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

Here's an example of how to add the current date to a text file named `today.txt` as a string using Python:

```python

import datetime

# Get the current date and format it as a string

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

# Open the file in append mode and write the date string to it

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

f.write(today + '\n')

```

This will append the current date to the `today.txt` file on a new line. If the file does not exist, it will be created.

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

Sure, here's how you can read the contents of a file named `today.txt` into a string variable named `today\_string`:

```python

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

today\_string = f.read()

```

This will open the file in read mode (`'r'`) and read its contents into the `today\_string` variable. The `with` statement ensures that the file is automatically closed after reading.

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

To parse a date from a string, we can use the `datetime` module in Python. Here's an example:

```python

import datetime

# assuming today\_string contains a date in the format 'YYYY-MM-DD'

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

```

This code uses the `strptime()` method to convert the `today\_string` to a `datetime` object, and then retrieves the date component using the `date()` method. The second argument to `strptime()` is a format string that specifies how to interpret the date string. In this case, we're assuming that the date is in the format 'YYYY-MM-DD'.

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

You can use the `os` module in Python to list the files in your current directory. Here is an example code snippet that prints all the files and directories in the current working directory:

```python

import os

for filename in os.listdir("."):

print(filename)

```

This will print the name of each file and directory in the current working directory. If you want to filter the results to only show files (and not directories), you can add an if statement to the loop:

```python

import os

for filename in os.listdir("."):

if os.path.isfile(filename):

print(filename)

```

This will only print the names of files in the current directory.

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

**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.**

Here's an example implementation using the `multiprocessing` module to create three separate processes:

```python

import multiprocessing

import time

import random

import datetime

def process():

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

time.sleep(wait\_time)

print(f"Process {multiprocessing.current\_process().name} completed at {datetime.datetime.now()}")

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

for i in range(3):

p = multiprocessing.Process(target=process, name=f"Process-{i+1}")

p.start()

```

In this example, we define a function `process()` that generates a random wait time between one and five seconds, sleeps for that time, and then prints the current time and the name of the process that completed. We then use a `for` loop to create three separate processes, each running the `process()` function. The `name` parameter is used to assign a unique name to each process. Finally, we call the `start()` method to start each process. When each process completes, it will print a message indicating its completion time. Note that the processes may complete in any order due to their random wait times.

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

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

To determine the day of the week of your birth, you can use the `weekday()` method of the `date` object. Here is an example code snippet:

```python

from datetime import date

# Replace these values with your birth year, month, and day

birth\_year = 1990

birth\_month = 4

birth\_day = 15

# Create a date object for your birthdate

birthdate = date(birth\_year, birth\_month, birth\_day)

# Use the weekday() method to get the day of the week

day\_of\_week = birthdate.weekday()

# Print the day of the week (0 = Monday, 6 = Sunday)

print(day\_of\_week)

```

In this example, the code creates a `date` object representing April 15, 1990 (replace the `birth\_year`, `birth\_month`, and `birth\_day` variables with your own values). The `weekday()` method is then used to determine that April 15, 1990 fell on a Sunday, which is represented by the integer value 6.

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

To find out when you will be (or were) 10,000 days old, you can use the `datetime` module in Python. Here's an example code snippet:

```python

import datetime

# Your birth date

birthdate = datetime.date(1990, 6, 1)

# Calculate the date when you will be 10,000 days old

ten\_thousand\_days = datetime.timedelta(days=10000)

ten\_thousand\_days\_old = birthdate + ten\_thousand\_days

print("You will be 10,000 days old on:", ten\_thousand\_days\_old)

```

Replace the `datetime.date(1990, 6, 1)` with your own birth date. This code calculates the date when you will be (or were) 10,000 days old, and prints it to the console.