**1.Print statements for checking that variables have the expected values**

(1). In Line B, assign s[:m-1] to first instead of s[:m]. Then, in the command shell, run

Hyphenator even.py. (That is, type python Hyphenator even.py in the command shell. Don’t

forget to save the file in Pycharm Edit first.) Give the input 'abcdef'.

* 1. Which variable has an unexpected value, and what should the variable’s value have been?

Unexpected value:

First is ab

Exped one:

First is abc

* 1. Which print line tells you this?

Line B

(2). Fix the mistake you made above. Now, introduce a new artificial mistake via Komodo Edit:

in Line C, assign s[m+1:] to second instead of s[m:]. Suppose you run Hyphenator even.py and

give it input 'abcdef'.

2.1) Which variable has an unexpected value, and what should the variable’s value have been?

Unexpected value:

second is ef

Exped one:

second is def

2.2) Which print line tells you this?

Line C

1. **Finding errors using print statements**

when you run the program in the command shell, you know what lines are incorrect.

Write in the blank spaces of the code above what print statements you included.

Run your altered program in the command shell. From the output of your print statements,

explain what are the errors in the program.

Read the docstring for Hyphenator broken.py (the comment in triple quotes) in the following

program to understand what the program is supposed to do.

# Hyphenator\_broken.py

# the CSC 7014

# Jan 2024

""" Inserts hyphens into a non-empty odd-length input string as follows:

A hyphen is inserted on either side of the middle character.

Example: "abcde" becomes "ab-c-de"

"""

### This program intentionally has at least one error in it!

s = input('Enter an odd-length string (remember to put quotes around it): ')

n = len(s)

m = n/2

first = s[0:m-1]

middle = s[m+1]

second = s[m+1:]

h = first+'-'+middle+'-'+second

# final output

print s,'becomes',h

**updated code:**

s = input('Enter an odd-length string (remember to put quotes around it): ')

n = len(s)

m = n//2

first = s[0:m]

middle = s[m]

second = s[m+1:]

h = first+'-'+middle+'-'+second

# final output

print(s,'becomes',h)

The errors I found in this code are:

1. At the content **m=n/2**

This will return a float value later in the code this reference is used as an index it will throw an error

1. Indexes for first and middle gets chage to **m**
2. Th syntax of the print statement

Other than than this the code is clean

1. **Boolean Expressions**

3.1. Numerical Examples. Assume that x,y and z are initialized as follows

>>> x = 1

>>> y = 5

>>> z = 10

Now complete this table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Expression** | **I think the value is** | **Python says** | **Notes** |
| **x<z** | **True** | **True** |  |
| **2\*y>=z** | True | True |  |
| **2\*y<z** | False | False |  |
| **(x>1) or(z!=7)** | True | True |  |
| **y!=(z/2)** | False | False |  |
| **(x>0)or((y>0)or(z<0))** | True | True |  |
| **((x>0)or(y>0))or(z<0)** | True | True |  |

3.2. String Examples. Assume that x, y, and z are initialized as follows

>>> x = 'Cornell'

>>> y = 'Harvard'

>>> z = 'Yale'

Now complete this table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Expression** | **I think the value is** | **Python says** | **Notes** |
| **x!=z** | **True** | **True** |  |
| **x==’cornell’** | False | False |  |
| **len(x)>len(y)** | False | False |  |
| **y[1:]>z[1:]** | True | True |  |
| **len(x+z)>len(y)** | True | True |  |

**4. The If-Else Construction**

a) i) s='apples'

if(s[-2:]=='es'):

print('Plural')

else:

print('Not Plural')

ii) s='apples'

if(s[-2:]=='ss'):

print('Plural')

else:

print('Not Plural')

By assigning s as “apples”

the initial output is “Plural”

After changing comparing string to “ss”

The output is “Not Plural”

**b)**

s='1246'

if(len(s)%2==0):

print(3\*int(s[-2:]))

else:

print(7\*int(s[len(s)//2]))

By assigning s as “12945”

the initial output is “63”

After changing s to “1246”

The output is “138”

**5. Pretty printing**

1. In other words, center the numbers under the column headings, display the Quotient through seven decimal places, and display the Error with three significant digits. Do this by playing with theformat specifications and blanks in the string ’%3d %3d %22.15f %10.6e’. What does your new print statement look like?

'%3d %3d %9.7f %1.2e'

1. Change the first print statement to

print '\n\n\n\nNumerator Denominator Quotient Error'

What does \n seem to do?

\n gives a line space in the output terminal.

**6.IF-Elif-Else**

Code is:

x=int(input('Enter x :'))

y=int(input('Enter y :'))

if(1<=x<=3 and 1<=y<=3):

    print('A')

elif(x>3):

    print('B')

elif(y<1):

    print('C')

elif(y>3):

    print('D')

else:

    print('E')

Complete the following table:

|  |  |  |
| --- | --- | --- |
| x | y | Output |
| 2 | 2 | A |
| 1 | 0 | C |
| 4 | 1 | B |
| 1 | 5 | D |
| 0 | 3 | E |
| 3 | 0 | C |