Assignment 11 – Dates and Times

Date and Time Objects

Retrieving Current Date and Time:

• date.today()	Returns the current date
• datetime.now()	Returns the current date and time

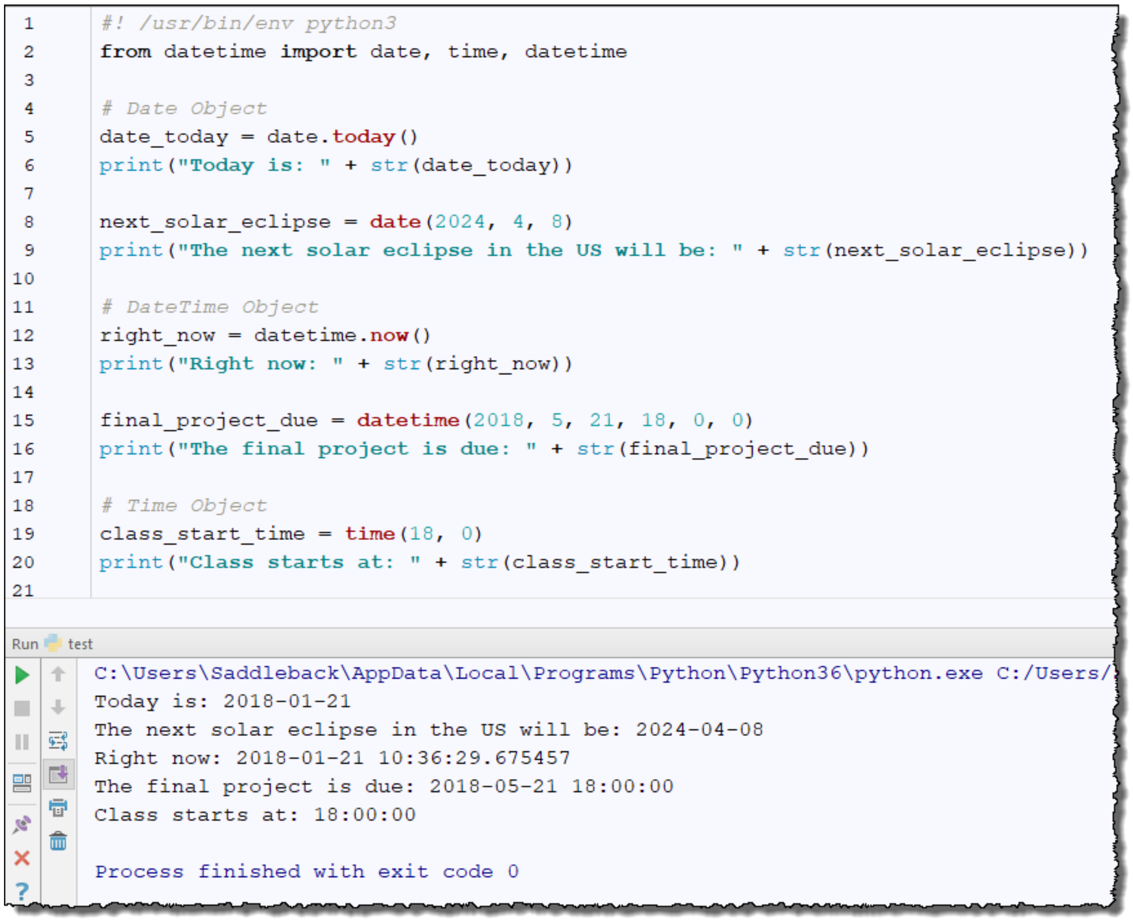
Creating Date and Time Objects:

- date(year, month, day)
- time([hour] [, min] [, sec] [, microsec])
- datetime(year, month, day [hour] [, min] [, sec] [, microsec])

Working with Date Time Objects

1. Code the following:

   Screen Capture #1 (1 points)



```
#! /usr/bin/env python3
from datetime import date, time, datetime

# Date Object
date_today = date.today()
print("Today is: " + str(date_today))

next_solar_eclipse = date(2024, 4, 8)
print("The next solar eclipse in the US will be: " + str(next_solar_eclipse))

# DateTime Object
right_now = datetime.now()
print("Right now: " + str(right_now))

final_project_due = datetime(2018, 5, 21, 18, 0, 0)
print("The final project is due: " + str(final_project_due))

# Time Object
class_start_time = time(18, 0)
print("Class starts at: " + str(class_start_time))

```

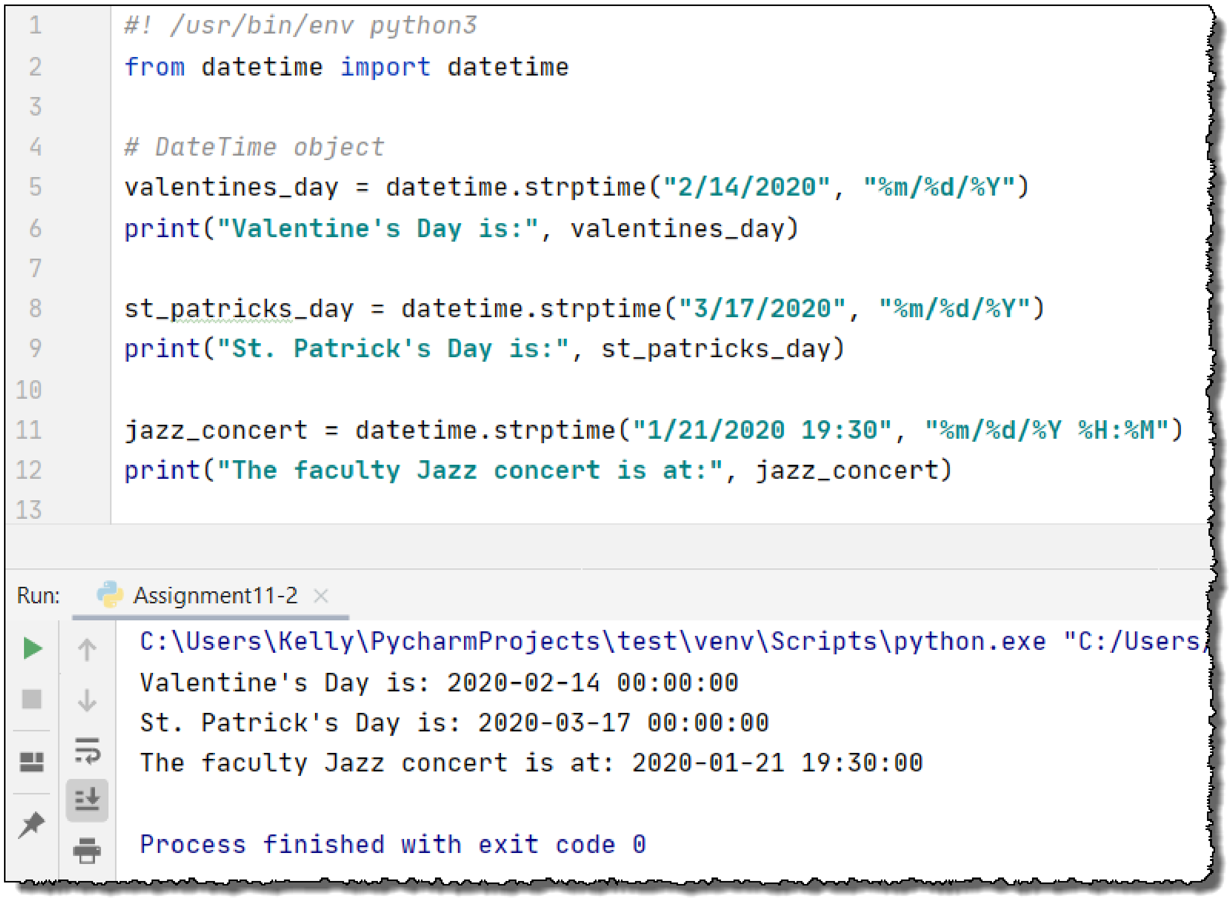
```
C:\Users\Saddleback\AppData\Local\Programs\Python\Python36\python.exe C:/Users/
Today is: 2018-01-21
The next solar eclipse in the US will be: 2024-04-08
Right now: 2018-01-21 10:36:29.675457
The final project is due: 2018-05-21 18:00:00
Class starts at: 18:00:00

Process finished with exit code 0
```

Create Date Time Objects by Parsing Strings

- datetime.strptime(datetime_string, format_string)

2. Code the following:
    a. Note the format for inputting the year
        i. When using the full 4-digit year, use the %Y
        ii. When using the full 2-digit year, use the %y

Screen Capture #2 (1 points)



```python
#! /usr/bin/env python3
from datetime import datetime

# DateTime object
valentines_day = datetime.strptime("2/14/2020", "%m/%d/%Y")
print("Valentine's Day is:", valentines_day)

st_patricks_day = datetime.strptime("3/17/2020", "%m/%d/%Y")
print("St. Patrick's Day is:", st_patricks_day)

jazz_concert = datetime.strptime("1/21/2020 19:30", "%m/%d/%Y %H:%M")
print("The faculty Jazz concert is at:", jazz_concert)

```

```
C:\Users\Kelly\PycharmProjects\test\venv\Scripts\python.exe "C:/Users
Valentine's Day is: 2020-02-14 00:00:00
St. Patrick's Day is: 2020-03-17 00:00:00
The faculty Jazz concert is at: 2020-01-21 19:30:00

Process finished with exit code 0
```

Formatting Date and Time

- datetime.strftime(format_string)

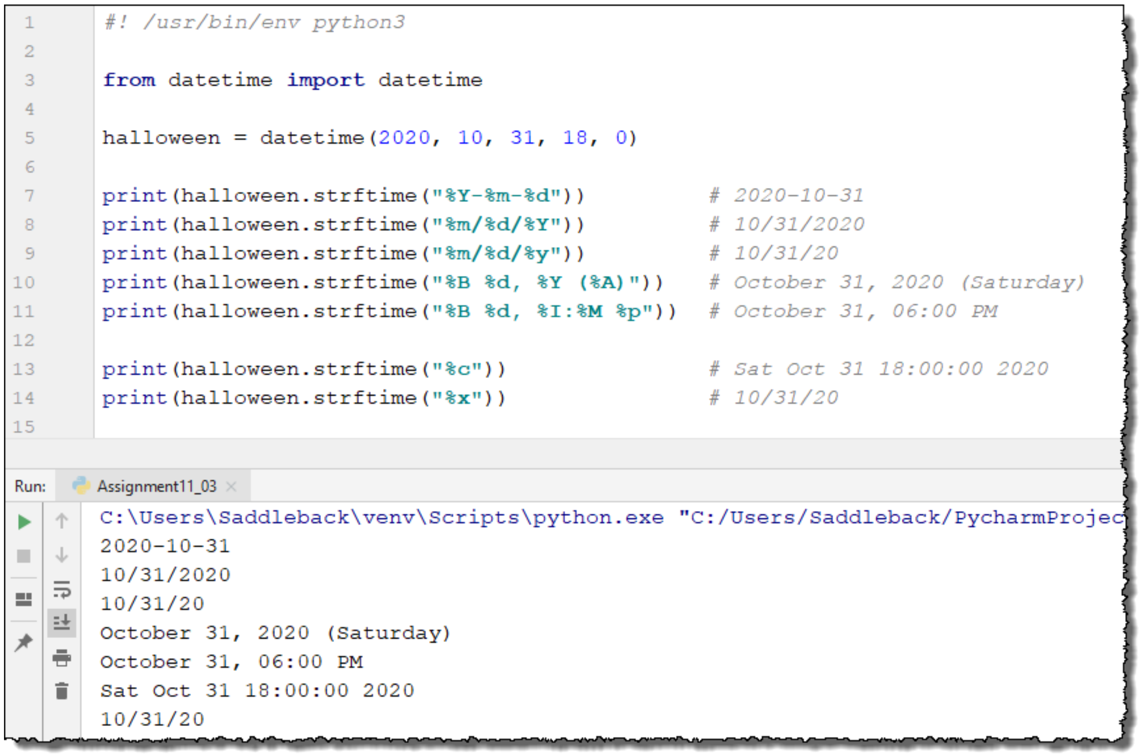
  Formatting Codes:

Code	Description	Example
%a	Abbreviated weekday name	Sat
%A	Full weekday name	Saturday
%b	Abbreviated month name	Jan
%B	Full month name	January
%d	Zero-padded day of month as number	01
%m	Zero-padded month as number	01
%Y	4-digit year	2018
%y	2-digit year	18
%H	Hour of day in 24-hour format	20
%I	Hour of day in 12-hour format	04
%M	Minute as number	30
%S	Second as number	45
%p	AM/PM specifier	AM
%f	Microsecond	232398

  Locale Formatting Codes:

Code	Description	Example
%c	Date and time formatted for locale	Sat Jan 20 01:15:30 2018
%x	Date formatted for locale	01/20/2018
%X	Time formatted for locale	01:15:30

3. Code the following:
   Screen Capture #3 (1 points)



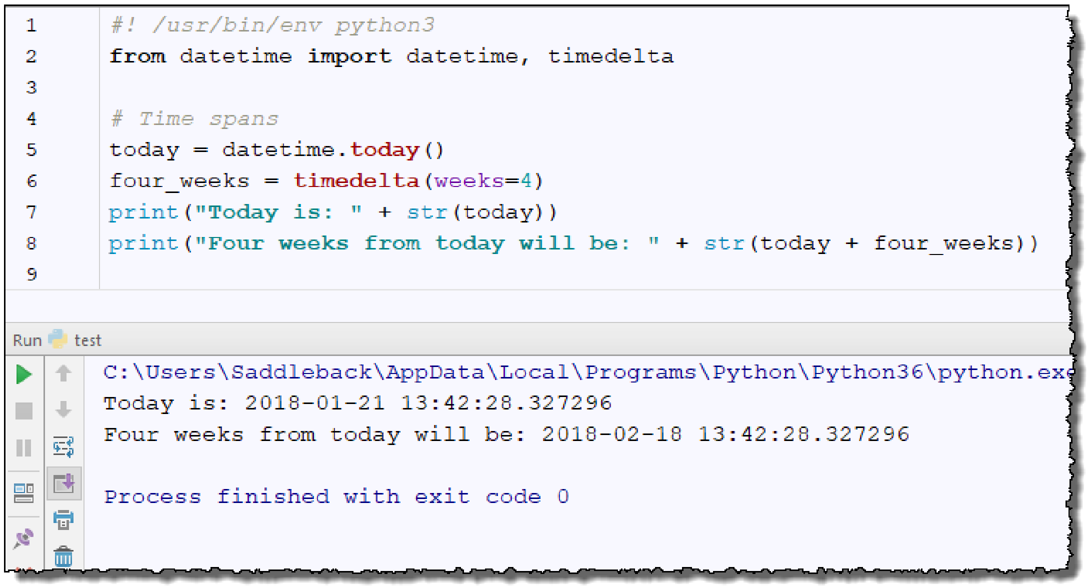
```
#! /usr/bin/env python3

from datetime import datetime

halloween = datetime(2020, 10, 31, 18, 0)

print(halloween.strftime("%Y-%m-%d"))          # 2020-10-31
print(halloween.strftime("%m/%d/%Y"))          # 10/31/2020
print(halloween.strftime("%m/%d/%y"))          # 10/31/20
print(halloween.strftime("%B %d, %Y (%A)"))    # October 31, 2020 (Saturday)
print(halloween.strftime("%B %d, %I:%M %p"))   # October 31, 06:00 PM

print(halloween.strftime("%c"))                # Sat Oct 31 18:00:00 2020
print(halloween.strftime("%x"))                # 10/31/20
```

Run: Assignment11_03

```
C:\Users\Saddleback\venv\Scripts\python.exe "C:/Users/Saddleback/PycharmProjec
2020-10-31
10/31/2020
10/31/20
October 31, 2020 (Saturday)
October 31, 06:00 PM
Sat Oct 31 18:00:00 2020
10/31/20
```

Setting Time Spans

4. Code the following:
   Screen Capture #4 (1 points)



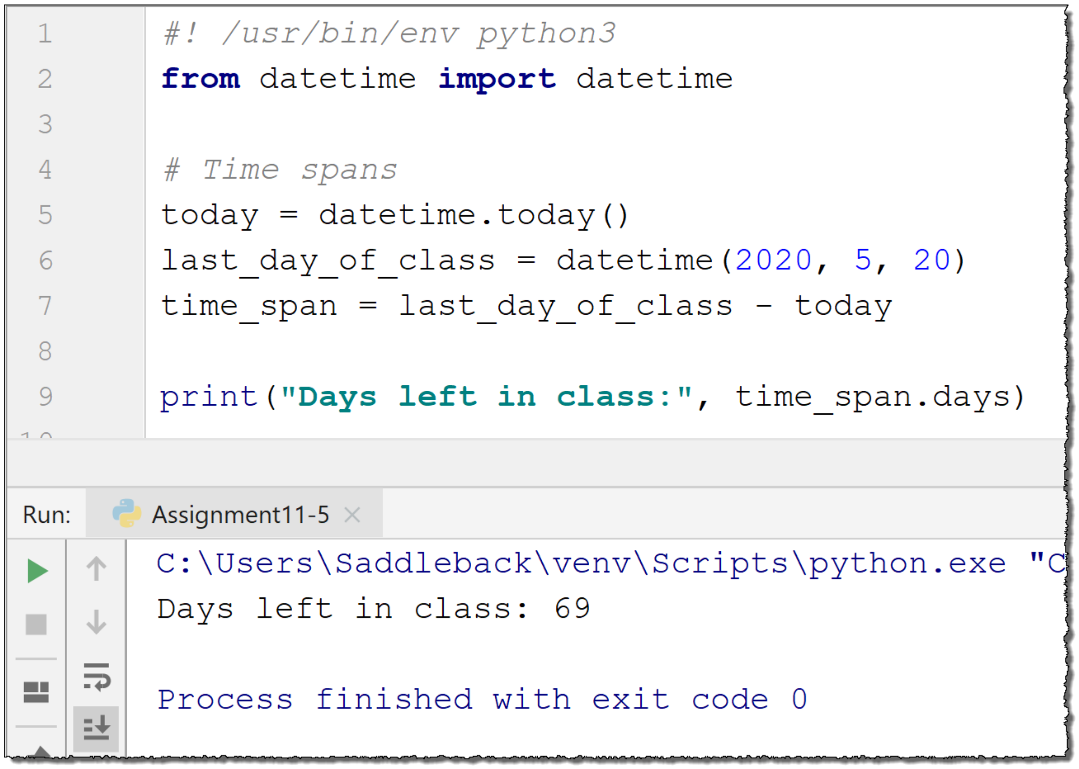
```
#! /usr/bin/env python3
from datetime import datetime, timedelta

# Time spans
today = datetime.today()
four_weeks = timedelta(weeks=4)
print("Today is: " + str(today))
print("Four weeks from today will be: " + str(today + four_weeks))
```

Run test

```
C:\Users\Saddleback\AppData\Local\Programs\Python\Python36\python.exe
Today is: 2018-01-21 13:42:28.327296
Four weeks from today will be: 2018-02-18 13:42:28.327296

Process finished with exit code 0
```

Retrieve Time Spans

5. Code the following:
   Screen Capture #5 (1 points) Use the actual end of class date



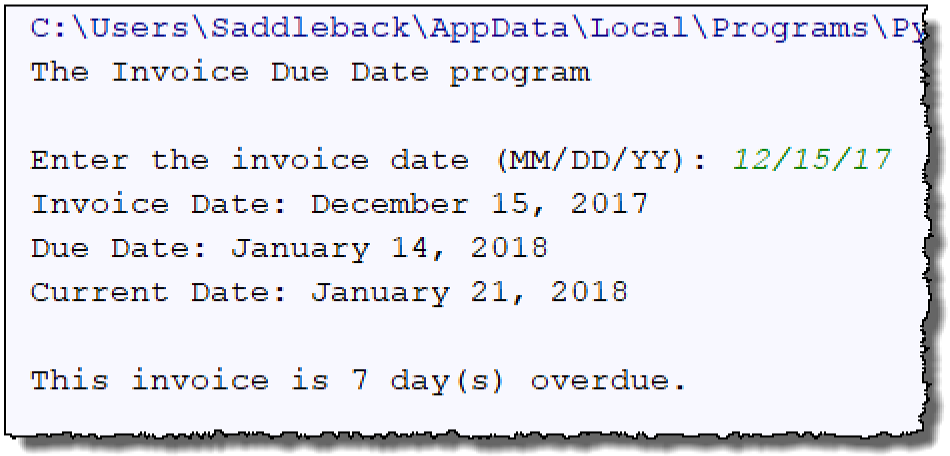
```python
#! /usr/bin/env python3
from datetime import datetime

# Time spans
today = datetime.today()
last_day_of_class = datetime(2020, 5, 20)
time_span = last_day_of_class - today

print("Days left in class:", time_span.days)
```

```
C:\Users\Saddleback\venv\Scripts\python.exe "C
Days left in class: 69

Process finished with exit code 0
```

Invoice Due Date Program

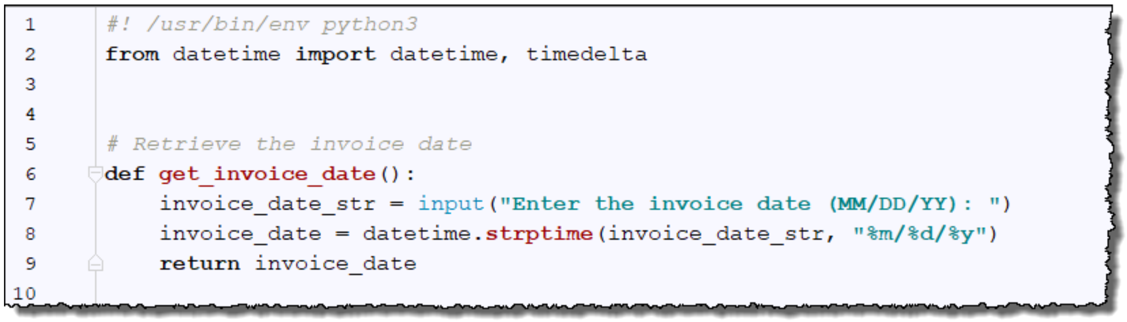
This program will prompt for an invoice date and then display the Invoice Date, Due Date (Invoice Date + 30 Days) and the current date. If the invoice date is 30 days overdue, display a line stating the invoice is 30 days overdue.

Create a new python file named **invoice.py** and submit this file with the assignment.



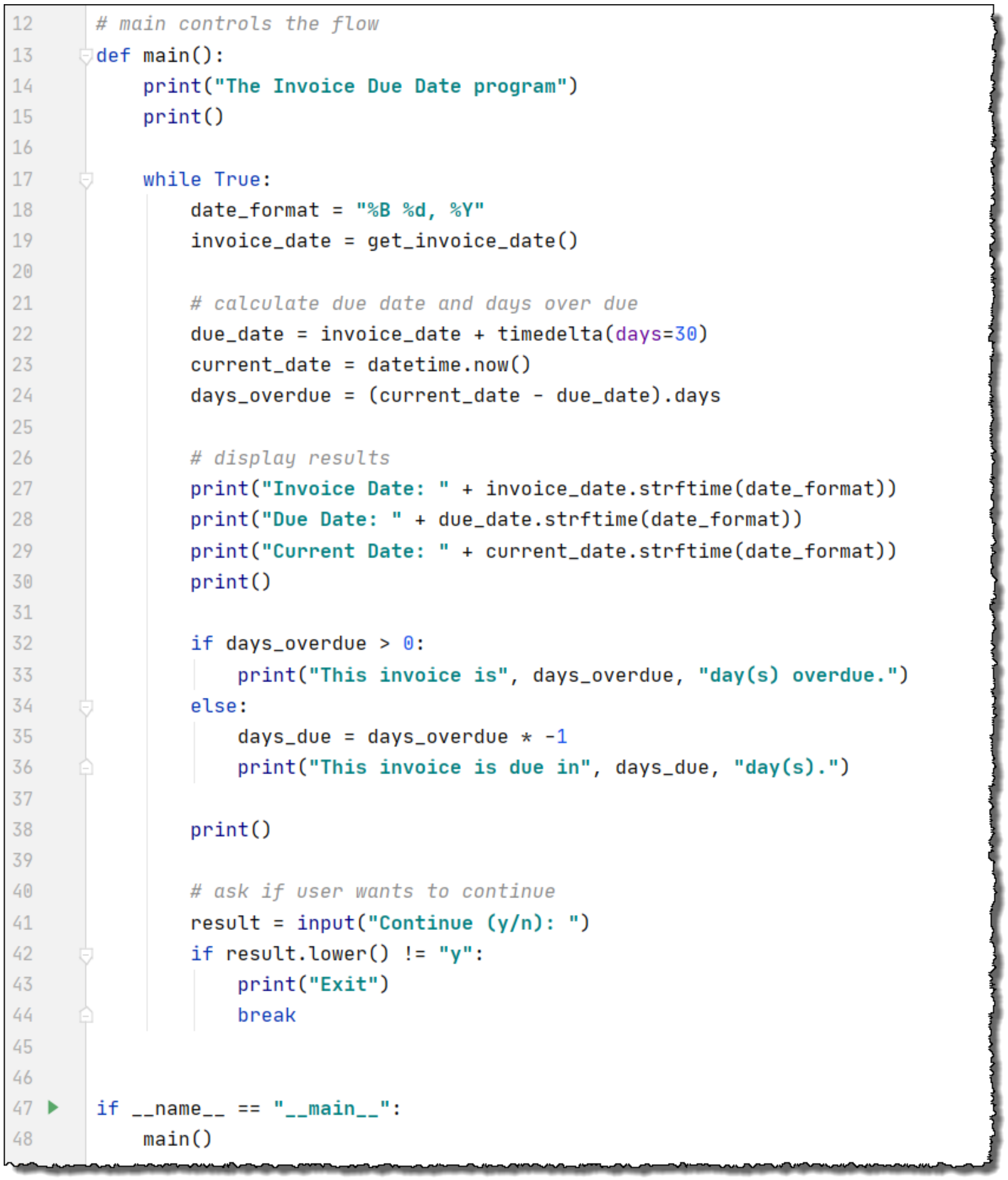
```
C:\Users\Saddleback\AppData\Local\Programs\Py
The Invoice Due Date program

Enter the invoice date (MM/DD/YY): 12/15/17
Invoice Date: December 15, 2017
Due Date: January 14, 2018
Current Date: January 21, 2018

This invoice is 7 day(s) overdue.
```

Code to Get Invoice Date



```python
#! /usr/bin/env python3
from datetime import datetime, timedelta


# Retrieve the invoice date
def get_invoice_date():
    invoice_date_str = input("Enter the invoice date (MM/DD/YY): ")
    invoice_date = datetime.strptime(invoice_date_str, "%m/%d/%y")
    return invoice_date

```

Code Main Function to Control the Flow

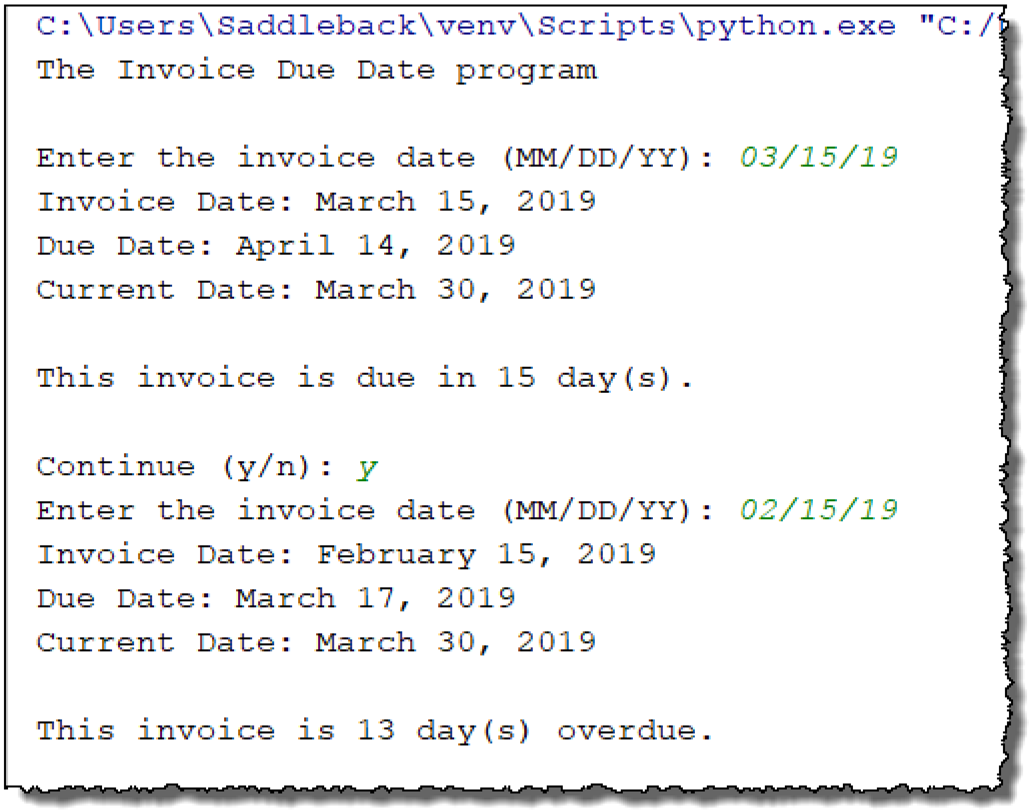


```python
# main controls the flow
def main():
    print("The Invoice Due Date program")
    print()

    while True:
        date_format = "%B %d, %Y"
        invoice_date = get_invoice_date()

        # calculate due date and days over due
        due_date = invoice_date + timedelta(days=30)
        current_date = datetime.now()
        days_overdue = (current_date - due_date).days

        # display results
        print("Invoice Date: " + invoice_date.strftime(date_format))
        print("Due Date: " + due_date.strftime(date_format))
        print("Current Date: " + current_date.strftime(date_format))
        print()

        if days_overdue > 0:
            print("This invoice is", days_overdue, "day(s) overdue.")
        else:
            days_due = days_overdue * -1
            print("This invoice is due in", days_due, "day(s).")

        print()

        # ask if user wants to continue
        result = input("Continue (y/n): ")
        if result.lower() != "y":
            print("Exit")
            break


if __name__ == "__main__":
    main()
```

6. Run the Code:
   a. Make sure you include dates that will show the invoice overdue and due in *x* days. (To get these outputs, use dates before and after today.)

Code Validation – invoice.py - (3 points)

Screen Capture #6 (3 points)

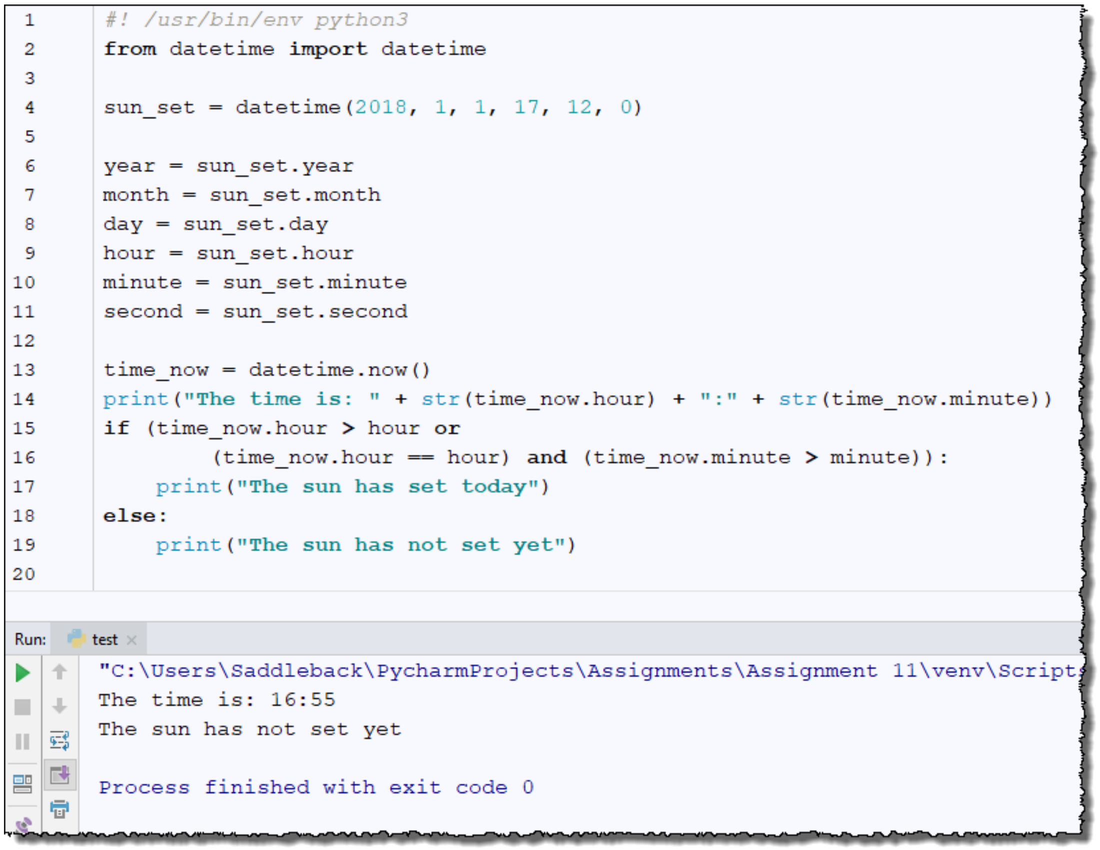


```
C:\Users\Saddleback\venv\Scripts\python.exe "C:/
The Invoice Due Date program

Enter the invoice date (MM/DD/YY): 03/15/19
Invoice Date: March 15, 2019
Due Date: April 14, 2019
Current Date: March 30, 2019

This invoice is due in 15 day(s).

Continue (y/n): y
Enter the invoice date (MM/DD/YY): 02/15/19
Invoice Date: February 15, 2019
Due Date: March 17, 2019
Current Date: March 30, 2019

This invoice is 13 day(s) overdue.
```

Date and Time Parts

Attribute	Description
year	Returns year as a four-digit number
month	Returns month as a number (1 to 12)
day	Returns day as a number (1 to 31)
hour	Returns hour as a number (0 to 23)
minute	Returns minute as a number (0 to 59)
second	Returns second as a number (0 to 59)

7. Code the following:
   Screen Capture #7 (1 points)



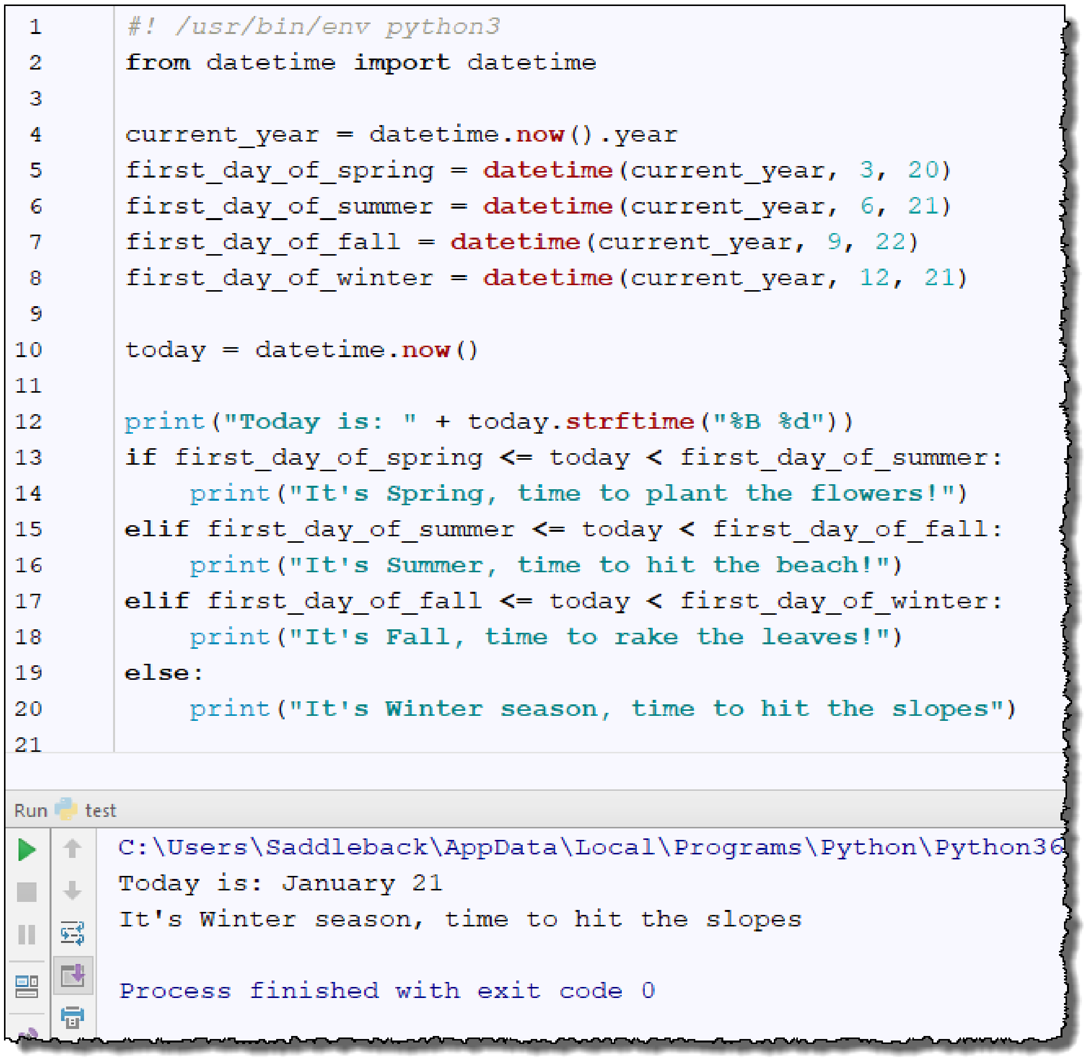
```python
#! /usr/bin/env python3
from datetime import datetime

sun_set = datetime(2018, 1, 1, 17, 12, 0)

year = sun_set.year
month = sun_set.month
day = sun_set.day
hour = sun_set.hour
minute = sun_set.minute
second = sun_set.second

time_now = datetime.now()
print("The time is: " + str(time_now.hour) + ":" + str(time_now.minute))
if (time_now.hour > hour or
        (time_now.hour == hour) and (time_now.minute > minute)):
    print("The sun has set today")
else:
    print("The sun has not set yet")
```

```
"C:\Users\Saddleback\PycharmProjects\Assignments\Assignment 11\venv\Script
The time is: 16:55
The sun has not set yet

Process finished with exit code 0
```

Comparing Dates

8. Code the following:

Screen Capture #8 (1 points)



```python
#! /usr/bin/env python3
from datetime import datetime

current_year = datetime.now().year
first_day_of_spring = datetime(current_year, 3, 20)
first_day_of_summer = datetime(current_year, 6, 21)
first_day_of_fall = datetime(current_year, 9, 22)
first_day_of_winter = datetime(current_year, 12, 21)

today = datetime.now()

print("Today is: " + today.strftime("%B %d"))
if first_day_of_spring <= today < first_day_of_summer:
    print("It's Spring, time to plant the flowers!")
elif first_day_of_summer <= today < first_day_of_fall:
    print("It's Summer, time to hit the beach!")
elif first_day_of_fall <= today < first_day_of_winter:
    print("It's Fall, time to rake the leaves!")
else:
    print("It's Winter season, time to hit the slopes")

```

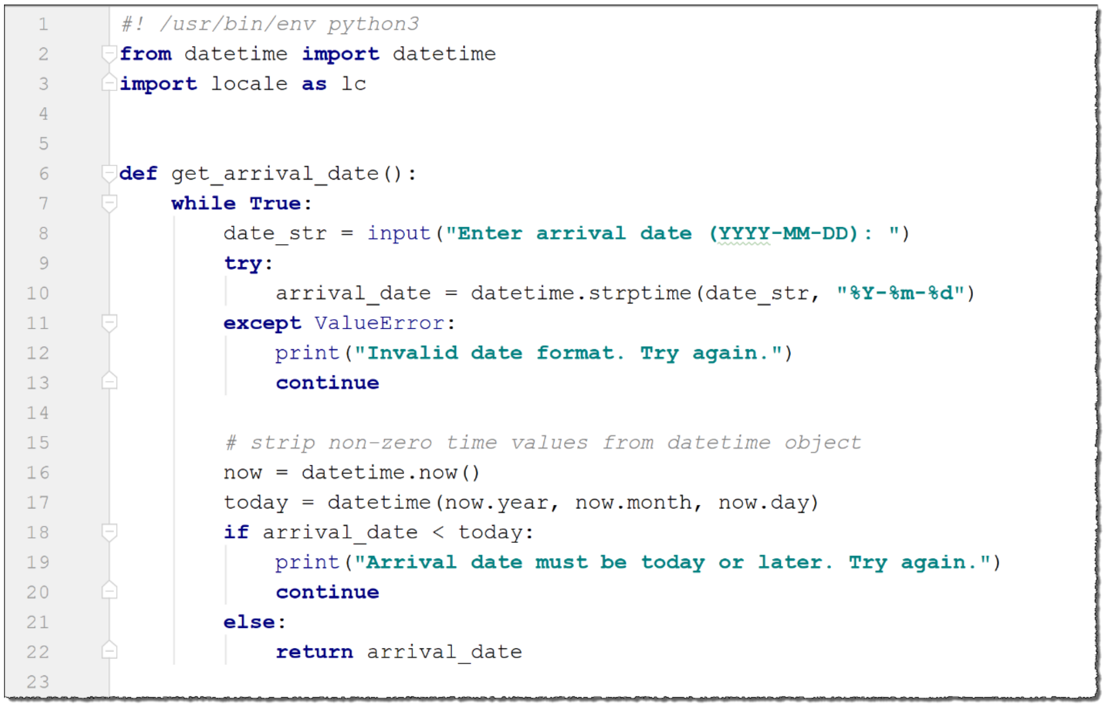
```
C:\Users\Saddleback\AppData\Local\Programs\Python\Python36
Today is: January 21
It's Winter season, time to hit the slopes

Process finished with exit code 0
```

Hotel Reservation Program

This program will prompt for the user to enter an arrival and departure date. August is the busiest month for the hotel so if the arrival date is in August, the price per night is $105 and the user should be notified, otherwise the price per night is $85.

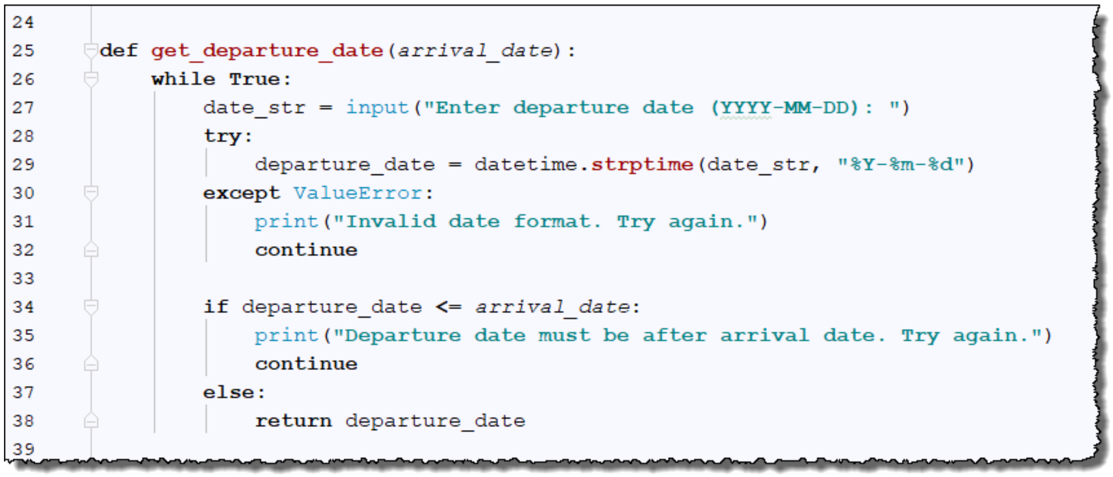
Create a new python file named **reservations.py** and submit this file with the assignment.

Code to Get Arrival Date



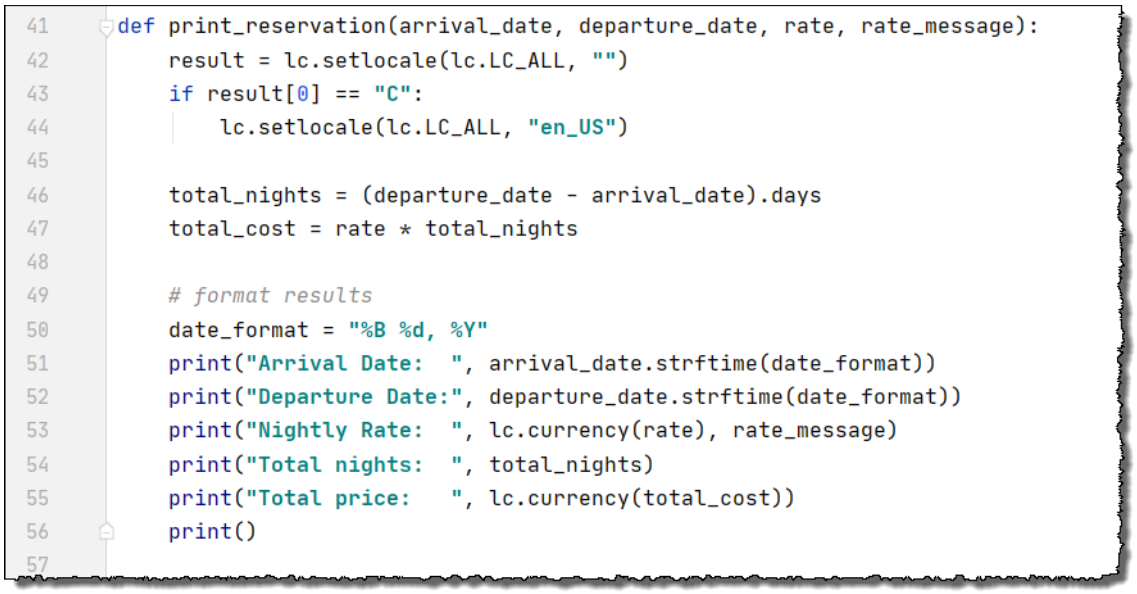
```python
#! /usr/bin/env python3
from datetime import datetime
import locale as lc


def get_arrival_date():
    while True:
        date_str = input("Enter arrival date (YYYY-MM-DD): ")
        try:
            arrival_date = datetime.strptime(date_str, "%Y-%m-%d")
        except ValueError:
            print("Invalid date format. Try again.")
            continue

        # strip non-zero time values from datetime object
        now = datetime.now()
        today = datetime(now.year, now.month, now.day)
        if arrival_date < today:
            print("Arrival date must be today or later. Try again.")
            continue
        else:
            return arrival_date

```

Code to Get Departure Date



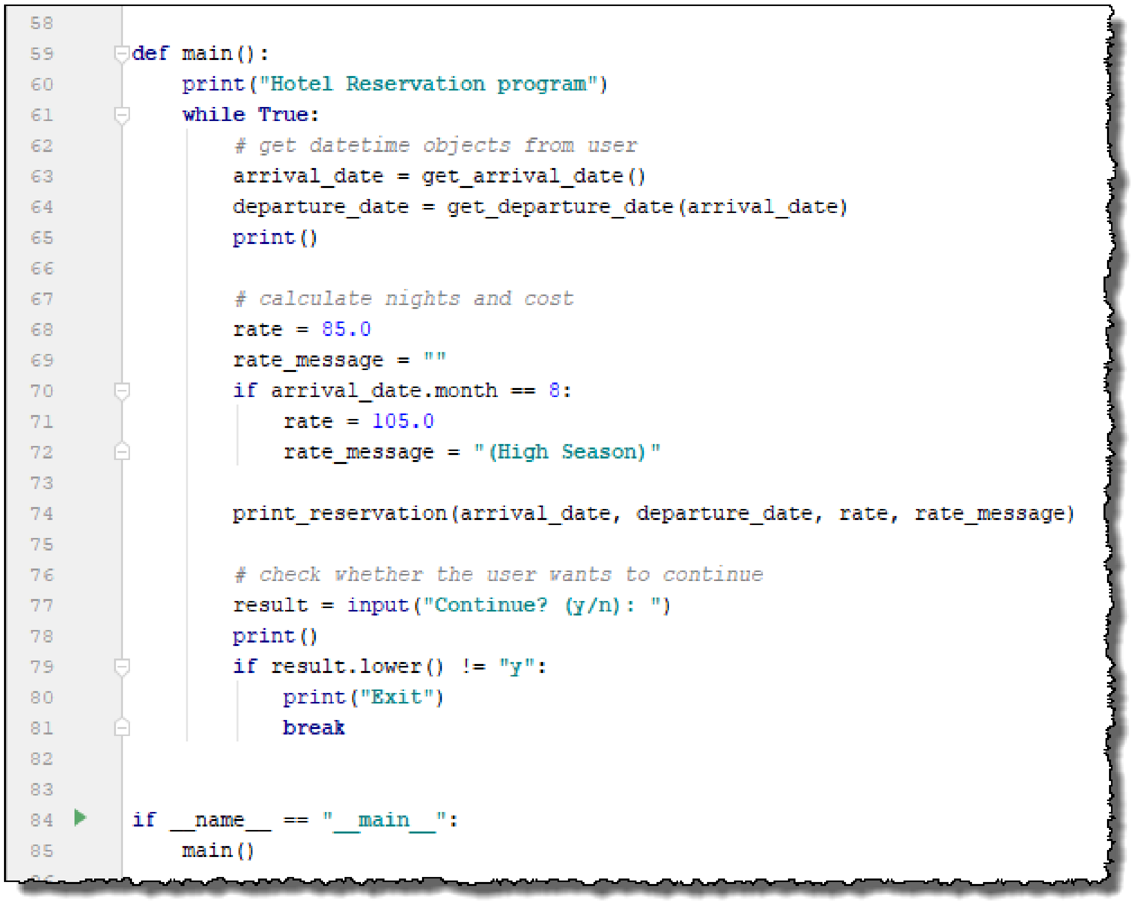
```python

def get_departure_date(arrival_date):
    while True:
        date_str = input("Enter departure date (YYYY-MM-DD): ")
        try:
            departure_date = datetime.strptime(date_str, "%Y-%m-%d")
        except ValueError:
            print("Invalid date format. Try again.")
            continue

        if departure_date <= arrival_date:
            print("Departure date must be after arrival date. Try again.")
            continue
        else:
            return departure_date

```

Code Print Reservation



```python
def print_reservation(arrival_date, departure_date, rate, rate_message):
    result = lc.setlocale(lc.LC_ALL, "")
    if result[0] == "C":
        lc.setlocale(lc.LC_ALL, "en_US")

    total_nights = (departure_date - arrival_date).days
    total_cost = rate * total_nights

    # format results
    date_format = "%B %d, %Y"
    print("Arrival Date:   ", arrival_date.strftime(date_format))
    print("Departure Date:", departure_date.strftime(date_format))
    print("Nightly Rate:   ", lc.currency(rate), rate_message)
    print("Total nights:   ", total_nights)
    print("Total price:    ", lc.currency(total_cost))
    print()

```

Code Main Function to Control the Flow

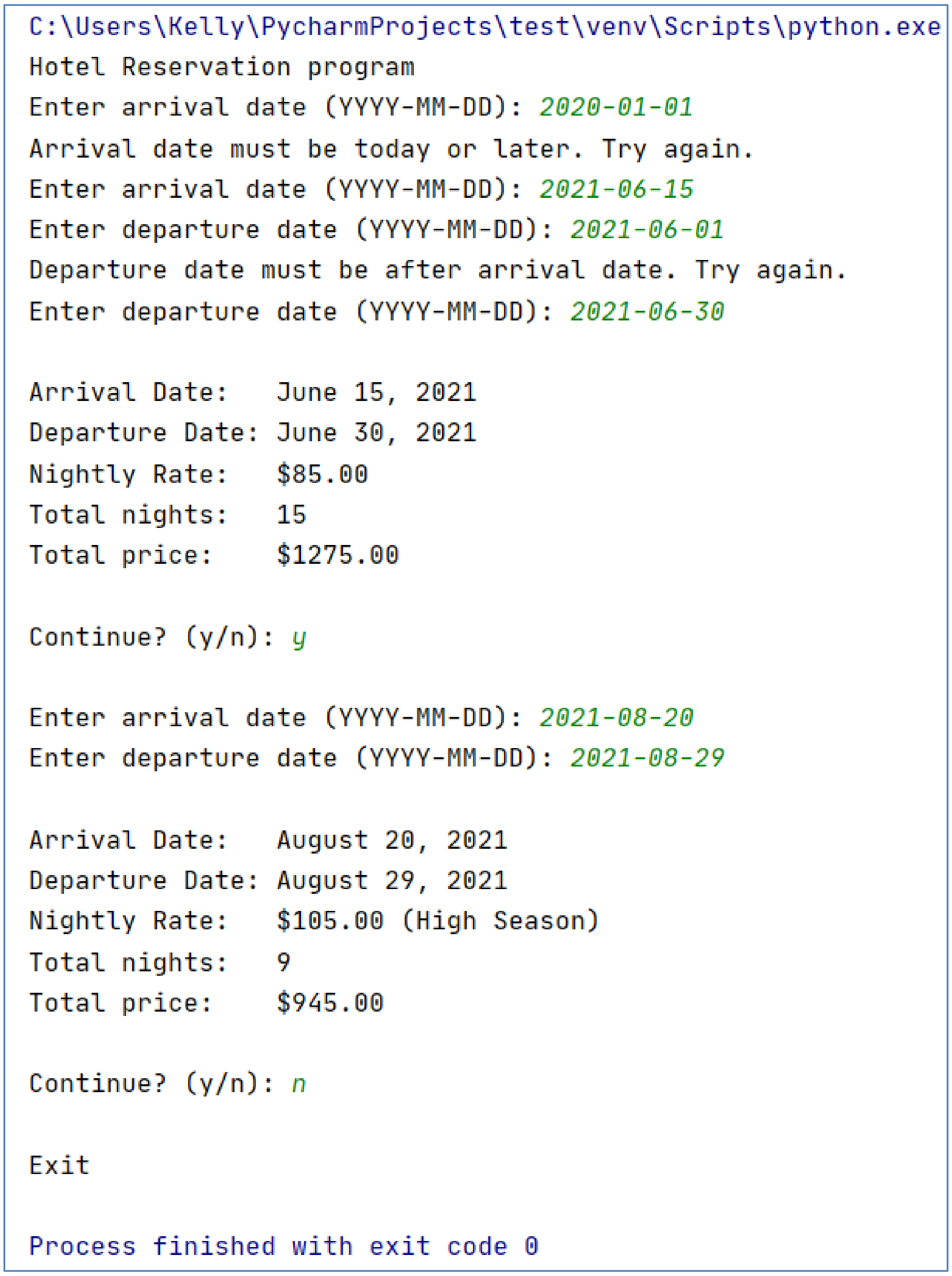


```python

def main():
    print("Hotel Reservation program")
    while True:
        # get datetime objects from user
        arrival_date = get_arrival_date()
        departure_date = get_departure_date(arrival_date)
        print()

        # calculate nights and cost
        rate = 85.0
        rate_message = ""
        if arrival_date.month == 8:
            rate = 105.0
            rate_message = "(High Season)"

        print_reservation(arrival_date, departure_date, rate, rate_message)

        # check whether the user wants to continue
        result = input("Continue? (y/n): ")
        print()
        if result.lower() != "y":
            print("Exit")
            break


if __name__ == "__main__":
    main()
```

Display the Output

9. Run the Code:
    a. Output should include arrival date error, departure date error, non-peak month rate and peak month rate.

Code Validation – reservations.py (3 points)
Screen Capture #9 (4 points)



```
C:\Users\Kelly\PycharmProjects\test\venv\Scripts\python.exe
Hotel Reservation program
Enter arrival date (YYYY-MM-DD): 2020-01-01
Arrival date must be today or later. Try again.
Enter arrival date (YYYY-MM-DD): 2021-06-15
Enter departure date (YYYY-MM-DD): 2021-06-01
Departure date must be after arrival date. Try again.
Enter departure date (YYYY-MM-DD): 2021-06-30

Arrival Date:   June 15, 2021
Departure Date: June 30, 2021
Nightly Rate:   $85.00
Total nights:   15
Total price:    $1275.00

Continue? (y/n): y

Enter arrival date (YYYY-MM-DD): 2021-08-20
Enter departure date (YYYY-MM-DD): 2021-08-29

Arrival Date:   August 20, 2021
Departure Date: August 29, 2021
Nightly Rate:   $105.00 (High Season)
Total nights:   9
Total price:    $945.00

Continue? (y/n): n

Exit

Process finished with exit code 0
```

Extra Credit

To get full points for each extra credit, you must include screen captures of the running output as well as the python (.py) code files.

Extra Credit #1 – Birthday Calculator (+1 Extra Credit)

Create a program that accepts a name and a birth date and displays the person’s birthday, the current day, the person’s age, and the number of days until the person’s next birthday.

```
Birthday Calculator

Enter name: Joel
Enter birthday (MM/DD/YY): 2/4/68
Birthday: Sunday, February 04, 1968
Today:    Tuesday, November 22, 2016
Joel is 48 years old.
Joel's birthday is in 73 days.

Continue? (y/n): y

Enter name: Django
Enter birthday (MM/DD/YY): 12/1/07
Birthday: Saturday, December 01, 2007
Today:    Tuesday, November 22, 2016
Django is 8 years old.
Django's birthday is in 9 days.

Continue? (y/n): y

Enter name: Mike
Enter birthday (MM/DD/YY): 11/22/86
Birthday: Saturday, November 22, 1986
Today:    Tuesday, November 22, 2016
Mike is 30 years old.
Mike's birthday is today!

Continue? (y/n): n
```

Specifications:

- Allow the user to enter a date in the MM/DD/YY format. Adjust the date so that it is correct even if the birth year is later than the current year.
- When you calculate the person’s age, don’t take leap year into account. If the person is more than 2 years old, display the person’s age in years. Otherwise, display the person’s age in days.
- When you display the message that indicates the number of day’s to the person’s birthday, you can use the following format for a person with a name of John:
    - Today - John’s birthday is today!
    - Tomorrow - John’s birthday is tomorrow!
    - Yesterday - John’s birthday was yesterday!
    - Other days - John’s birthday is in *X* days.

Extra Credit #2 – Arrival Time Estimator (+1 Extra Credit)

Create a program that calculates the estimated duration of a trip in hours and minutes. This should include an estimated date/time of departure and an estimated date/time of arrival.

```
Arrival Time Estimator

Estimated date of departure (YYYY-MM-DD): 2016-11-23
Estimated time of departure (HH:MM AM/PM): 10:30 AM
Enter miles: 200
Enter miles per hour: 65

Estimated travel time
Hours: 3
Minutes: 5
Estimated date of arrival: 2016-11-23
Estimated time of arrival: 01:35 PM

Continue? (y/n): y

Estimated date of departure (YYYY-MM-DD): 2016-11-29
Estimated time of departure (HH:MM AM/PM): 11:15 PM
Enter miles: 500
Enter miles per hour: 80

Estimated travel time
Hours: 6
Minutes: 20
Estimated date of arrival: 2016-11-30
Estimated time of arrival: 05:35 AM

Continue? (y/n): n

Bye!
```

Specifications:

- For the date/time of departure and arrival, the program should use the YYYY-MM-DD format for dates and the HH:MM AM/PM format for times.
- For the miles and miles per hour, the program should only accept integer entries like 200 and 65.
- Assume that a user will enter valid dates.