02 Prove: Calling Functions

Purpose

Prove that you can write a Python program that calls functions and methods to get the current date and to append values to a text file.

Problem Statement

Many companies wish to understand the needs and wants of their customers more deeply so the company can create products that fill those needs and wants. One way to understand customers more deeply is to record the values entered by customers while they are using a program and then to analyze those values. One common way to record values is for a program to store them in a file.

Assignment

The previous lesson's prove assignment required you to write a program named `tire_volume.py` that computes the approximate volume of air inside a tire. Add code near the end of that program that does the following:

1. Gets the current date from the computer's clock.
2. Opens a text file named `volumes.txt` for appending.
3. Appends to the end of the `volumes.txt` file one line of text that contains the following five values:
    a. current date
    b. width of the tire
    c. aspect ratio of the tire
    d. diameter of the wheel
    e. volume of the tire

Helpful Documentation

- The prepare content for this lesson explains how to call a function and a method.
- The `datetime.now()` method from the standard Python `datetime` module will get the current date and time from your computer's operating system. Here is an excerpt from the official documentation for the `datetime.now` method:

    > `datetime.now(tz=None)`
    > Return the current local date and time.
    >
    > *tz* is optional, but if it not `None`, it must be `tzinfo` (time zone information) object

    These two Microsoft videos explain how to use methods from the standard `datetime` module.

    Date data types (8 minutes)

    Demonstration: Dates (9 minutes)

    The following Python code imports the `datetime` class from the `datetime` module and calls the `datetime.now` method to get the current date and time from a computer's operating system. Then it uses an f-string to format and print the current date and time.

```
 1  # Import the datetime class from the datetime
 2  # module so that it can be used in this program.
 3  from datetime import datetime
 4
 5  # Call the now() method to get the current date and
 6  # time as a datetime object from the computer's clock.
 7  current_date_and_time = datetime.now()
 8
 9  # Print only the date part of the current date and time.
10  print(f"{current_date_and_time:%Y-%m-%d}")
```

After the computer executes line 7 in the above code, the variable *current_date_and_time* will hold the current date and time. Within the f-string at line 10, the string sequences that begin with the percent symbol (%) are called format codes. The format codes and their meaning are listed in this document. When executed, the previous example code will print the current date and time to the terminal window like this:

```
> python date_example.py
2020-07-24
```

- The built-in `open()` function opens a file for reading or writing. Here is an excerpt from the official documentation for the `open` function:

  > `open(filename, mode="rt")`
  > Open a file and return a corresponding file object.
  >
  > *filename* is the name of the file to be opened.
  >
  > *mode* is an optional string that specifies the mode in which the file will be opened. It defaults to `'rt'` which means open for reading in text mode. Other common values are `'wt'` for writing a text file (truncating the file if it already exists), and `'at'` for appending to the end of a text file.

- The built-in `print()` function prints text to a terminal window or to a text file. Here is an excerpt from the official documentation for the `print` function:

  > `print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)`
  > Print objects to the text stream *file*, separated by *sep* and followed by *end*. *sep*, *end*, *file* and *flush*, if present, must be given as named arguments.

The following example code calls the `open` function to open a file for appending text and then calls the `print` function two times to print two lines of text to the file.

```
model = "GMC Acadia"
length = 193
width = 75
height = 67

# Open a text file named dimensions.txt in append mode.
with open("dimensions.txt", "at") as dimens_file:

    # Print a car's model and dimensions to the file.
    print(model, file=dimens_file)
    print(f"{length}, {width}, {height}", file=dimens_file)
```

Testing Procedure

Verify that your program works correctly by following each step in this testing procedure:

1. Run your program using the input shown in the sample run section below. Ensure that your program's output matches the sample run output. Use VS Code to open the `volumes.txt` file and verify that the last line of text in the file looks like this except the date will be different:
   `2020-03-18, 185, 50, 14, 24.09`

2. Run your program using these values: 205, 60, 15 and verify that your program outputs 39.92 for the volume. Use VS Code to open the `volumes.txt` file and verify that the last two lines of text in the file look like this except the dates will be different:
   `2020-03-18, 185, 50, 14, 24.09`
   `2020-04-16, 205, 60, 15, 39.92`

Sample Run

```
> python tire_volume.py
Enter the width of the tire in mm (ex 205): 185
Enter the aspect ratio of the tire (ex 60): 50
Enter the diameter of the wheel in inches (ex 15): 14

The approximate volume is 24.09 liters
```

Exceeding the Requirements

Here are a few suggestions for additional features you could add to your program.

1. Find tire prices for four or more tire sizes online. Add a set of `if` ... `elif` ... `else` statements in your program that use the tire width, tire aspect ratio, and wheel diameter that the user enters to find a price and then print the price.

2. After your program prints the tire volume to the terminal, your program should ask the user if she wants to buy tires with the dimensions that she entered. If the user answers "yes", your program should ask for her phone number and store her phone number in the `volumes.txt` file.

Submission

To submit your program, return to I-Learn and do these two things:

1. Upload your program (the .py file) for feedback.

2. Add a submission comment that specifies the grading category that best describes your program along with a one or two sentence justification for your choice. The grading criteria are:
   - 1. Some attempt made
   - 2. Developing but significantly deficient
   - 3. Slightly deficient
   - 4. Meets requirements
   - 5. Exceeds requirements