Python file methods

#

def main():

  # Open a file for writing and create it if it doesn't exist

  file1 =open("testfile1.txt", "w+" )

   # write some lines of data to the file

  for i in range(10):

    file1.write("this is test lines  "+str(i)+"\r\n")

  # close the file when done

  file1.close()

  # Open the file for appending text to the end

  file2 =open("testfile1.txt", "a" )

   # write some lines of data to the file

  for i in range(10):

    file2.write("this is test lines  "+str(i)+"\r\n")

  # close the file when done

  file2.close()

  # Open the file back up and read the contents

  file3 =open("testfile1.txt", "r" )

  if file3.mode == 'r':

    contents=file3.read()

    print(contents)

  file3.close()

  #read line by line

  file4 =open("testfile1.txt", "r" )

  if file4.mode == 'r':

    contentline=file4.readlines()

    for x in contentline:

      print(x)

  file4.close()

if \_\_name\_\_ == "\_\_main\_\_":

  main()

### Working with OS path utilities

OS.

This module gives us the ability to work. with Operating System related features.

import os

import platform

from os import path

import datetime

from datetime import date, time, timedelta

import time

def main():

  # Print the name of the OS

  print(os.name)

  print(platform.system(), platform.release())

PS D:\Python> & C:/Users/Dell/AppData/Local/Programs/Python/Python39/python.exe d:/Python/Python\_notes/ospathutils\_start.py

nt

Windows 10

Here, nt == windows &

Posix == linux

#

# Example file for working with os.path module

#

import os

import platform

from os import path

import datetime

from datetime import date, time, timedelta

import time

def main():

  # Check for item existence and type

  print("Item exists: "+ str(path.exists("testfile1.txt")))

  print("Item is dir: "+ str(path.isdir("testfile1.txt")))

  print("Item is file: "+ str(path.isfile("testfile1.txt")))

  # Work with file paths

Item exists: True

Item is dir: False

Item is file: True

 # Get the modification time

  t = time.ctime(path.getmtime("testfile1.txt"))

  print(t)

  print(datetime.datetime.fromtimestamp(path.getmtime("testfile1.txt")))

Sat Apr 24 16:09:59 2021

2021-04-24 16:09:59.687795

PS D:\Python>

 # Calculate how long ago the item was modified

  timeago= datetime.datetime.now() - datetime.datetime.fromtimestamp(path.getmtime("testfile1.txt"))

  print(timeago)

  print(timeago.total\_seconds())

0:56:37.459696

3397.459696

### Using file system shell methods

#

# Example file for working with filesystem shell methods

#

import os

from os import path

import shutil

from shutil import make\_archive

from zipfile import ZipFile

def main():

  # make a duplicate of an existing file

  if path.exists("testfile1.txt"):

    # get the path to the file in the current directory

    src = path.realpath("testfile1.txt")

    print("path exist ",src)

    # let's make a backup copy by appending "bak" to the name

    dest = src + ".bak"

    print("dest path ",dest)

    # copy over the permissions, modification times, and other info

    shutil.copy(src, dest)

    shutil.copystat(src, dest)

    # rename the original file

    os.rename("testfile1.txt","testfile2.txt")

    # now put things into a ZIP archive

  if path.exists("Ex\_Files\_Learning\_Python\_Upd"):

    get the path to the file in the current directory

    srcdir = path.realpath("Ex\_Files\_Learning\_Python\_Upd")

    print("path exist ",srcdir)

    rootdir, filename =path.split(srcdir)

    shutil.make\_archive("archive","zip",rootdir)

    # more fine-grained control over ZIP files

    with ZipFile("zipfile1.zip","w") as newzip:

      newzip.write("D:\Python\Ex\_Files\_Learning\_Python\_Upd\Ex\_Files\_Learning\_Python\_Upd\Exercise Files\Ch2")

      newzip.write("testfile2.txt")

if \_\_name\_\_ == "\_\_main\_\_":

  main()

## Web data

### Fetching internet data

One of the areas where Python really shines is in retrieving and working with data from the internet, such as jayson, xml, and html

Following is the example to get code and read the data

import urllib.request

def main():

    weburl = urllib.request.urlopen("https://www.google.com/")

    print("result code: " + str(weburl.getcode()))

    data = weburl.read()

    print(data)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

### Working with JSON data

The load S function takes a string of JSON and parses it into a native Python object. So once we've done that we can access that object like we would any other Python object.

#

# Example file for parsing and processing JSON

#

import urllib.request

import json

def printResults(data):

  # Use the json module to load the string data into a dictionary

  theJSON = json.loads(data)

  # now we can access the contents of the JSON like any other Python object

  if "title" in theJSON["metadata"]:

    print(theJSON["metadata"]["title"],"json title ")

  # output the number of events, plus the magnitude and each event name

    count =theJSON["metadata"]["count"]

    print(count)

  # for each event, print the place where it occurred

  for i in theJSON["features"]:

    print(i["properties"]["place"])

  print("---------\n")

  # print the events that only have a magnitude greater than 4

  for i in theJSON["features"]:

    if i["properties"]["mag"] >= 4.0:

      print("%2.1f" % i["properties"]["mag"], i["properties"]["place"])

  print("---------\n")

  # print only the events where at least 1 person reported feeling something

  for i in theJSON["features"]:

    feltreports=i["properties"]["felt"]

    if  feltreports != None:

      if i["properties"]["felt"] > 0:

        print("%2.1f" % i["properties"]["felt"], i["properties"]["place"])

  print("---------\n")

def main():

  # define a variable to hold the source URL

  # In this case we'll use the free data feed from the USGS

  # This feed lists all earthquakes for the last day larger than Mag 2.5

  urlData = "http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/2.5\_day.geojson"

  # Open the URL and read the data

  webUrl = urllib.request.urlopen(urlData)

  urlcode= webUrl.getcode()

  print ("result code: " + str(urlcode))

  if (urlcode == 200):

    data=webUrl.read()

    #print(data)

  else:

    print("recived error"+urlcode)

  printResults(data)

if \_\_name\_\_ == "\_\_main\_\_":

  main()

PS D:\Python> & C:/Users/Dell/AppData/Local/Programs/Python/Python39/python.exe d:/Python/Python\_notes/jsondata\_start.py

result code: 200

USGS Magnitude 2.5+ Earthquakes, Past Day json title

34

Gulf of Alaska

5 km SE of La Parguera, Puerto Rico

47 km W of Mentone, Texas

40 km NW of Elfin Cove, Alaska

Rat Islands, Aleutian Islands, Alaska

8 km WSW of Guánica, Puerto Rico

131 km SSE of Amahai, Indonesia

35 km NNE of Benton, California

3 km E of Pole Ojea, Puerto Rico

Puerto Rico region

4 km SE of La Parguera, Puerto Rico

30km SSW of Qualeys Camp, NV

63 km SW of Santiago de Cao, Peru

31 km SE of Mina, Nevada

107 km NNW of Davila, Philippines

149 km SE of Kuril’sk, Russia

109 km NE of Iquique, Chile

90 km ESE of Adak, Alaska

48 km NNE of Chickaloon, Alaska

northern Mid-Atlantic Ridge

223 km WNW of Pangai, Tonga

112 km SSE of Sand Point, Alaska

38 km NW of Elfin Cove, Alaska

76 km SSW of Isangel, Vanuatu

262 km WNW of Haveluloto, Tonga

109 km ESE of Sarangani, Philippines

central Mid-Atlantic Ridge

14 km N of Downs, Kansas

123 km E of Sarangani, Philippines

South Shetland Islands

5 km SE of La Parguera, Puerto Rico

65 km S of San Francisco del Mar Viejo, Mexico

Izu Islands, Japan region

South Shetland Islands

---------

5.0 Rat Islands, Aleutian Islands, Alaska

4.8 131 km SSE of Amahai, Indonesia

4.6 63 km SW of Santiago de Cao, Peru

4.5 107 km NNW of Davila, Philippines

4.3 149 km SE of Kuril’sk, Russia

4.2 109 km NE of Iquique, Chile

5.0 northern Mid-Atlantic Ridge

6.5 223 km WNW of Pangai, Tonga

4.7 76 km SSW of Isangel, Vanuatu

5.1 262 km WNW of Haveluloto, Tonga

4.5 109 km ESE of Sarangani, Philippines

5.0 central Mid-Atlantic Ridge

5.0 123 km E of Sarangani, Philippines

4.8 South Shetland Islands

4.8 65 km S of San Francisco del Mar Viejo, Mexico

4.4 Izu Islands, Japan region

5.0 South Shetland Islands

---------

4.0 63 km SW of Santiago de Cao, Peru

1.0 109 km NE of Iquique, Chile

2.0 48 km NNE of Chickaloon, Alaska

3.0 223 km WNW of Pangai, Tonga

1.0 14 km N of Downs, Kansas

1.0 65 km S of San Francisco del Mar Viejo, Mexico

### Parsing and processing HTML

#

# Example file for parsing and processing HTML

#

from html.parser import HTMLParser

metacount = 0;

class MyHTMLParser(HTMLParser):

  def handle\_comment(self, data):

    print("comment encountred:",data)

    pos =self.getpos()

    print("\tAt line:",pos[0], "position", pos[1])

  def handle\_starttag(self, tag, attrs):

    global metacount

    if tag =="meta":

      metacount += 1

    print("tag encountred:",tag)

    pos =self.getpos()

    print("\tAt line:",pos[0], "position", pos[1])

    if attrs.\_\_len\_\_() > 0:

      print("\t Attributes")

      for a in attrs:

        print("\t",a[0],"=",a[1])

  def handle\_endtag(self, tag):

    print("endtag encountred:",tag)

    pos =self.getpos()

    print("\tAt line:",pos[0], "position", pos[1])

  def handle\_data(self, data):

    if data.isspace():

      return

    print("data encountred:",data)

    pos =self.getpos()

    print("\tAt line:",pos[0], "position", pos[1])

def main():

  # instantiate the parser and feed it some HTML

  parser = MyHTMLParser()

  f=open("samplehtml.html")

  if (f.mode =='r'):

    contents= f.read()

    parser.feed(contents)

if \_\_name\_\_ == "\_\_main\_\_":

  main();

PS D:\Python> & 'C:\Users\Dell\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\Dell\.vscode\extensions\ms-python.python-2021.4.765268190\pythonFiles\lib\python\debugpy\launcher' '62587' '--' 'd:\Python\Python\_notes\htmlparsing\_start.py'

tag encountred: html

At line: 2 position 0

Attributes

lang = en

tag encountred: head

At line: 3 position 2

tag encountred: meta

At line: 4 position 4

Attributes

charset = utf-8

endtag encountred: meta

At line: 4 position 4

tag encountred: title

At line: 5 position 4

data encountred: Sample HTML Document

At line: 5 position 11

endtag encountred: title

At line: 5 position 31

tag encountred: meta

At line: 6 position 4

Attributes

name = description

content = This is a sample HTML file

endtag encountred: meta

At line: 6 position 4

tag encountred: meta

At line: 7 position 4

Attributes

name = author

content = Administrator

endtag encountred: meta

At line: 7 position 4

tag encountred: meta

At line: 8 position 4

Attributes

name = viewport

content = width=device-width; initial-scale=1.0

endtag encountred: meta

At line: 8 position 4

comment encountred: Replace favicon.ico & apple-touch-icon.png in the root of your domain and delete these references

At line: 9 position 4

tag encountred: link

At line: 10 position 4

Attributes

rel = shortcut icon

href = /favicon.ico

endtag encountred: link

At line: 10 position 4

tag encountred: link

At line: 11 position 4

Attributes

rel = apple-touch-icon

href = /apple-touch-icon.png

endtag encountred: link

At line: 11 position 4

endtag encountred: head

At line: 12 position 2

tag encountred: body

At line: 14 position 2

tag encountred: div

At line: 15 position 4

tag encountred: header

At line: 16 position 6

tag encountred: h1

At line: 17 position 8

data encountred: HTML Sample File

At line: 17 position 12

endtag encountred: h1

At line: 17 position 28

endtag encountred: header

At line: 18 position 6

tag encountred: nav

At line: 19 position 6

tag encountred: p

At line: 20 position 8

tag encountred: a

At line: 21 position 10

Attributes

href = /

data encountred: Home

At line: 21 position 22

endtag encountred: a

At line: 21 position 26

endtag encountred: p

At line: 22 position 8

tag encountred: p

At line: 23 position 8

tag encountred: a

At line: 24 position 10

Attributes

href = /contact

data encountred: Contact

At line: 24 position 29

endtag encountred: a

At line: 24 position 36

endtag encountred: p

At line: 25 position 8

endtag encountred: nav

At line: 26 position 6

tag encountred: div

At line: 27 position 6

endtag encountred: div

At line: 29 position 6

tag encountred: footer

At line: 30 position 6

tag encountred: p

At line: 31 position 8

data encountred: © Copyright by Administrator

At line: 31 position 11

endtag encountred: p

At line: 31 position 44

endtag encountred: footer

At line: 32 position 6

endtag encountred: div

At line: 33 position 4

endtag encountred: body

At line: 34 position 2

endtag encountred: html

At line: 35 position 0

PS D:\Python>

### Manipulating XML

miniDOM class that Python provides, to load an XML file and then operate on the document while it's in memory

#

# Example file for parsing and processing XML

#

import xml.dom.minidom

def main():

  # use the parse() function to load and parse an XML file

  doc = xml.dom.minidom.parse("samplexml.xml")

  # print out the document node and the name of the first child tag

  print(doc.nodeName)

  print (doc.firstChild.tagName)

  # get a list of XML tags from the document and print each one

  skills = doc.getElementsByTagName("skill")

  print("%d skills"%skills.length)

  for skill in skills:

    print(skill.getAttribute("name"))

  # create a new XML tag and add it into the document

  newskill = doc.createElement("skill")

  newskill.setAttribute("name", "jquery")

  doc.firstChild.appendChild(newskill)

  skills = doc.getElementsByTagName("skill")

  print("%d skills"%skills.length)

  for skill in skills:

    print(skill.getAttribute("name"))

if \_\_name\_\_ == "\_\_main\_\_":

  main()

PS D:\Python> & 'C:\Users\Dell\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\Dell\.vscode\extensions\ms-python.python-2021.4.765268190\pythonFiles\lib\python\debugpy\launcher' '62822' '--' 'd:\Python\Python\_notes\xmlparsing\_start.py'

#document

person

4 skills

JavaScript

Python

C#

HTML

5 skills

JavaScript

Python

C#

HTML

jquery

PS D:\Python>