3.4. Know Your Error Messages

Many problems in your program will lead to an error message. For example as I was writing and testing this chapter of the book I wrote the following version of the example program in the previous section.

```
current_time_str = input("What is the current time (in hours 0-23)?")
wait_time_str = input("How many hours do you want to wait")

current_time_int = int(current_time_str)
wait_time_int = int(wait_time_int)

final_time_int = current_time_int + wait_time_int
print(final_time_int)
```

Can you see what is wrong, just by looking at the code? Maybe, maybe not. Our brain tends to see what we think is there, so sometimes it is very hard to find the problem just by looking at the code. Especially when it is our own code and we are sure that we have done everything right!

Let's try the program again, but this time in an activecode:

Save & Run 2/1/2023, 11:45:30 PM - 5 of 5 Show CodeLens

```
1 current_time_str = input("What is the current time (in hours 0-23)?")
2 wait_time_str = input("How many hours do you want to wait")
3
4 current_time_int = input(10)
5 wait_time_int = input(5)
6
7 final_time_int = current_time_int + wait_time_int
8 print(final_time_int)
9
```

Activity: 3.4.1 ActiveCode (db_ex3_5)

Aha! Now we have an error message that might be useful. The name error tells us that `wait_time_int` is not defined. It also tells us that the error is on line 5. That's **really** useful information. Now look at line five and you will see that `wait_time_int` is used on both the left and the right hand side of the assignment statement.

Note

The error descriptions you see in activecode may be different (and more understandable!) than in a regular Python interpreter. The interpreter in activecode is limited in many ways, but it is intended for beginners, including the wording chosen to describe errors.

Q-2: Which of the following explains why `wait_time_int = int(wait_time_int)` is an error?

- **A. You cannot use a variable on both the left and right hand sides of an assignment statement.**
- **B. wait_time_int does not have a value so it cannot be used on the right hand side.**
- **C. This is not really an error, Python is broken.**

Check Me Compare me

✓ Yes. Variables must already have values in order to be used on the right hand side.

Activity: 3.4.2 Multiple Choice (db_qex32)

(BeginningtipsforDebugging.html) (Summary.html)

Before you keep reading...

In writing and using this book over the last few years we have collected a lot of statistics about the programs in this book. Here are some statistics about error messages for the exercise we have been looking at.

Message	Number	Percent
ParseError:	4999	54.74%
TypeError:	1305	14.29%
NameError:	1009	11.05%
ValueError:	893	9.78%
URIError:	334	3.66%
TokenError:	244	2.67%
SyntaxError:	227	2.49%
TimeLimitError:	44	0.48%
IndentationError:	28	0.31%
AttributeError:	27	0.30%
ImportError:	16	0.18%
IndexError:	6	0.07%

Nearly 90% of the error messages encountered for this problem are ParseError, TypeError, NameError, or ValueError. We will look at these errors in three stages:

- First we will define what these four error messages mean.
- Then, we will look at some examples that cause these errors to occur.
- Finally we will look at ways to help uncover the root cause of these messages.

3.4.1. ParseError

Parse errors happen when you make an error in the syntax of your program. Syntax errors are like making grammatical errors in writing. If you don't use periods and commas in your writing then you are making it hard for other readers to figure out what you are trying to say. Similarly Python has certain grammatical rules that must be followed or else Python can't figure out what you are trying to say.

Usually ParseErrors can be traced back to missing punctuation characters, such as parentheses, quotation marks, or commas. Remember that in Python commas are used to separate parameters to functions. Paretheses must be balanced, or else Python thinks that you are trying to include everything that follows as a parameter to some function.

Here are a couple examples of Parse errors in the example program we have been using. See if you can figure out what caused them.

Question Answer

(BeginningtipsforDebugging.html) (Summary.html)

```
current_time_str = input("What is the current time (in hours 0-23)?")
wait_time_str = input("How many hours do you want to wait"

current_time_int = int(current_time_str)
wait_time_int = int(wait_time_str)

final_time_int = current_time_int + wait_time_int
print(final_time_int)
```

Since the error message points us to line 4 this might be a bit confusing. If you look at line four carefully you will see that there is no problem with the syntax. So, in this case the next step should be to back up and look at the previous line. In this case if you look at line 2 carefully you will see that there is a missing right parenthesis at the end of the line. Remember that parenthses must be balanced. Since Python allows statements to continue over multiple lines inside parentheses python will continue to scan subsequent lines looking for the balancing right parenthesis. However in this case it finds the name `current_time_int` and it will want to interpret that as another parameter to the input function. But, there is not a comma to separate the previous string from the variable so as far as Python is concerned the error here is a missing comma. From your perspective its a missing parenthesis.

Finding Clues How can you help yourself find these problems? One trick that can be very valuable in this situation is to simply start by commenting out the line number that is flagged as having the error. If you comment out line four, the error message now changes to point to line 5. Now you ask yourself, am I really that bad that I have two lines in a row that have errors on them? Maybe, so taken to the extreme, you could comment out all of the remaining lines in the program. Now the error message changes to `TokenError: EOF in multi-line statement` This is a very technical way of saying that Python got to the end of file (EOF) while it was still looking for something. In this case a right parenthesis.

Question Answer

Find and fix the error in the following code.

Save & Run 2/1/2023, 11:56:58 PM - 4 of 4 Show CodeLens

```
1 current_time_str = input("What is the current time (in hours 0-23
2 wait_time_str = input("How many hours do you want to wait")
3
4 current_time_int = int(10)
5 wait_time_int = int(5)
6
7 final_time_int = current_time_int + wait_time_int
8 print(final_time_int)
9
```

Activity: 3.4.1.2 ActiveCode (db_ex3_7)

Finding Clues If you follow the same advice as for the last problem, comment out line one, you will immediately get a different error message. Here's where you need to be very careful and not panic. The error message you get now is: `NameError: name 'current_time_str' is not defined on line 4`. You might be very tempted to think that this is somehow related to the earlier problem and immediately conclude that there is something wrong with the variable name `current_time_str` but if you reflect for a minute you will see that by commenting out line one you have caused a new and unrelated error. That is you have commented out the creation of the name `current_time_str`. So of course when you want to convert it to an `int` you will get the NameError. Yes, this can be confusing, but it will become much easier with experience. It's also important to keep calm, and evaluate each new clue carefully so you don't waste time chasing problems that are not really there.

Uncomment line 1 and you are back to the ParseError. Another track is to eliminate a possible source of error. Rather than commenting out the entire line you might just try to assign `current_time_str` to a constant value. For example you might make line one look like this: `current_time_str = "10"`

(BeginningtipsforDebugging.html) (Summary.html)

`#input("What is the "current time" (in hours 0-23)?")`. Now you have assigned `current_time_str` to the string 10, and commented out the input statement. And now the program works! So you conclude that the problem must have something to do with the input function.

3.4.2. TypeError

TypeErrors occur when you try to combine two objects that are not compatible. For example you try to add together an integer and a string. Usually type errors can be isolated to lines that are using mathematical operators, and usually the line number given by the error message is an accurate indication of the line.

Here's an example of a type error created by a Polish learner. See if you can find and fix the error.

Save & Run Original - 1 of 1 Show CodeLens

```
a = input('wpisz godzine')
x = input('wpisz liczbe godzin')
int(x)
int(a)
h = x // 24
s = x % 24
print (h, s)
a = a + s
print ('godzina teraz', a)

```

Activity: 3.4.2.1 ActiveCode (db_ex3_8)

Show me the Solution

Finding Clues One thing that can help you in this situation is to print out the values and the types of the variables involved in the statement that is causing the error. You might try adding a print statement after line 4 `print(x, type(x))` You will see that at least we have confirmed that x is of type string. Now you need to start to work backward through the program. You need to ask yourself, where is x used in the program? x is used on lines 2, 3, and of course 5 and 6 (where we are getting an error). So maybe you move the print statement to be after line 2 and again after 3. Line 3 is where you expect the value of x to be changed to an integer. Could line 4 be mysteriously changing x back to a string? Not very likely. So the value and type of x is just what you would expect it to be after line 2, but not after line 3. This helps you isolate the problem to line 3. In fact if you employ one of our earlier techniques of commenting out line 3 you will see that this has no impact on the error, and is a big clue that line 3 as it is currently written is useless.

3.4.3. NameError

Name errors almost always mean that you have used a variable before it has a value. Often NameErrors are simply caused by typos in your code. They can be hard to spot if you don't have a good eye for catching spelling mistakes. Other times you may simply mis-remember the name of a variable or even a function you want to call. You have seen one example of a NameError at the beginning of this section. Here is another one. See if you can get this program to run successfully:

Save & Run Original - 1 of 1 Show CodeLens

```
str_time = input("What time is it now?")
str_wait_time = input("What is the number of nours to wait?")
time = int(str_time)
wai_time = int(str_wait_time)

time_when_alarm_go_off = time + wait_time
print(time_when_alarm_go_off)

```

(BeginningtipsforDebugging.html)

Activity: 3.4.3.1 ActiveCode (db_ex3_9)

(Summary.html)

Show me the Solution

Finding Clues With name errors one of the best things you can do is use the editor, or browser search function. Quite often if you search for the exact word in the error message one of two things will happen:

1. The word you are searching for will appear only once in your code, it's also likely that it will be on the right hand side of an assignment statement, or as a parameter to a function. That should confirm for you that you have a typo somewhere. If the name in question **is** what you thought it should be then you probably have a typo on the left hand side of an assignment statement on a line before your error message occurs. Start looking backward at your assignment statements. In some cases it's really nice to leave all the highlighted strings from the search function visible as they will help you very quickly find a line where you might have expected your variable to be highlighted.
2. The second thing that may happen is that you will be looking directly at a line where you expected the search to find the string in question, but it will not be highlighted. Most often that will be the typo right there.

Here is another one for you to try:

Save & Run Original - 1 of 1 Show CodeLens

```
n = input("What time is it now (in hours)?")
n = imt(n)
m = input("How many hours do you want to wait?")
m = int(m)
q = m % 12
print("The time is now", q)

```

Activity: 3.4.3.2 ActiveCode (db_ex3_10)

Show me the Solution

And one last bit of code to fix.

Save & Run Original - 1 of 1 Show CodeLens

```
present_time = input("Enter the present timein hours:")
set_alarm = input("Set the hours for alarm:")
int (present_time, set_time, alarm_time)
alarm_time = present_time + set_alarm
print(alarm_time)

```

Activity: 3.4.3.3 ActiveCode (db_ex3_11)

Show me the Solution

3.4.4. ValueError

Value errors occur when you pass a parameter to a function and the function is expecting a certain limitations on the values, and the value passed is not compatible. We can illustrate that with this particular program in two different ways.

Save & Run Original - 1 of 1 Show CodeLens

```
current_time_str = input("What is the current time (in hours 0-23)?")
current_time_int = int(current_time_str)

wait_time_str = input("How many hours do you want to wait")
wait_time_int = int(wait_time_int)

```

(BeginningtipsforDebugging.html) (Summary.html)

```
final_time_int = current_time_int + wait_time_int
print(final_time_int)

```

Activity: 3.4.4.1 ActiveCode (db_ex3_12)

Run the program but instead of typing in anything to the dialog box just click OK. You should see the following error message: `ValueError: invalid literal for int() with base 10: '' on line: 4` This error is not because you have made a mistake in your program. Although sometimes we do want to check the user input to make sure its valid, but we don't have all the tools we need for that yet. The error happens because the user did not give us something we can convert to an integer, instead we gave it an empty string. Try running the program again. Now this time enter "ten" instead of the number 10. You will get a similar error message.

ValueErrors are not always caused by user input error, but in this program that is the case. We'll look again at ValueErrors again when we get to more complicated programs. For now it is worth repeating that you need to keep track of the restrictions needed for your variables, and understand what your function is expecting. You can do this by writing comments in your code, or by naming your variables in a way that reminds you of their proper form.

You have attempted 10 of 10 activities on this page

✔ **Completed. Well Done!**

Continue to page 5 of 5 in the reading assignment. (/ns/books/published/cmsc-210-spring-2023/debugging/Summary.html)

(BeginningtipsforDebugging.html) (Summary.html)