Homework 6 - Nihal Wadhwa

Problem 1: Caesar Cipher

1. Give 2 examples of inputs for which the provided code gives a correct answer despite the fact that it is flawed.

Example 1: caesar("very", "aafd")

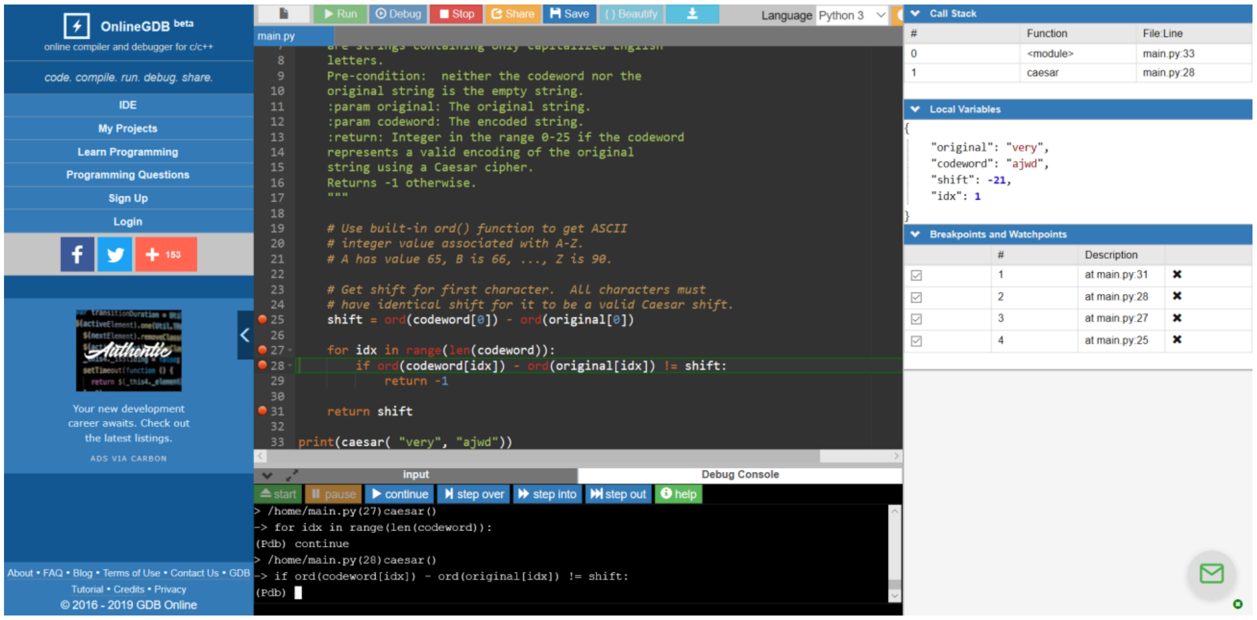
Example 2: caesar ("nihal", "gbasa")

2. For each example, explain why the faulty code produced the correct answer, despite the flaw(s).

Example 1 and 2 return the correct answer because the code itself calculates the shift and the tests the shift for each character incorrectly as well, which will always result in the return of -1 for any input. These examples were meant to return -1, therefore the code seems like it works.

3. Describe the bug(s) present in the code, and for each bug, indicate what the fix is.

The very first bug I encountered was the incorrect calculation of the shift, which would sometimes come out to be negative, which would make the if statement ineffective because the if statement would be checking the characters with the incorrect shift. To fix this, I added an if statement that would add 26 to shift if it was negative so that the shift was corrected. Next issue I found is that the if statement wasn't executing correctly because the shift can occur in two ways. For example, a shift of 5 is also equivalent to a shift of -21, however the if statement would only check one of the two possible outcomes. So, I added an additional condition to the if statement so that the code would check to see if the shift is equivalent to the positive or negative variation of the shift. After solving this issue, the code worked perfectly for all cases.



```python
    are strings containing only capitalized English
    letters.
    Pre-condition:  neither the codeword nor the
    original string is the empty string.
    :param original: The original string.
    :param codeword: The encoded string.
    :return: Integer in the range 0-25 if the codeword
    represents a valid encoding of the original
    string using a Caesar cipher.
    Returns -1 otherwise.
    """

    # Use built-in ord() function to get ASCII
    # integer value associated with A-Z.
    # A has value 65, B is 66, ..., Z is 90.

    # Get shift for first character.  All characters must
    # have identical shift for it to be a valid Caesar shift.
    shift = ord(codeword[0]) - ord(original[0])

    for idx in range(len(codeword)):
        if ord(codeword[idx]) - ord(original[idx]) != shift:
            return -1

    return shift

print(caesar( "very", "ajwd"))
```

```
> /home/main.py(27)caesar()
-> for idx in range(len(codeword)):
(Pdb) continue
> /home/main.py(28)caesar()
-> if ord(codeword[idx]) - ord(original[idx]) != shift:
(Pdb)
```

Call Stack

#	Function	File:Line
0	<module>	main.py:33
1	caesar	main.py:28

Local Variables

```
{
    "original": "very",
    "codeword": "ajwd",
    "shift": -21,
    "idx": 1
}
```

Breakpoints and Watchpoints

	#	Description	
☑	1	at main.py:31	✖
☑	2	at main.py:28	✖
☑	3	at main.py:27	✖
☑	4	at main.py:25	✖

Problem 2: Longest Consecutive Matching Substring

1. Give 2 examples of inputs for which the provided code gives a correct answer despite the fact that it is flawed.

Example 1: match("established", "ballistic")

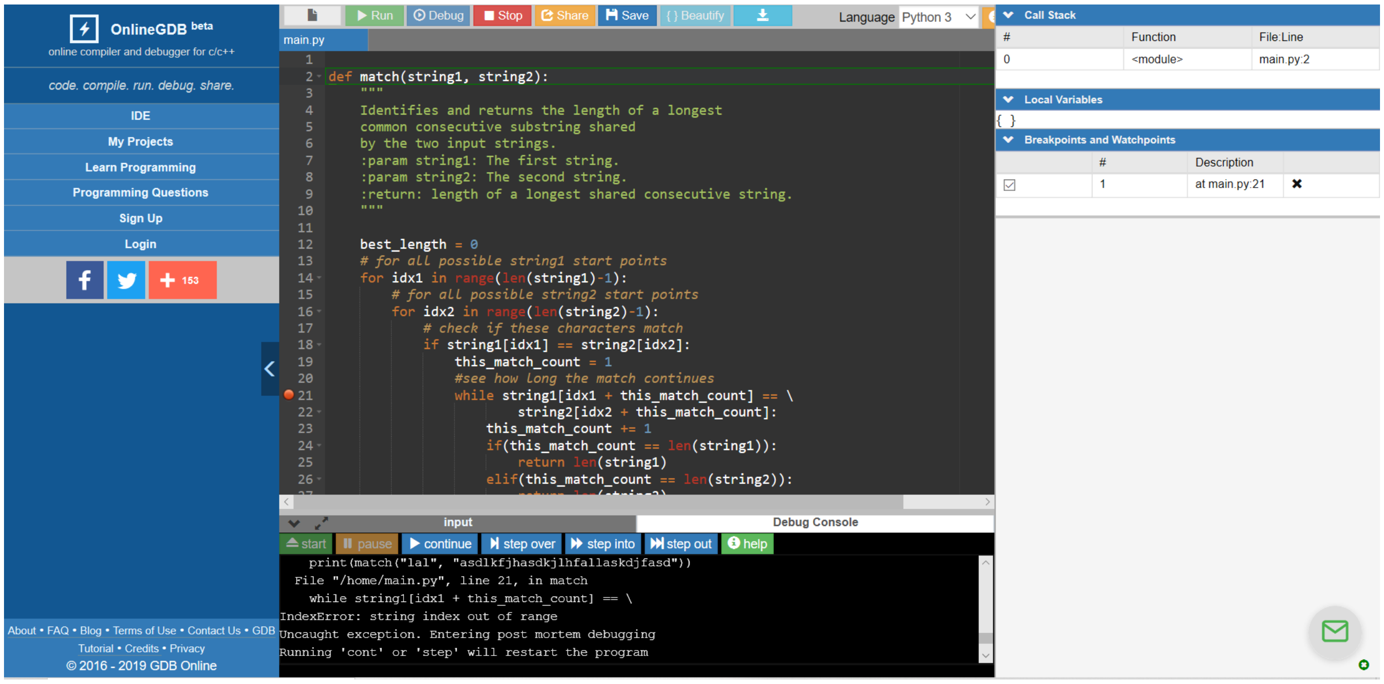
Example 2: match(“clapped”, “flappers”)

2. For each example, explain why the faulty code produced the correct answer, despite the flaw(s).

These produced the correct outcome because the consecutive string that was found in both parameters were in the middle, but the error only occurred in the code when the string that was found in both the strings is present at the end of the second string. Since, in these examples, the consecutive string was in the middle of the string of the second parameter, the code executed correctly.

3. Describe the bug(s) present in the code, and for each bug, indicate what the fix is.

The main issue with this code is that the while statement and incrementation of this_match_count was continuing to return the error of index out of range. In order to fix this error, I completely redid the while statement so that the code would detect this error and break out of the loop while successfully returning the best length.



```python
def match(string1, string2):
    """
    Identifies and returns the length of a longest
    common consecutive substring shared
    by the two input strings.
    :param string1: The first string.
    :param string2: The second string.
    :return: length of a longest shared consecutive string.
    """

    best_length = 0
    # for all possible string1 start points
    for idx1 in range(len(string1)-1):
        # for all possible string2 start points
        for idx2 in range(len(string2)-1):
            # check if these characters match
            if string1[idx1] == string2[idx2]:
                this_match_count = 1
                #see how long the match continues
                while string1[idx1 + this_match_count] == \
                        string2[idx2 + this_match_count]:
                    this_match_count += 1
                    if(this_match_count == len(string1)):
                        return len(string1)
                    elif(this_match_count == len(string2)):
```

```
    print(match("lal", "asdlkfjhasdkjlhfallaskdjfasd"))
  File "/home/main.py", line 21, in match
    while string1[idx1 + this_match_count] == \
IndexError: string index out of range
Uncaught exception. Entering post mortem debugging
Running 'cont' or 'step' will restart the program
```

Call Stack

#	Function	File:Line
0	<module>	main.py:2

Local Variables

{ }

Breakpoints and Watchpoints

	#	Description	
✓	1	at main.py:21	✖

About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy

© 2016 - 2019 GDB Online