**FSDS MAY BATCH 2022(Python Basics 21)**

**Submitted by: Shubham Tiwari**

Q1. Add the current date to the text file today.txt as a string.

Ans: Following is the code:

from datetime import date

today = date.today().strftime('%Y-%m-%d')

with open('today.txt', 'a') as f:

f.write(today + '\n')

The **date.today()** method returns the current date as a **date** object, which is then formatted as a string using the **strftime()** method with the format string **'%Y-%m-%d'**, which represents the date in the format "YYYY-MM-DD".

Then, we open the "today.txt" file in "append" mode using a **with** statement to ensure that the file is properly closed when we're done. We then write the current date string followed by a newline character to the file using the **write()** method. The newline character is added to ensure that each date is written on a new line in the file.

Q2. Read the text file today.txt into the string today\_string.

Ans:

with open('today.txt', 'r') as f:

today\_string = f.read()

This code opens the file "today.txt" in "read" mode using a **with** statement to ensure that the file is properly closed when we're done. It then reads the entire contents of the file into the variable **today\_string** using the **read()** method. The contents of the file are returned as a string, including any newline characters present in the file.

Q3. Parse the date from today\_string.

Ans: To parse the date from **today\_string**, you can use the **datetime** module in Python along with the **strptime()** method to convert the string into a **datetime** object, and then use the **date()** method to extract the date component.

from datetime import datetime

with open('today.txt', 'r') as f:

today\_string = f.read()

today\_date = datetime.strptime(today\_string.strip(), '%Y-%m-%d').date()

print(today\_date)

In this code, we first read the contents of the file "today.txt" into the variable **today\_string**, as shown in the previous answer.

Then, we use the **strptime()** method of the **datetime** module to parse the date from the string. The **strip()** method is used to remove any leading or trailing whitespace from the string, and the format string **'%Y-%m-%d'** is used to match the format of the date string in the file, which is in the format "YYYY-MM-DD". The **strptime()** method returns a **datetime** object.

Finally, we use the **date()** method to extract the date component of the **datetime** object and store it in the variable **today\_date**. We then print the date to verify that it was parsed correctly.

Q4. List the files in your current directory.

Ans:

**import os**

**files = os.listdir('.')**

**print(files)**

In this code, we first import the **os** module, which provides a portable way of using operating system dependent functionality like reading or writing to the file system.

We then call the **listdir()** function from the **os** module, passing it the string **'.'** as an argument, which represents the current directory. This function returns a list of all the files and directories in the current directory.

Finally, we print the list of files to the console using the **print()** function. This will show you the names of all the files in the current directory.

Q5. Create a list of all of the files in your parent directory (minimum five files should be available).

Ans:

import os

parent\_dir = os.path.abspath(os.path.join(os.getcwd(), '..'))

files = os.listdir(parent\_dir)

print(files)

In this code, we first import the **os** module, as in the previous answer.

We then use the **os.getcwd()** function to get the current working directory, and the **os.path.join()** function to join the current working directory with the string **'..'**, which represents the parent directory. We then use the **os.path.abspath()** function to convert the resulting path to an absolute path.

The **listdir()** function is then called on this parent directory path to get a list of all the files and directories in the parent directory.

Finally, we print the list of files to the console using the **print()** function. This will show you the names of all the files in the parent directory.

Q6. Use multiprocessing to create three separate processes. Make each one wait a random number of seconds between one and five, print the current time, and then exit.

Ans:

import multiprocessing

import time

import random

from datetime import datetime

def worker():

wait\_time = random.randint(1, 5)

time.sleep(wait\_time)

print(f"{datetime.now()}: Worker finished after {wait\_time} seconds.")

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

processes = []

for i in range(3):

p = multiprocessing.Process(target=worker)

p.start()

processes.append(p)

for p in processes:

p.join()

In this code, we define a **worker** function that generates a random wait time between one and five seconds using the **random.randint()** function, sleeps for that amount of time using the **time.sleep()** function, prints the current time using the **datetime.now()** function, and then exits.

We then use a **for** loop to create three separate processes, each of which runs the **worker** function as a target using the **multiprocessing.Process** class. We start each process using the **start()** method and store a reference to each process in a list.

Finally, we use another **for** loop to wait for each process to finish using the **join()** method. The **if \_\_name\_\_ == '\_\_main\_\_':** line is required when using the **multiprocessing** module to ensure that the child processes are able to import the necessary modules and execute the **worker** function.

Q7. Create a date object of your day of birth.

Ans:

from datetime import date

my\_birthday = date(1995, 3, 25)

print(my\_birthday)

In this code, we first import the **date** class from the **datetime** module.

We then create a **date** object by calling the **date()** constructor and passing it three arguments: the year (1995), the month (10), and the day (15) of my birthday. We store this object in the **my\_birthday** variable.

Finally, we print the **my\_birthday** variable to the console using the **print()** function. This will show the date object in the format **YYYY-MM-DD**, where **YYYY** is the year, **MM** is the month, and **DD** is the day.

Q8. What day of the week was your day of birth?

Ans:

from datetime import date

my\_birthday = date(1995, 3, 25)

day\_of\_week = my\_birthday.weekday()

weekday\_names = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

print(f"My birthday ({my\_birthday}) was on a {weekday\_names[day\_of\_week]}.")

In this code, we first create the **my\_birthday** variable as in the previous answer.

We then use the **weekday()** method to get the day of the week as an integer between 0 and 6, where 0 is Monday and 6 is Sunday. We store this integer in the **day\_of\_week** variable.

We also create a list of weekday names in the same order as the **weekday()** method returns, where the first element is **'Monday'** and the last element is **'Sunday'**. We store this list in the **weekday\_names** variable.

Finally, we print a message to the console using the **print()** function, where we use string formatting to insert the **my\_birthday** and **weekday\_names[day\_of\_week]** variables into the message. This will show the day of the week of my birthday.

Q9. When will you be (or when were you) 10,000 days old?

Ans: To find out when I will be (or when I was) 10,000 days old, we can use the **date** object and the **timedelta** object from the **datetime** module.

from datetime import date, timedelta

my\_birthday = date(1995, 3, 25)

days\_old = 10000

future\_date = my\_birthday + timedelta(days=days\_old)

print(f"I will be (or was) {days\_old} days old on {future\_date}.")

In this code, we first create the **my\_birthday** variable as in the previous answer.

We also create a **days\_old** variable with the number of days for which we want to calculate the future or past date.

We then create a **timedelta** object using the **timedelta()** constructor and passing it the number of days as a keyword argument (**days=days\_old**).

We add this **timedelta** object to the **my\_birthday** date object using the **+** operator, and store the resulting date object in the **future\_date** variable.

Finally, we print a message to the console using the **print()** function, where we use string formatting to insert the **days\_old** and **future\_date** variables into the message. This will show the date when I will be (or was) 10,000 days old.