**Assignment 21**

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

**You can add the current date to the text file "today.txt" in Python using the following code:**

**python**

**Copy code**

**from datetime import date**

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

**with open("today.txt", "w") as file:**

**file.write(today)**

**In the above code, we first import the date class from the datetime module. Then, we get the current date using the date.today() method and convert it to a string in the format of "YYYY-MM-DD" using the strftime() method.**

**Next, we open the file "today.txt" in write mode using the open() function and write the current date string to the file using the write() method. Finally, we close the file using the with statement.**

**2. Read the text file today.txt into the string today\_string**

**To read the text file "today.txt" into the string today\_string, you can use the following code:**

**python**

**Copy code**

**with open("today.txt", "r") as file:**

**today\_string = file.read()**

**print(today\_string)**

**In the above code, we open the file "today.txt" in read mode using the open() function and assign it to the variable file. Then, we read the contents of the file using the read() method and assign it to the variable today\_string.**

**Finally, we print the contents of today\_string to verify that we have successfully read the file.**

**3. Parse the date from today\_string.**

**To parse the date from the string today\_string, you can use the datetime module in Python. Here's how you can do it:**

**python**

**Copy code**

**from datetime import datetime**

**date\_format = "%Y-%m-%d"**

**parsed\_date = datetime.strptime(today\_string, date\_format).date()**

**print(parsed\_date)**

**In the above code, we import the datetime module and define the date format that matches the format we used to write the date to the file.**

**Then, we use the strptime() method to parse the date from the string today\_string according to the specified format. This method returns a datetime object, which we then convert to a date object using the date() method and assign it to the variable parsed\_date.**

**Finally, we print the contents of parsed\_date to verify that we have successfully parsed the date from the string.**

**4. List the files in your current directory**

**To list the files in the current directory in Python, you can use the os module. Here's how you can do it:**

**python**

**Copy code**

**import os**

**files = os.listdir()**

**for file in files:**

**print(file)**

**In the above code, we import the os module, which provides a way to interact with the operating system. We use the listdir() method to get a list of all the files in the current directory and assign it to the variable files.**

**Then, we loop over the list of files and print each file name using the print() function.**

**When you run this code, it will print the names of all the files in the current directory to the console.**

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

**To create a list of all the files in the parent directory, you can use the os module in Python. Here's how you can do it:**

**python**

**Copy code**

**import os**

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

**files = os.listdir(parent\_dir)**

**for file in files:**

**print(file)**

**In the above code, we first use the os.getcwd() method to get the current working directory and then use the os.pardir constant to get the path to the parent directory. We then use the os.path.join() method to join the current working directory and the parent directory paths and get the absolute path to the parent directory using the os.path.abspath() method.**

**Next, we use the os.listdir() method to get a list of all the files in the parent directory and assign it to the variable files.**

**Finally, we loop over the list of files and print each file name using the print() function.**

**When you run this code, it will print the names of all the files in the parent directory to the console. Note that this will include both files and directories in the parent directory.**

**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.**

**To create three separate processes in Python using the multiprocessing module and make each one wait a random number of seconds between one and five, print the current time, and then exit, you can use the following code:**

**python**

**Copy code**

**import multiprocessing**

**import random**

**import time**

**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):**

**process = multiprocessing.Process(target=worker)**

**process.start()**

**processes.append(process)**

**for process in processes:**

**process.join()**

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

**We then create three separate processes by looping over a range of three, creating a new Process object using the multiprocessing.Process() method, and starting the process using the start() method. We add each process to a list called processes.**

**Finally, we loop over the list of processes using the join() method to wait for each process to finish before exiting the program.**

**When you run this code, it will create three separate processes that run concurrently, each waiting a random amount of time between one and five seconds, printing the current time and the wait time, and then exiting.**

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

**To create a date object of your day of birth in Python, you can use the datetime module. Here's an example:**

**python**

**Copy code**

**from datetime import date**

**my\_birth\_date = date(2000, 5, 10)**

**print(my\_birth\_date)**

**In the above code, we use the date() constructor from the datetime module to create a date object representing our birth date. We pass in the year, month, and day of our birth as arguments to the constructor and assign the resulting date object to the variable my\_birth\_date.**

**Finally, we print the contents of my\_birth\_date to verify that we have successfully created a date object of our birth date. Note that the year, month, and day values used in the example above are just placeholders, and you should replace them with your actual birth year, month, and day to create a date object of your own birth date.**

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

**To find out what day of the week your day of birth was in Python, you can use the weekday() method of the date object. Here's an example:**

**python**

**Copy code**

**from datetime import date**

**my\_birth\_date = date(2000, 5, 10)**

**day\_of\_week = my\_birth\_date.weekday()**

**print(day\_of\_week)**

**In the above code, we first create a date object representing our birth date using the date() constructor from the datetime module. We pass in the year, month, and day of our birth as arguments to the constructor and assign the resulting date object to the variable my\_birth\_date.**

**We then use the weekday() method of the date object to get the day of the week of our birth date as an integer, where Monday is 0 and Sunday is 6. We assign the resulting integer to the variable day\_of\_week.**

**Finally, we print the contents of day\_of\_week to verify the result. To get the day of the week as a string, you can use a list or tuple of weekday names and index into it using the day\_of\_week integer, like this:**

**python**

**Copy code**

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

**day\_of\_week\_name = weekday\_names[day\_of\_week]**

**print(day\_of\_week\_name)**

**This will print the name of the day of the week of your birth date. Note that the year, month, and day values used in the example above are just placeholders, and you should replace them with your actual birth year, month, and day to get the correct result.**

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

**To find out when you will be (or when you were) 10,000 days old, you can use the date object in Python and add 10,000 days to your birth date. Here's an example:**

**python**

**Copy code**

**from datetime import date, timedelta**

**my\_birth\_date = date(2000, 5, 10)**

**days\_to\_add = 10000**

**my\_10000th\_day = my\_birth\_date + timedelta(days=days\_to\_add)**

**print(my\_10000th\_day)**

**In the above code, we first create a date object representing our birth date using the date() constructor from the datetime module. We pass in the year, month, and day of our birth as arguments to the constructor and assign the resulting date object to the variable my\_birth\_date.**

**We then define a variable days\_to\_add with a value of 10,000, which is the number of days we want to add to our birth date.**

**Next, we use the timedelta() function from the datetime module to create a time delta object with a number of days equal to days\_to\_add. We then add this time delta object to our birth date using the + operator and assign the resulting date object to the variable my\_10000th\_day.**

**Finally, we print the contents of my\_10000th\_day to see the date on which we will be (or were) 10,000 days old. Note that the year, month, and day values used in the example above are just placeholders, and you should replace them with your actual birth year, month, and day to get the correct result.**