

# lab1\_sp25

January 21, 2025

## 1 Lab Assignment 1: How to Get Yourself Unstuck

### 1.1 DS 6001

#### 1.2 Problem 1

Please read the following blog post describing some common Python errors: <https://www.tutorialsteacher.com/python/error-types-in-python>. Then for each of the following situations, state the what type of exception is Python likely to produce. (I'm looking for specific Exceptions that appear in the table on this website.) [0.5 points each]

##### 1.2.1 Part a

I accidentally type two closing parentheses `)` instead of one at the end of a line of code.

**This would produce a `SyntaxError`.**

##### 1.2.2 Part b

I found some code where someone created a useful function using the `def` statement. I copy that code and paste it into my own script. However, the copy-and-pasting somehow got rid of all my tabs so that every line is aligned on the left.

**This would produce an `IndentationError`.**

##### 1.2.3 Part c

I use Python to calculate the number of atoms within every star in every galaxy in the universe.

**This would produce an `MemoryError`.**

##### 1.2.4 Part d

I divide  $\pi$  by the number of [Super Bowls won by the Cleveland Browns](#).

**This would produce a `ZeroDivisionError`.**

##### 1.2.5 Part e

I am hungry, so I reset my Python kernel and start a new script. I write one line of code: `print(tacos)`.

**This would produce a `NameError`.**

### 1.2.6 Part f

I download the entire internet so I can run a regression on it.

**This would produce a `MemoryError`.**

### 1.2.7 Part g

I take the square root of the words “twenty five”.

**This would produce a `TypeError`.**

### 1.2.8 Part h

Something entirely different and unexpected happens, causing Python to stop short, its words caught in its mouth, as if reason itself boarded the last train to Seattle in the chill of an unforgiving winter’s night. There is no explanation, nor can there ever be. Only one thing is certain: something went wrong.

**This would be a `RuntimeError`.**

## 1.3 Problem 2

Use the built-in docstrings or `help()` text for each method to answer the following questions:

### 1.3.1 Part a

I am loading data from a CSV file using `pd.read_csv()`. But it is taking a long time. Find a parameter that can speed up this method by using the C engine instead of the Python one. [1 point]

```
[1]: import pandas as pd

# Making this a note so as to not include the output in my HW upload but this_
# is what I consulted:
# help(pd.read_csv)
```

We can introduce the argument `engine="c"` to use the C engine parser which is faster.

### 1.3.2 Part b

I create the following dictionary:

```
mytrips = {'Portland': 'So many bicycles',
           'Nashville': 'Lost in Opryland Mall again',
           'Richmond': 'Got tattoo and became a hipster',
           'San Diego': 'Chased off beach by sea lion'}
```

How can I see only the cities I’ve visited (the keys) but not the travel notes (the values)? [1 point]

```
[ ]: mytrips = {'Portland': 'So many bicycles',
               'Nashville': 'Lost in Opryland Mall again',
               'Richmond': 'Got tattoo and became a hipster',
```

```
'San Diego': 'Chased off beach by sea lion'}

# Making this a note so as to not include the output in my HW upload but this
↳ is what I consulted:
#help(dict)
```

```
[3]: mytrips.keys()
```

```
[3]: dict_keys(['Portland', 'Nashville', 'Richmond', 'San Diego'])
```

.keys() retrieves the dictionary's keys.

### 1.3.3 Part c

I am using `px.scatter()` from plotly express to create an HTML enabled scatterplot. How would I add a box plot to the x-axis to visualize the distribution of the x variable? [1 point]

```
[4]: import plotly.express as px

# Making this a note so as to not include the output in my HW upload but this
↳ is what I consulted:
# help(px.scatter)
```

```
[5]: from IPython.display import Image

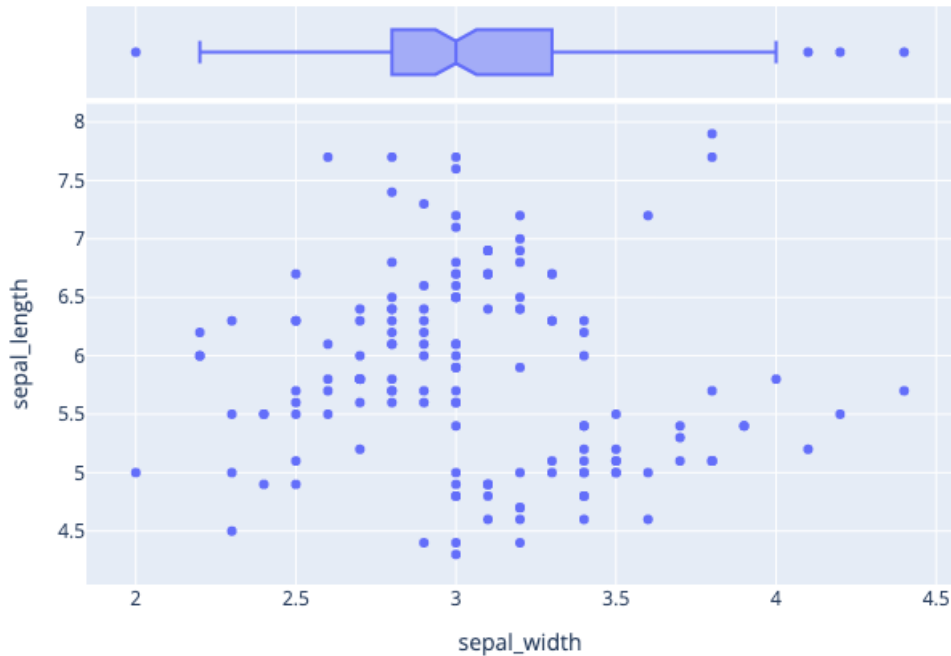
df = px.data.iris()

fig = px.scatter(df, x="sepal_width", y="sepal_length", marginal_x="box")

fig.write_image("scatter_plot.png")

Image("scatter_plot.png")
```

```
[5]:
```



We can use the argument `marginal_x="box"` to visualize a boxplot along the x-axis.

## 1.4 Problem 3

For the following questions, find some examples of people using Stack Overflow badly, either by being toxic while answering a question, or by asking a poorly worded or researched question. You will have to do some digging as these sorts of behaviors aren't the first ones that pop up in a normal search. For toxic answers, look down the page for responses that are NOT the ones accepted by the question-asker. (Remember, these responses are pretty well hidden to us, but they came through loud and clear for the original asker.) For poor questions, look for posts with few responses or maybe some downvotes.

### 1.4.1 Part a

Search through the questions on Stack Overflow tagged as Python questions: <https://stackoverflow.com/questions/tagged/python>. Find a question in which an answerer exhibits passive toxic behavior as defined in this module's notebook. Provide a link, and describe what specific behavior leads you to identify this answer as toxic. [1 point]

<https://stackoverflow.com/questions/77984729/importerror-cannot-import-name-vectorstoreindex-from-llama-index-unknown-l>

The final comment, with a -1 rating, comes across as passive toxic behavior to me because (1) the commenter says "first, you need to install:" and provides a line of code that the original author

already said they have in their code and (2) then says “as it was mentioned by others [the OP needs to write import statements].” This comment is certainly unhelpful but also slightly passive and toxic in the way it does not really provide any new information, repeats exactly what the OP wrote in their code, and refers to other answers (if you’re just going to say the other answers are correct, you don’t need to write a new answer.)

### 1.4.2 Part b

Search through the questions on Stack Overflow tagged as Python questions: <https://stackoverflow.com/questions/tagged/python>. Find a question in which a questioner self-sabotages by asking the question in a way that the community does not appreciate. Provide a link, and describe what the questioner did specifically to annoy the community of answerers. [1 point]

<https://stackoverflow.com/questions/79369715/how-do-you-find-location-of-download-button-on-web-page-to-provide-to-python-w>

The questioner self-sabotages and annoys the community of answerers in: - Lack of effort in describing the problem or showing prior attempts. - Ignoring constructive feedback while expecting a full solution. - Framing the problem in a way that shifts the entire burden of the task onto the answerers.

## 1.5 Problem 4

These days there are so many Marvel superheroes, but only six superheroes count as original Avengers: Hulk, Captain America, Iron Man, Black Widow, Hawkeye, and Thor. I wrote a function, `is_avenger()`, that takes a string as an input. The function looks to see if this string is the name of one of the original six Avengers. If so, it prints that the string is an original Avenger, and if not, it prints that the string is not an original Avenger. Here’s the code for the function:

```
[6]: def is_avenger(name):  
    if name=="Hulk" or "Captain America" or "Iron Man" or "Black Widow" or "  
    ↪Hawkeye" or "Thor":  
        print(name + "'s an original Avenger!")  
    else:  
        print(name + " is NOT an original Avenger.")
```

To test whether this function is working, I pass the names of some original Avengers to the function:

```
[7]: is_avenger("Black Widow")
```

Black Widow's an original Avenger!

```
[8]: is_avenger("Iron Man")
```

Iron Man's an original Avenger!

```
[9]: is_avenger("Hulk")
```

Hulk's an original Avenger!

Looks good! But next, I pass some other strings to the function:

```
[10]: is_avenger("Spiderman")
```

Spiderman's an original Avenger!

```
[11]: is_avenger("Beyonce")
```

Beyonce's an original Avenger!

Beyonce is a hero, but she was too busy going on tour to be in the Avengers movie. Also, Spiderman definitely was NOT an original Avenger. It turns out that this function will display that any string we write here is an original Avenger, which is incorrect. To fix this function, let's turn to Stack Overflow.

**Part a** The first step to solving a problem using Stack Overflow is to do a comprehensive search of available resources to try to solve the problem. There is a post on Stack Overflow that very specifically solves our problem. Do a Google search and find this post. In your lab report, write the link to this Stack Overflow page, and the search terms you entered into Google to find this page.

<https://stackoverflow.com/questions/6531482/how-to-check-if-a-string-contains-an-element-from-a-list-in-python>

This Stack Overflow post solves the problem in that we would need to use `any()`.

I searched: `check if string is in list of strings site:stackoverflow.com` on Google to find this post.

**Part b** Suppose that no Stack Overflow posts yet existed to help us solve this problem. It would be time to consider writing a post ourselves. In your notebook, write a good title for this post. Do NOT copy the title to the posts you found for part a. (Hint: for details on how to write a good title see the slides or <https://stackoverflow.com/help/how-to-ask>) [1 point]

A good title could be: “How do I check if a string matches one of a predefined list of names in Python?”

**Part c** One characteristic of a Stack Overflow post that is likely to get good responses is a minimal working example. A minimal working example is code with the following properties:

1. It can be executed on anyone's local machine without needing a data file or a hard-to-get package or module
2. It always produces the problematic output
3. It using as few lines of code as possible, and is written in the simplest way to write that code

Write a minimal working example for this problem. (Hint: the problem at hand has nothing to do with the Avengers, really.) [2 points]

```
[12]: def is_mammal(name):  
      if name=="Cat" or "Dog":  
          print(name + " is a mammal.")  
      else:
```

```
print(name + " is not a mammal.")

is_mammal("Cat")
is_mammal("Snake")
```

Cat is a mammal.  
Snake is a mammal.

## 1.6 Problem 5

This problem focuses on how to use an LLM as a code assistant for coding tasks. In this course you are free to use any LLM you want, or none at all, so long as you remember that LLM code assistants are wonderfully useful if you know how to code and quite destructive to your ability to learn the basics if you do not, so pay attention to your use of AI and catch yourself if you see yourself using these tools as crutches.

However, for this problem only, please download

- [VS Code](#)
- The [Python plug-in](#) for VS Code
- The [Jupyter plug-in](#) for VS Code
- And the [Codeium plug-in](#) for VS Code (after creating a free account on [codeium.com](https://codeium.com))

Please review [section 1.9 of Surfing the Data Pipeline with Python](#) to learn about how to use the autocomplete, in-line prompting, and external chat functionality that Codeium provides.

Once you have these tools installed, we will use them to address the following problem:

The [Collatz Conjecture](#) states that if you take a positive integer and apply the following algorithm:

- If even, divide by two
- If odd, multiply by 3 and add 1

And then repeat these steps for the result, then eventually every number (no matter how large) will converge to a sequence of 4, 2, 1, 4, 2, 1, ... This statement is a conjecture because it is not proven for all positive integers.

Source: <https://xkcd.com/710/>

### 1.6.1 Part a

Use the Codeium external chat window to ask for a Python function that applies the Collatz algorithm described above one time to a given positive integer. Write your prompt in your notebook and paste the code it returns in a code cell. [2 points]

“Give me a Python function that applies the Collatz algorithm to a given positive integer just one time.”

```
[13]: def collatz_step(n):
      """
      Applies the Collatz algorithm to a given positive integer one time.
```

```

Args:
    n (int): A positive integer.

Returns:
    int: The result of one step of the Collatz algorithm.
    """
    if n <= 0:
        raise ValueError("Input must be a positive integer")

    if n % 2 == 0:
        return n // 2
    else:
        return 3 * n + 1

print(collatz_step(10)) # 5

```

5

### 1.6.2 Part b

Copy the function you received in part a in a new code cell. Place your cursor somewhere inside the function prior to the `return` statements, and use in-line prompting to ask Codeium to add a `ValueError` if the inputted number is negative and a `TypeError` if the inputted number is not an integer. [2 points]

```

[14]: def collatz_step(n):
        """
        Applies the Collatz algorithm to a given positive integer one time.

        Args:
            n (int): A positive integer.

        Returns:
            int: The result of one step of the Collatz algorithm.
            """
        if n <= 0:
            raise ValueError("Input must be a positive integer")
        if not isinstance(n, int):
            raise TypeError("Input must be an integer")
        if n % 2 == 0:
            return n // 2
        else:
            return 3 * n + 1

```

### 1.6.3 Part c

To test whether applying the Collatz function repeatedly will eventually result in an output of 1, write a while loop that begins with a user-specified positive integer and applies the function from



part b over and over until a value of 1 is achieved. Instead of prompting the LLM to create this loop, try coding it yourself but use the autocompletion feature as much as you can. [2 points]

```
[15]: def collatz_loop(n):  
        while n != 1:  
            print(n)  
            n = collatz_step(n)  
  
        print(n)  
  
n = 11  
collatz_loop(n)
```

```
11  
34  
17  
52  
26  
13  
40  
20  
10  
5  
16  
8  
4  
2  
1
```