# Python

# Anaconda SSL error

conda config --set ssl\_verify false

# Linux basic commands

find ./GFG -name sample.txt

# Os

## Assure path exist

def assure\_path\_exists(path):

dir = os.path.dirname(path)

if not os.path.exists(dir):

os.makedirs(dir)

# convert python



## .py to .c

* Install cython

pip install cython or pip3 install cython

* Create file > test.py

print("Convert python to cython")

* In cmd

>cython test.py

* It will create a test.c file

## .py to .pyc

* Create file > test.py

print("Convert python to cython")

* In cmd
* > python
* >>> import py\_compile
* >>> py\_compile.compile(‘test.py’)
* This will create test.pyc file

## .py to .pyx

* Create Pythagoras.py

import time

def count\_triples(limit):

result = 0

for a in range(1, limit + 1):

for b in range(a + 1, limit + 1):

for c in range (b + 1, limit + 1):

if c \* c > a \* a + b \* b:

break

if c \* c == (a \* a + b \* b):

result += 1

return result

if \_\_name\_\_ == '\_\_main\_\_':

start = time.time()

result = count\_triples(1000)

duration = time.time() - start

print(result, duration)

* Python Pythagoras.py
  + 881 9.497968435287476
* Create pyth\_triples.pyx

def count\_triples(limit):

result = 0

for a in range(1, limit + 1):

for b in range(a + 1, limit + 1):

for c in range (b + 1, limit + 1):

if c \* c > a \* a + b \* b:

break

if c \* c == (a \* a + b \* b):

result += 1

return result

* In cmd
* pip install cython
* create main.py

import time

import pyximport; pyximport.install()

import pyth\_triples

def main():

start = time.time()

result = pyth\_triples.count\_triples(1000)

duration = time.time() - start

print(result, duration)

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

main()

* python main.py
  + 881 6.446093320846558
* Create file setup.py

from distutils.core import setup

from Cython.Build import cythonize

setup(

ext\_modules = cythonize("pyth\_triples.pyx")

)

* python setup.py build\_ext –inplace
* it will create pyth\_triples.c, pyth\_triples.cp36-win\_amd64.pyd and a build folder
* edit main.py

import time

# import pyximport; pyximport.install()

import pyth\_triples

def main():

start = time.time()

result = pyth\_triples.count\_triples(1000)

duration = time.time() - start

print(result, duration)

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

main()

* python main.py
  + 881 6.341957330703735
* Edit pyth\_triples.pyx as

### Cdef declarations

list

cdef list foo = []

string

cdef char\* c\_string = NULL

float

cdef float x = 5.0

file

cdef FILE\* p

cdef int x,y,z

cdef char \*s

cdef float x = 5.2 (single precision)

cdef double x = 40.5 (double precision)

cdef list languages

cdef dict abc\_dict

cdef object thing

def count\_triples(limit):

cdef int result = 0

cdef int a = 0

cdef int b = 0

cdef int c = 0

for a in range(1, limit + 1):

for b in range(a + 1, limit + 1):

for c in range (b + 1, limit + 1):

if c \* c > a \* a + b \* b:

break

if c \* c == (a \* a + b \* b):

result += 1

return result

* Delete or move file > pyth\_triples.c, pyth\_triples.cp36-win\_amd64.pyd
* Python main.py
  + 881 0.03200125694274902
* python setup.py build\_ext –i
* python main.py
* in ubuntu >
* python setup.py build\_ext -i
* It will crate an .so file in the same directory

# Debugger

* import pdb
* pdb.pm()

## Jupyter notebook

from IPython.core.debugger import set\_trace

set\_trace()

# pickling

## save

import pickle

# take user input to take the amount of data

number\_of\_data = int(input('Enter the number of data : '))

data = []

# take input of the data

for i in range(number\_of\_data):

raw = input('Enter data '+str(i)+' : ')

data.append(raw)

# open a file, where you ant to store the data

file = open('important', 'wb')

# or

file = open('index.pickle', 'wb')

# dump information to that file

pickle.dump(data, file)

# close the file

file.close()

## load

import pickle

# open a file, where you stored the pickled data

file = open('important', 'rb')

# dump information to that file

data = pickle.load(file)

# close the file

file.close()

print('Showing the pickled data:')

cnt = 0

for item in data:

print('The data ', cnt, ' is : ', item)

cnt += 1



import pdb

def transform(x, y):

pdb.set\_trace()

z = x+ y

z = 5

x = 50

y = 60

pdb.set\_trace()

transform(5, 10)

print('z = ' + str(z))

# Pandas

# Program to create series

import pandas as pd # Import Panda Library

# Program to Create series with scalar values

Data =[1, 3, 4, 5, 6, 2, 9] # Numeric data

# Creating series with default index values

s = pd.Series(Data)

# predefined index values

Index =['a', 'b', 'c', 'd', 'e', 'f', 'g']

# Creating series with predefined index values

si = pd.Series(Data, Index)

## When Data contains Dictionary

# Program to Create Dictionary series

dictionary ={'a':1, 'b':2, 'c':3, 'd':4, 'e':5}

# Creating series of Dictionary type

sd = pd.Series(dictionary)

## When Data contains Ndarray

# Program to Create ndarray series

Data =[[2, 3, 4], [5, 6, 7]] # Defining 2darray

# Creating series of 2darray

snd = pd.Series(Data)

## Creation of DataFrame

# Program to Create DataFrame

import pandas as pd # Import Library

a = pd.DataFrame(Data) # Create DataFrame with Data

## One or more dictionaries

Here, Data can be:

* One or more dictionaries
* One or more Series
* 2D-numpy Ndarray

# Program to Create Data Frame with two dictionaries

dict1 ={'a':1, 'b':2, 'c':3, 'd':4}  # Define Dictionary 1

dict2 ={'a':5, 'b':6, 'c':7, 'd':8, 'e':9} # Define Dictionary 2

Data = {'first':dict1, 'second':dict2} # Define Data with dict1 and dict2

df = pd.DataFrame(Data) # Create DataFrame

## When Data is Series

# Program to create Dataframe of three series

import pandas as pd

s1 = pd.Series([1, 3, 4, 5, 6, 2, 9])        # Define series 1

s2 = pd.Series([1.1, 3.5, 4.7, 5.8, 2.9, 9.3]) # Define series 2

s3 = pd.Series(['a', 'b', 'c', 'd', 'e'])    # Define series 3

Data ={'first':s1, 'second':s2, 'third':s3} # Define Data

dfseries = pd.DataFrame(Data)            # Create DataFrame

## When Data is 2D-numpy ndarray

Note: One constraint has to be maintained while creating DataFrame of 2D arrays – Dimensions of 2D array must be same.

# Program to create DataFrame from 2D array

import pandas as pd # Import Library

d1 =[[2, 3, 4], [5, 6, 7]] # Define 2d array 1

d2 =[[2, 4, 8], [1, 3, 9]] # Define 2d array 2

Data ={'first': d1, 'second': d2} # Define Data

df2d = pd.DataFrame(Data) # Create DataFrame

## Save dataframe

from pandas import DataFrame

cars = {'Brand': ['Honda Civic','Toyota Corolla','Ford Focus','Audi A4'],

'Price': [22000,25000,27000,35000]

}

df = DataFrame(cars, columns= ['Brand', 'Price'])

export\_csv = df.to\_csv (r'C:\Users\Ron\Desktop\export\_dataframe.csv', index = None, header=True) #Don't forget to add '.csv' at the end of the path

print (df)

# Threding

## Simple thread

import threading

import time

def sleeper(n, name):

print('Hi {} sleep 5 sec'.format(name))

time.sleep(n)

print('{} has woken up from sleep'.format(name))

t = threading.Thread(target=sleeper, name = 'thread1', args = (5,'thread1'))

t.start()

print('hello')

# after compleating exicution of thread the later code will exicute

t.join()

print('hello')

## threading cuncurrent

import threading

import time

def sleeper(n, name):

print('Hi {} sleep 5 sec'.format(name))

time.sleep(n)

print('{} has woken up from sleep'.format(name))

threads\_list = []

start = time.time()

for i in range(5):

t = threading.Thread(target = sleeper,

name = "thread{}".format(i),

args = (5, 'thread{}'.format(i)))

threads\_list.append(t)

t.start()

print('{} has started'.format(t.name))

for t in threads\_list:

t.join()

end = time.time()

print('time taken:', end-start)

print('all five threads finished')

normal code

start = time.time()

for i in range(5):

print("iteration %d started"%i)

sleeper(5,i)

end = time.time()

print('time taken:', end-start)

print('all five threads finished')

## Thread with return value

import concurrent.futures

def foo(bar):

print('hello {}'.format(bar))

return 'foo'

with concurrent.futures.ThreadPoolExecutor() as executor:

future = executor.submit(foo, 'world!')

return\_value = future.result()

print(return\_value)

## multithread with return value

import concurrent.futures

import time

URLS = ['http://www.foxnews.com/',

'http://www.cnn.com/',

'http://europe.wsj.com/',

'http://www.bbc.co.uk/',

'http://some-made-up-domain.com/']

# Retrieve a single page and report the URL and contents

def load\_url(url, timeout):

print("start ", url)

time.sleep(5)

return url, timeout

# We can use a with statement to ensure threads are cleaned up promptly

with concurrent.futures.ThreadPoolExecutor(max\_workers=5) as executor:

# Start the load operations and mark each future with its URL

future\_to\_url = {executor.submit(load\_url, url, 60): url for url in URLS}

for future in concurrent.futures.as\_completed(future\_to\_url):

url = future\_to\_url[future]

try:

data = future.result()

except Exception as exc:

print('%r generated an exception: %s' % (url, exc))

else:

print(data)

# database

## Sqlite3

### Create a Table

#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')

print ("Opened database successfully")

conn.execute('''CREATE TABLE COMPANY

(ID INT PRIMARY KEY NOT NULL,

NAME TEXT NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR(50),

SALARY REAL);''')

print ("Table created successfully")

conn.close()

### INSERT Operation

#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')

print ("Opened database successfully")

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \

VALUES (1, 'Paul', 32, 'California', 20000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \

VALUES (2, 'Allen', 25, 'Texas', 15000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \

VALUES (3, 'Teddy', 23, 'Norway', 20000.00 )");

conn.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \

VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )");

conn.commit()

print ("Records created successfully")

conn.close()

### SELECT Operation

#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')

print ("Opened database successfully")

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")

for row in cursor:

print ("ID = ", row[0])

print ("NAME = ", row[1])

print ("ADDRESS = ", row[2])

print ("SALARY = ", row[3], "\n")

print ("Operation done successfully")

conn.close()

### UPDATE Operation

#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')

print ("Opened database successfully")

conn.execute("UPDATE COMPANY set SALARY = 25000.00 where ID = 1")

conn.commit()

print ("Total number of rows updated :", conn.total\_changes)

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")

for row in cursor:

print ("ID = ", row[0])

print ("NAME = ", row[1])

print ("ADDRESS = ", row[2])

print ("SALARY = ", row[3], "\n")

print ("Operation done successfully")

conn.close()

### DELETE Operation

#!/usr/bin/python

import sqlite3

conn = sqlite3.connect('test.db')

print ("Opened database successfully")

conn.execute("DELETE from COMPANY where ID = 2;")

conn.commit()

print ("Total number of rows deleted :", conn.total\_changes)

cursor = conn.execute("SELECT id, name, address, salary from COMPANY")

for row in cursor:

print ("ID = ", row[0])

print ("NAME = ", row[1])

print ("ADDRESS = ", row[2])

print ("SALARY = ", row[3], "\n")

print("Operation done successfully")

conn.close()

# Scheduler

* pip install schedule

import schedule

import time

def job():

print("I'm working...")

schedule.every(10).minutes.do(job)

schedule.every().hour.do(job)

schedule.every().day.at("10:30").do(job)

schedule.every(5).to(10).minutes.do(job)

schedule.every().monday.do(job)

schedule.every().wednesday.at("13:15").do(job)

schedule.every().minute.at(":17").do(job)

while True:

schedule.run\_pending()

time.sleep(1)