Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Exercise 1

Name: Chuay Xiang Ze

IC Number: 021224070255

Date : 22/06/2023

Introduction : Learning how to implement slicing when doing for loops

Conclusion : Managed to solve problems with the codes I've learnt.

EXERCISE 1

RUN ME

Please run the code snippet below. It is required for running tests for your solution.

In [9]:

```
def test(got, expected):
    if got == expected:
        prefix = ' OK '
    else:
        prefix = '  FAIL '
    print(('%s got: %s expected: %s' % (prefix, repr(got), repr(expected))))
```

Question 1

In [10]:

```python
# A. donuts
# Given an int count of a number of donuts, return a string
# of the form 'Number of donuts: <count>', where <count> is the number
# passed in. However, if the count is 10 or more, then use the word 'many'
# instead of the actual count.
# So donuts(5) returns 'Number of donuts: 5'
# and donuts(23) returns 'Number of donuts: many'
def donuts(count):
    #++ your code here ++

    if count >= 10:
        num = "many"
    else:
        num = str(count)

    return "Number of donuts: " + num


print('donuts')
# Each line calls donuts, compares its result to the expected for that call.
test(donuts(4), 'Number of donuts: 4')
test(donuts(9), 'Number of donuts: 9')
test(donuts(10), 'Number of donuts: many')
test(donuts(99), 'Number of donuts: many')
```

```
donuts
 OK  got: 'Number of donuts: 4' expected: 'Number of donuts: 4'
 OK  got: 'Number of donuts: 9' expected: 'Number of donuts: 9'
 OK  got: 'Number of donuts: many' expected: 'Number of donuts: many'
 OK  got: 'Number of donuts: many' expected: 'Number of donuts: many'
```

Question 2

In [16]:

```
# B. both_ends
# Given a string s, return a string made of the first 2
# and the last 2 chars of the original string,
# so 'spring' yields 'spng'. However, if the string length
# is less than 2, return instead the empty string.
def both_ends(s):
    if len(s) < 2:
        return ''

    first_two = s[:2]
    last_two = s[-2:]
    return first_two + last_two

print('both_ends')
test(both_ends('spring'), 'spng')
test(both_ends('Hello'), 'Helo')
test(both_ends('a'), '')
test(both_ends('xyz'), 'xyyz')
```

```
both_ends
 OK  got: 'spng' expected: 'spng'
 OK  got: 'Helo' expected: 'Helo'
 OK  got: '' expected: ''
 OK  got: 'xyyz' expected: 'xyyz'
```

Question 3

In [15]:

```
# C. fix_start
# Given a string s, return a string
# where all  occurences of its first char have
# been changed to '*', except do not change
# the first char itself.
# e.g. 'babble' yields 'ba**le'
# Assume that the string is length 1 or more.
# Hint: s.replace(stra, strb) returns a version of string s
# where all instances of stra have been replaced by strb.
def fix_start(s):
    first_char = s[0]
    modified_string = s.replace(first_char, '*')
    modified_string = first_char + modified_string[1:]
    return modified_string

print('fix_start')
test(fix_start('babble'), 'ba**le')
test(fix_start('aardvark'), 'a*rdv*rk')
test(fix_start('google'), 'goo*le')
test(fix_start('donut'), 'donut')
```

```
fix_start
 OK  got: 'ba**le' expected: 'ba**le'
 OK  got: 'a*rdv*rk' expected: 'a*rdv*rk'
 OK  got: 'goo*le' expected: 'goo*le'
 OK  got: 'donut' expected: 'donut'
```

Question 4

In [19]:

```
def mix_up(a, b):
    mixed_a = b[:2] + a[2:]
    mixed_b = a[:2] + b[2:]
    return mixed_a + ' ' + mixed_b

print('mix_up')
test(mix_up('mix', 'pod'), 'pox mid')
test(mix_up('dog', 'dinner'), 'dig donner')
test(mix_up('gnash', 'sport'), 'spash gnort')
test(mix_up('pezzy', 'firm'), 'fizzy perm')
```

```
mix_up
 OK  got: 'pox mid' expected: 'pox mid'
 OK  got: 'dig donner' expected: 'dig donner'
 OK  got: 'spash gnort' expected: 'spash gnort'
 OK  got: 'fizzy perm' expected: 'fizzy perm'
```

In [ ]: