

```
def tests():
    print('The shift for ABCD and BCDE is', caesar('ABCD', 'BCDE'))

    print('The shift for WXYZ and BCDE is', caesar('WXYZ', 'BCDE'))

    print('The shift for ABC and XYZ is', caesar('ABC', 'XYZ'))

    print('The shift for ABCD and ZBA is', caesar('ABCD', 'ZAB'))

    print('The shift for BDED and ACDC is', caesar('BDED', 'ACDC'))

    print('The shift for 123 and ABC is', caesar('123', 'ABC'))
```

Test functions and their results.

```
The shift for ABCD and BCDE is 1
The shift for WXYZ and BCDE is -21
The shift for ABC and XYZ is 23
The shift for ABCD and ZBA is False
The shift for BDED and ACDC is -1
The shift for 123 and ABC is False
```

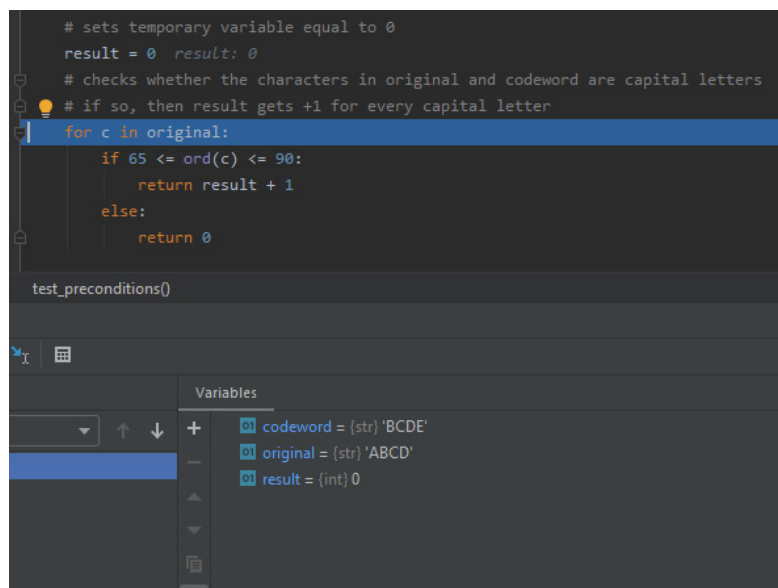
Important Functions with correct yield:

WXYZ and BCDE are represented by 21 letters to the left rather than 5 to the left.

ABCD and ZBA are represented by False, since the cipher is not consistent.

123 and ABC are represented by False since you cannot compare integers to strings.

Bugs and fixes



```
# sets temporary variable equal to 0
result = 0
# checks whether the characters in original and codeword are capital letters
# if so, then result gets +1 for every capital letter
for c in original:
    if 65 <= ord(c) <= 90:
        return result + 1
    else:
        return 0
```

test_preconditions()

Variables

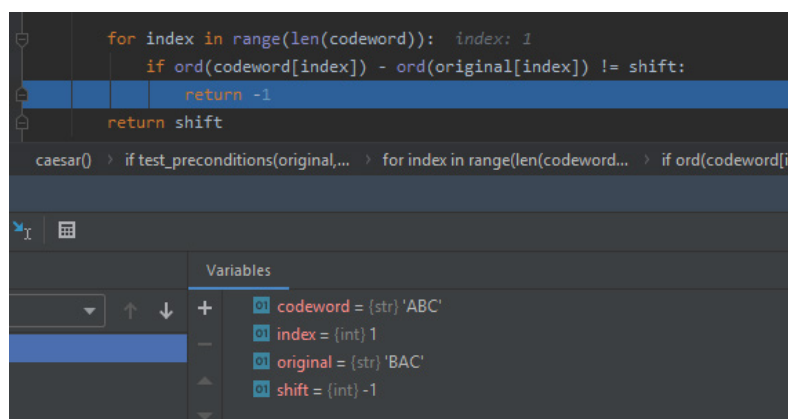
- codeword = {str} 'BCDE'
- original = {str} 'ABCD'
- result = {int} 0

Bug:

Returning 'result + 1' and '0' caused the program to crash.

Fix:

Simply updating the result as 'result + 1' if the parameters were met and 'result = 0' when they were not fixed the program.



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

caesar() > if test_preconditions(original,... > for index in range(len(codeword... > if ord(codeword[i

Variables

- codeword = {str} 'ABC'
- index = {int} 1
- original = {str} 'BAC'
- shift = {int} -1

Bug:

Returning -1 caused the program to return a stub rather than the actual shift.

Fix:

Return False rather than -1. Returning False is more descriptive and represents that there is no cipher.

```
def tests():  
    print('The longest substring is', match('abcdefg', 'hijkl'), 'characters long.')  
  
    print('The longest substring is', match('something', 'somewhere'), 'characters long.')  
  
    print('The longest substring is', match('phone', 'mouse'), 'characters long.')  
  
    print('The longest substring is', match('phone', 'three'), 'characters long.')
```

```
The longest substring is 0 characters long.  
The longest substring is 4 characters long.  
The longest substring is 1 characters long.  
The longest substring is 1 characters long.
```

Test functions along with results

- 1) abcdefg and hijkl have no characters in common, therefore the function returned 0
- 2) something and somewhere share the word 'some' with each other, therefore 4 was returned
- 3) phone and mouse have just the letter 'o' in common. therefore 1 was returned.
- 4) phone and three have both 'h' and 'e' in common, yet they are not consecutive, therefore the function only returned 1