Name :- Deepawali . B. Mhaisagar
Assignment no 21

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

ANS
To add the current date to the text file `today.txt` as a string, you can follow these steps:

Step 1: Import the `datetime` module:

```
import datetime
```

Step 2: Get the current date as a string:

```
current_date = datetime.date.today().strftime("%Y-%m-%d")
```

Step 3: Open the `today.txt` file in append mode and write the current date:

```
with open('today.txt', 'a') as file:
    file.write(current_date + '\n')
```

In this code, we use the `datetime` module to get the current date using `datetime.date.today()`. Then, we format the date as a string in the desired format using `.strftime("%Y-%m-%d")`.

After that, we open the `today.txt` file in append mode using a `with` statement, and write the current date to the file by appending it with a newline character `'\n'`.

Make sure you have the file `today.txt` present in the current working directory before running this code. The current date will be appended to the file as a string on a new line each time you run the code.

2. Read the text file today.txt into the string today_string
ANS

To read the contents of the text file `today.txt` into a string named `today_string`, you can follow these steps:

Step 1: Open the `today.txt` file in read mode:

```
with open('today.txt', 'r') as file:
    today_string = file.read()
```

In this code, we use a `with` statement to open the `today.txt` file in read mode. The file content is then read using the `read()` method and assigned to the `today_string` variable.

Make sure you have the file `today.txt` present in the current working directory before running this code. The entire contents of the file will be stored in the `today_string` variable as a string.

3. Parse the date from today_string.
ANS

To parse the date from the `today_string` variable, which contains the contents of the file `today.txt`, you can follow these steps:

Step 1: Import the `datetime` module:

~~~
import datetime
```
~~~

Step 2: Use the `strptime()` function from `datetime` to parse the date from the `today_string`:

~~~
parsed_date = datetime.datetime.strptime(today_string.strip(), "%Y-%m-%d")
```
~~~

In this code, we use the `strptime()` function from the `datetime` module to parse the date from the `today_string` variable. The `strptime()` function converts the string to a `datetime` object.

The `strip()` method is used to remove any leading or trailing whitespace characters from the `today_string` before parsing the date.

The format `%Y-%m-%d` in the `strptime()` function specifies the expected format of the date string. Adjust the format accordingly if the format of your date string is different.

After executing this code, the `parsed_date` variable will contain the parsed date as a `datetime` object. You can then perform various operations on it or format it as desired.

4. List the files in your current directory
ANS

To list the files in the current directory, you can use the `os` module in Python. Here's an example:

```
import os

# Get the current directory
current_directory = os.getcwd()

# List the files in the current directory
file_list = os.listdir(current_directory)

# Print the file names
for file_name in file_list:
    print(file_name)
```

In this code, we use the `os` module to interact with the operating system. The `getcwd()` function retrieves the current working directory. Then, the `listdir()` function is used to obtain a list of file names in the current directory.

Finally, we iterate over the `file_list` and print each file name using a loop. This will display the names of the files present in the current directory.

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

To create a list of all the files in the parent directory, you can use the `os` module in Python. Here's an example:

```
import os

# Get the parent directory
parent_directory = os.path.dirname(os.getcwd())

# List the files in the parent directory
file_list = os.listdir(parent_directory)

# Print the file names
for file_name in file_list:
    print(file_name)
```

In this code, we use the `os` module to interact with the operating system. The `os.getcwd()` function retrieves the current working directory, and the `os.path.dirname()` function obtains the parent directory by extracting the directory name from the current working directory.

Then, the `os.listdir()` function is used to obtain a list of file names in the parent directory. Finally, we iterate over the `file_list` and print each file name using a loop. This will display the names of the files present in the parent directory.

Make sure that there are at least five files available in the parent directory for this code to work as expected.

6. 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

Here's an example of using the `multiprocessing` module in Python to create three separate processes. Each process will wait for a random number of seconds between one and five, print the current time, and then exit:

```
import multiprocessing
import random
import time
from datetime import datetime

def worker():
    wait_time = random.randint(1, 5)
    time.sleep(wait_time)
    current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    print(f"Process {multiprocessing.current_process().name}: Current time -
{current_time}")

if __name__ == '__main__':
    processes = []

    for i in range(3):
        p = multiprocessing.Process(target=worker, name=f"Process-{i+1}")
        p.start()
        processes.append(p)

    for p in processes:
        p.join()
```

In this code, we define a `worker()` function that represents the task performed by each process. Inside the `worker()`, we generate a random wait time between one and five seconds using `random.randint()`. The process then sleeps for the specified time using `time.sleep()`.

After waiting, the current time is obtained using `datetime.now().strftime()` with the desired format. We then print the current time along with the name of the process using `multiprocessing.current_process().name`.

In the main block, we create three separate processes using `multiprocessing.Process()`, assigning each a unique name. We start each process using `p.start()` and add them to a list `processes`.

Finally, we iterate over the list of processes and call `p.join()` to ensure that each process completes before the main program terminates.

When you run this code, each process will execute in parallel, wait for a random time, print the current time, and then exit. The output will show the current time for each process as it completes its task.

7. Create a date object of your day of birth.
ANS

As an AI language model, I don't have personal information or a specific day of birth. However, I can show you an example of creating a `date` object for a specific date.

Here's an example of creating a `date` object for a specific day of birth, such as January 1, 2000:

```
import datetime

# Create a date object for the day of birth
date_of_birth = datetime.date(2000, 1, 1)

# Print the date of birth
print(date_of_birth)
```

In this example, we use the `datetime` module to create a `date` object. The `datetime.date()` function takes the year, month, and day as arguments and returns a `date` object.

Replace the values `2000`, `1`, and `1` in the `datetime.date()` function with your actual birth year, birth month, and birth day respectively.

When you run this code, it will print the date of birth in the format `YYYY-MM-DD`, based on the values provided.

Please ensure you provide the correct values for your own date of birth when creating the `date` object.

8. What day of the week was your day of birth?
ANS

Date as January 1, 2000:

```python
import datetime

# Create a date object for the day of birth
date_of_birth = datetime.date(2000, 1, 1)

# Get the day of the week (0 = Monday, 1 = Tuesday, ..., 6 = Sunday)
day_of_week = date_of_birth.weekday()

# Convert the day of the week to its corresponding name
day_name = date_of_birth.strftime('%A')

# Print the day of the week
print(day_name)
```

In this example, we use the `weekday()` method of the `date` object to get the day of the week for the given date. The `weekday()` method returns an integer from 0 to 6, where 0 represents Monday, 1 represents Tuesday, and so on.

We then use the `strftime()` method to format the `date` object and extract the name of the day (`'%A'` format specifier). This gives us the corresponding name of the day of the week.

Finally, we print the name of the day of the week for the given date.

When you run this code, it will print the day of the week for the provided date, such as "Saturday" for January 1, 2000.

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

As an AI language model, I don't have a personal day of birth or an age. Therefore, I don't have a specific date to calculate when I would be or when I was 10,000 days old.

However, I can provide you with an example of how to calculate the date when someone would be or was 10,000 days old from their birth date.

Here's an example:

```
import datetime

# Define the birth date
birth_date = datetime.date(1990, 1, 1)

# Calculate the target date
target_date = birth_date + datetime.timedelta(days=10000)

# Print the target date
print(target_date)
```

In this example, we start by defining the birth date using the `datetime.date()` function. Replace the values `1990`, `1`, and `1` with the actual birth year, birth month, and birth day, respectively.
We then calculate the target date by adding 10,000 days to the birth date using the `datetime.timedelta()` function. This function allows us to create a time difference in days.
Finally, we print the target date, which represents the date when someone would be or was 10,000 days old.
When you run this code, it will output the target date based on the birth date provided.