TASK

Defensive Programming - Error Handling

Visit our website

Introduction

WELCOME TO THE DEFENSIVE PROGRAMMING - ERROR HANDLING TASK!

You should now be quite comfortable with basic variable identification, declaration, and implementation. You should also be familiar with the process of writing basic code which adheres to correct Python formatting to create a running program. In this task, you'll be exposed to error handling and basic debugging to fix issues in your code, as well as the code of others — a skill that is extremely useful in a software development/programming career!

WHAT IS DEFENSIVE PROGRAMMING?

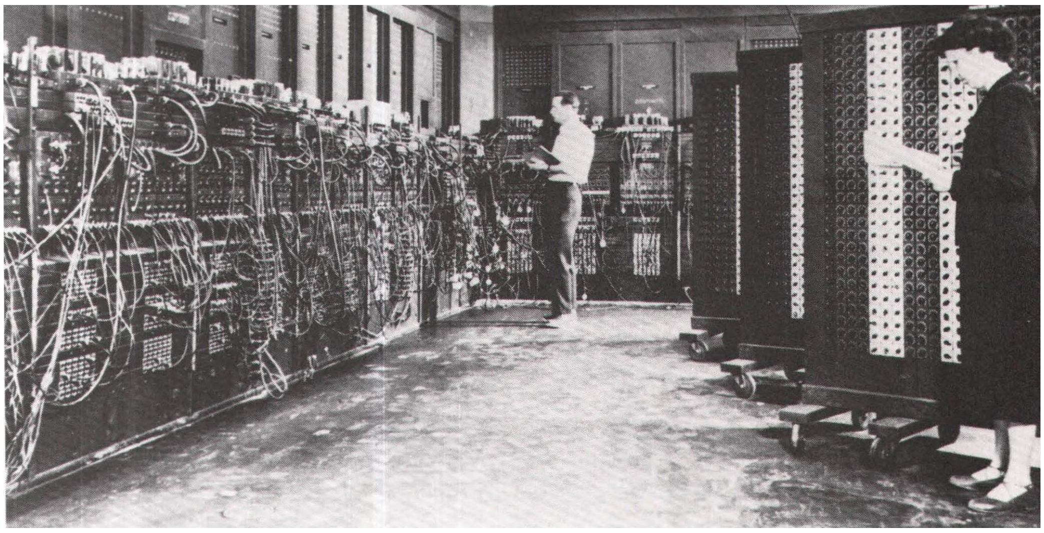
Defensive programming is an approach to writing code where the programmer tries to anticipate problems that could affect the program and then takes steps to defend the program against these problems. MANY problems could cause a program to run unexpectedly!

DEBUGGING

Debugging is something that you, as a software developer/programmer, will come to breathe, sleep, and live. Debugging is when a programmer looks through their code to try and identify a problem that is causing some error.

The word 'debugging' comes from the first computers, back when a computer was roughly the size of an entire room. Bugs (living insects) would get into the computer and cause the device to function incorrectly. The programmers would need to go in and remove the insects, hence the name - debugging.

One such computer is the ENIAC. It was the first true electronic computer and was located in the University of Pennsylvania.



An image of the ENIAC.

Computers we know them today are significantly smaller and also much more powerful than older computers like the ENIAC (despite their enormous size). It goes to show how, in just a few years, we can expand our computational power immensely. In terms of the rate of growth on computational power, Gordon Moore (co-founder of Fairchild Semiconductor and Intel (and former CEO of the latter)), predicted in the 1960s that the number of transistors in an integrated circuit would double about every two years, a prediction which has been borne out as correct over time. In the world of tech it is necessary to actively work to keep our knowledge and expertise current!

If you're interested, you can do some more research about the ENIAC on the University of Pennsylvania's website:
http:// www.seas.upenn.edu/about-seas/eniac/.

Software developers live by the motto: "try, try, try again!" Testing and debugging your code repeatedly is essential for developing effective and efficient code and achieving mastery as a developer/programmer.

DEALING WITH ERRORS

Everyone makes mistakes. Likely you've made some already. It's important to be able to DEBUG your code. You debug your code by removing errors from it.

Types of errors

There are three different categories of errors that can occur:

- **Syntax errors**: These errors are due to incorrect syntax used in the program (e.g. wrong spelling, incorrect indentation, missing parentheses, etc.). The compiler can detect such errors. If syntax errors exist when the program is compiled, the compilation of the program fails and the program is terminated. Syntax errors are also known as compilation errors. They are the most common type of error.

- **Runtime errors**: Runtime errors occur during the execution of your program. Your program will compile, but the error occurs while the program is running. An error message will likely also pop up when trying to run the buggy program. Examples of what might cause a runtime error include dividing by zero, and referencing an out of range list item (you haven't learned about lists yet but soon will - if you're excited about them and want to get a sneak peak, have a look **here**!).

- **Logical errors:** These occur when your program's syntax is correct, and it runs and compiles, but the output is not what you are expecting. This means that the logic you applied to the problem contains an error. Logical errors can be the most difficult errors to spot and resolve, because you need to identify where you have gone wrong in your thought process. Building **metacognitive skills** can help you gain proficiency in detecting logical errors.

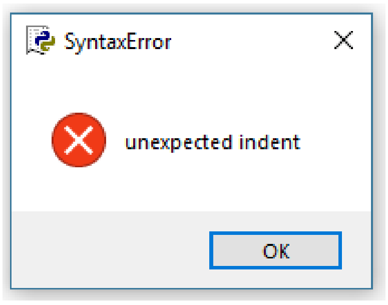
Syntax errors

Syntax errors are comparable to making spelling or grammar mistakes when writing in English (or any other language), except that a computer requires perfect syntax to run correctly.

Common python syntax errors include:

- Leaving out a keyword or putting it in the wrong place
- Misspelling a keyword (for example a function or variable name)
- Leaving out an important symbol (such as a colon, comma, or parentheses)
- Incorrect indentation
- Leaving an empty block (for example, an if statement containing no indented statements)

Below is a typical example of a compilation error message. Compilation errors are the most common type of error in Python. They can get a little complicated to fix when dealing with loops and if statements.



```
unexpected indent
```

When a syntax error occurs, the line in which the error is found will often be highlighted (although exactly how depends on your IDE). The cursor may also automatically be placed at the point the error occurred to make the error easy to identify. However, watch out - this can be misleading! For example, if you leave out a symbol, such as a closing bracket or quotation mark, the error message could refer to a place later on in the code, which is not the actual source of the problem.

Go to the line indicated by the error message and correct the error, then try to compile your code again. If you cannot see an error on that line, see if you have made any syntax errors on previous lines that could be the source of the problem.

Debugging is an essential part of being a programmer, and although sometimes frustrating, it can be quite fun to problem-solve the causes of your errors!

Runtime errors

Using your knowledge of strings, take a look at the code below and see if you can identify the runtime error:

```python
greeting = "Hello!"
print(greeting[6])
```

This would be the error you get, but why?

```
Exception has occurred: IndexError
string index out of range
```

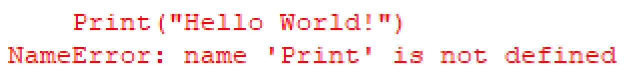
Look carefully at the description of the error. It must have something to do with the string index. If you recall from Task 5, we number the index from 0, and so the final index for “!” would be 5. That means that nothing exists for index 6, so we get a runtime error. It's important to read error messages carefully and think in a deductive way to solve them. It may at times be useful to copy and paste the error message into Google to figure out how to fix the problem.

Logical errors

You are likely to encounter logical errors, especially when dealing with loops and control statements. Even after years of programming, it takes time to sit down and design an algorithm. Finding out why your program isn't working takes time and effort but becomes easier with practice and experience. It’s also worth noting that taking the time to plan your program using pseudo code significantly reduces the incidence of logical errors.

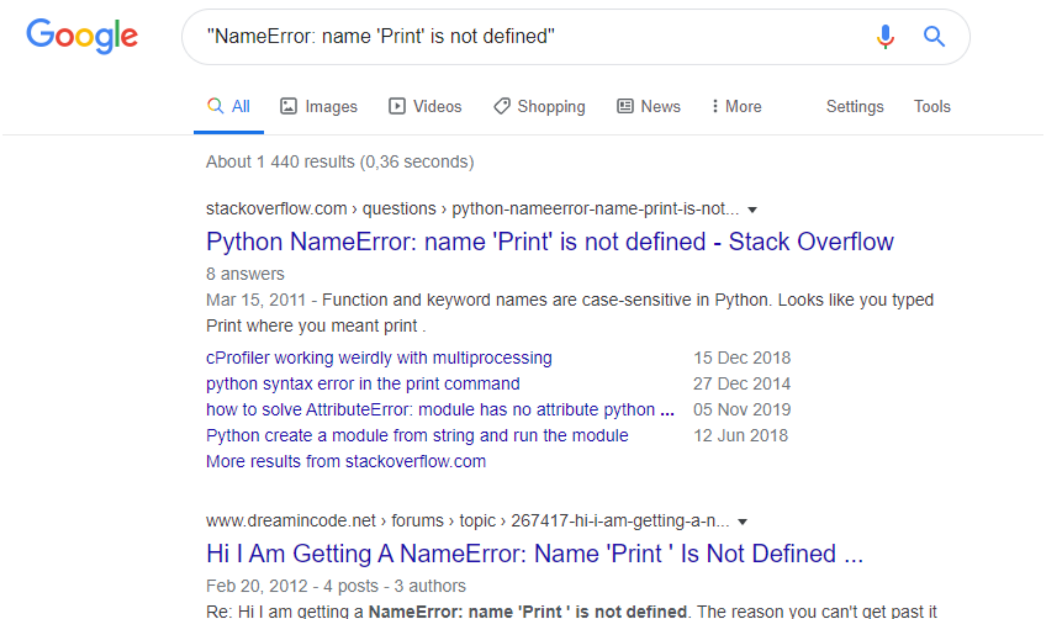
TIPS FOR DEBUGGING

You have probably seen plenty of errors similar to the one shown in the image below in the Python shell.



```
Print("Hello World!")
NameError: name 'Print' is not defined
```

Many of the errors are quite easy to identify and correct. However, often you may find yourself thinking, “What does that mean?”. One of the easiest, and often most effective, ways of dealing with error messages that you don’t understand, is to simply copy and paste the error into Google. Many forums and websites are available that can help you to identify and correct errors easily. **Stack Overflow** is an excellent resource that software developers around the world use to get help.



About 1 440 results (0,36 seconds)

stackoverflow.com › questions › python-nameerror-name-print-is-not...

Python NameError: name 'Print' is not defined - Stack Overflow

8 answers

Mar 15, 2011 - Function and keyword names are case-sensitive in Python. Looks like you typed Print where you meant print .

cProfiler working weirdly with multiprocessing	15 Dec 2018
python syntax error in the print command	27 Dec 2014
how to solve AttributeError: module has no attribute python ...	05 Nov 2019
Python create a module from string and run the module	12 Jun 2018

More results from stackoverflow.com

www.dreamincode.net › forums › topic › 267417-hi-i-am-getting-a-n...

Hi I Am Getting A NameError: Name 'Print ' Is Not Defined ...

Feb 20, 2012 - 4 posts - 3 authors

Re: Hi I am getting a **NameError: name 'Print ' is not defined**. The reason you can't get past it

LLMs like ChatGPT can also be really helpful in debugging tricky errors!

In addition, your IDE may have some other built-in features that can also help you identify and correct subtler errors. Find out if it has a debugger and how to turn it on if it is not on by default.

Many debuggers will allow you to do things like “Go”, “Step”, etc. This part of debugger functionality is especially important if you use breakpoints in your code. A breakpoint is a point in your code where you tell it to pause the execution of the code so that you can check what is going on. For example, if you don’t understand why an *if statement* isn’t doing what you think it should, you could create a breakpoint at the *if statement*. You can follow this **video** to learn how to set breakpoints in VSCode.

Instructions

- First, read and run the **example files** provided. Feel free to write and run your own example code before doing the compulsory task to become more comfortable with the concepts covered in this task.

Compulsory Task 1

For both the **errors.py** and **errors2.py** task files in your folder, open the files and follow these steps:

- Attempt to run the program. You will encounter various errors.
- Fix the errors and then run the program.
- Each time you fix an error, add a **# comment** in the line where you fixed it and indicate which of the three types of errors it was with a brief explanation of why that is.
- Save the corrected file.

Compulsory Task 2

Follow these steps:

- Create a new Python file called **logic.py**.
- Write a program that displays a logical error (be as creative as possible!).

Optional Bonus Task

Follow these steps:

- Create a new Python file in this folder called **optional_task.py**.
- Write a program with two compilation errors, a runtime error and a logical error.
- Next to each error, add a comment that explains what type of error it is and why it occurs.

Rate us

Share your thoughts

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we’ve done a good job?

Click here to share your thoughts anonymously.