GERMANNA
ACADEMIC CENTER
FOR EXCELLENCE

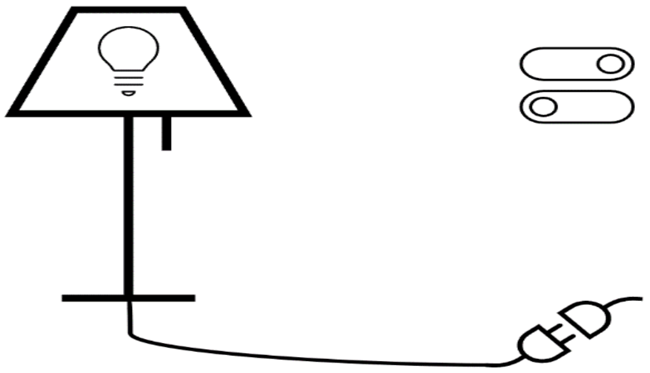
Coding Errors: Debugging and Troubleshooting Techniques

Problem solving is a fundamental skill required for programming that involves the development of an organized approach to identifying problems, analyzing probable causes, determining solutions, and testing solutions. Students can apply this process when debugging and troubleshooting coding errors. This handout will demonstrate the process of troubleshooting by examining a specific situation: an inoperable lamp.

Step 1: Identify the problem.

The first critical step, in the troubleshooting process, is to identify the problem. It narrows the focus for making corrections, and it provides a clear goal.

Example: The lamp will not turn on.



Step 2: Understand the situation.

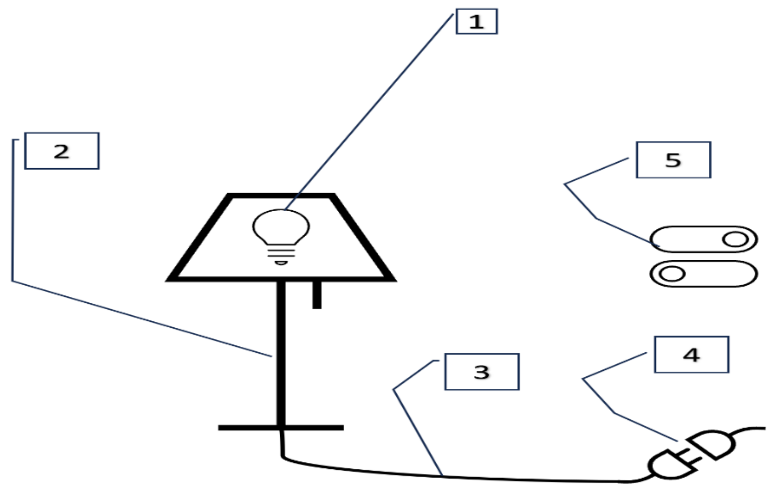
During this stage, it is important to gather relevant information to find potential variables that could be causing the problem. By doing this, the issue can be broken into smaller components, making it easier to identify and apply a solution. The following example includes a series of questions that can be applied to the situation.

Example:

1. Is the cord from the lamp to the wall damaged?
2. Is the cord plugged into the wall?
3. Is there a light bulb inside the lamp? Is it secured tightly?
4. Is the wall switch flipped to "on?"

Step 3: Define the goals.

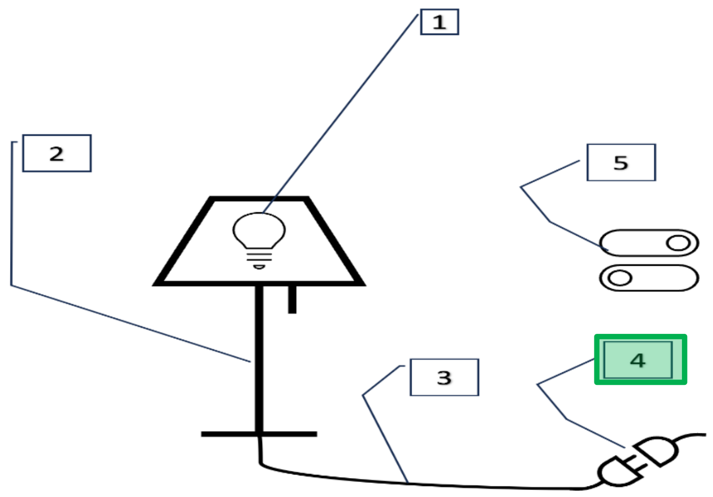
The goal of troubleshooting is to resolve a problem. In this example, the goal is to have the lamp turned on and functioning properly. Any other information that will not fix the problem does not matter, such as the color of the bulb.



Step 4: Generate probable causes.

After contemplating several scenarios of what might be causing the problem, propose steps that will lead to a potential solution.

1. The light bulb may be missing or not working.
   - Test the light bulb to ensure it works by putting it in a lamp/socket that is known to be functional.
2. The lamp itself may be turned off or damaged.
   - Check the switch to ensure it is set to “on.”
   - Examine the lamp for any visible damage to the switch, socket, or wire.
3. The cord may be broken.
   - Examine the cord for any damage.
4. The lamp may not be plugged in
   - Check to see if the cord is plugged into the wall socket.
5. The wall switch may be turned off.
   - Check to see if the wall switch is set to "on.”



Step 5: Test potential solutions starting with the best option first.

During the last step of troubleshooting, not all the scenarios in the previous step may need to be resolved. In the diagram above, scenario #4, “The lamp may not be plugged in,” appears to be the problem and, therefore, the best place to begin. The solution is to plug the lamp into the wall socket. Another tip is to start with the simplest solution first instead of a more complex solution.

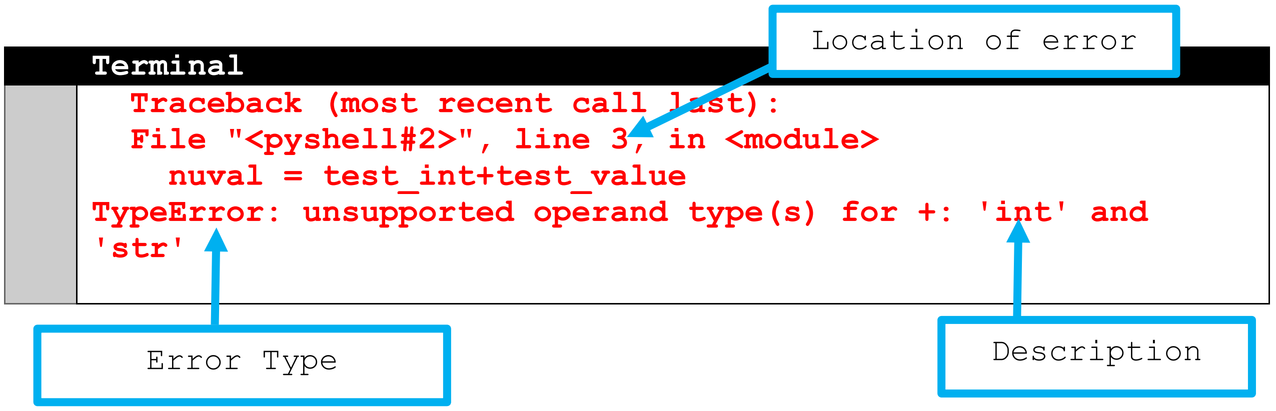
Applying Troubleshooting to Debugging Code

The process of troubleshooting an inoperable lamp is similar to debugging code. While troubleshooting finds the root of a problem, debugging is the process of fixing errors, often referred to as “bugs,” in code.

Unlike the beginning process of troubleshooting, where the problem must be identified, a program will display a message identifying the error. These messages are written in red and provide the location of the error, the type of error, and a brief description of why the error occurred. Within a printed black and white copy of this handout, the error has been identified to the reader in the terminal section.

Example:

	Console
1	`test_value = "Hello"`
2	`test_int= 100`
3	`nuval = test_int+test_value`



```
Terminal
  Traceback (most recent call last):
  File "<pyshell#2>", line 3, in <module>
    nuval = test_int+test_value
TypeError: unsupported operand type(s) for +: 'int' and
'str'
```

In this example of Python code, the message shows that the error has occurred on line 3. The reason for the error is that the user is trying to add incompatible data types, “integers” and “strings.”

Example:
Returning to the lamp scenario, an error message might resemble:

```
Terminal

ERROR: No power at bulb
```

This would identify the location of the problem but provide no information about why there is no power reaching the bulb. Often when coding, fixing a portion of code can lead to other error messages that further break down the debugging process, such as:

```
Terminal

ERROR: No Power at bulb
ERROR: Switch set to 'off'
```

The error message indicates that the light bulb is not turning on, and the switch is set to “off.”

Tip: Error messages vary in style and function between different programming languages. It is critical to understand the programming language being used to understand the relationship between each occurrence and use the tools provided to fix the problem.

Troubleshooting and debugging skills are crucial to problem solving. By isolating the issue, testing each possibility, and finding solutions, students build a greater skill set for programming. Applying these skills develops a foundation for critical thinking which can be applied to fixing more complex error scenarios. For more information about problem solving, please refer to the Computer Programming Problem Solving Process handout.