Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Exercise 2

Name: Justin Chong Mun Chen

IC Number: 960327-07-5097

Date : 21/6/2023

Introduction : Learning to familiarise with build-in functions in Python.

Conclusion : This practice has allowed me to use the build-in functions a lot better in my expressions.

EXERCISE 2

RUN ME

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

In [2]:
```
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 [2]:
```
# D. verbing
# Given a string, if its length is at least 3,
# add 'ing' to its end.
# Unless it already ends in 'ing', in which case
# add 'ly' instead.
# If the string length is less than 3, leave it unchanged.
# Return the resulting string.
def verbing(s):
   #++ your code here ++

    return s if len(s) < 3 else s + "ly" if "ing" in s else s + "ing"

print('verbing')
test(verbing('hail'), 'hailing')
test(verbing('swiming'), 'swimingly')
test(verbing('do'), 'do')
```

```
verbing
 OK  got: 'hailing' expected: 'hailing'
 OK  got: 'swimingly' expected: 'swimingly'
 OK  got: 'do' expected: 'do'
```

Question 2

In [3]:

```python
# E. not_bad
# Given a string, find the first appearance of the
# substring 'not' and 'bad'. If the 'bad' follows
# the 'not', replace the whole 'not'...'bad' substring
# with 'good'.
# Return the resulting string.
# So 'This dinner is not that bad!' yields:
# This dinner is good!
def not_bad(s):
    #++ your code here ++

    notIndex = s.find("not")
    badIndex = s.find("bad")

    if (notIndex == -1 and badIndex == -1 or badIndex < notIndex):
        return s

    return s.replace(s[notIndex : badIndex + 3], "good")


print()
print('not_bad')
test(not_bad('This movie is not so bad'), 'This movie is good')
test(not_bad('This dinner is not that bad!'), 'This dinner is good!')
test(not_bad('This tea is not hot'), 'This tea is not hot')
test(not_bad("It's bad yet not"), "It's bad yet not")
```

```
not_bad
 OK  got: 'This movie is good' expected: 'This movie is good'
 OK  got: 'This dinner is good!' expected: 'This dinner is good!'
 OK  got: 'This tea is not hot' expected: 'This tea is not hot'
 OK  got: "It's bad yet not" expected: "It's bad yet not"
```

Question 3

In [4]:
```
import math

# F. front_back
# Consider dividing a string into two halves.
# If the length is even, the front and back halves are the same length.
# If the length is odd, we'll say that the extra char goes in the front hal
# e.g. 'abcde', the front half is 'abc', the back half 'de'.
# Given 2 strings, a and b, return a string of the form
#  a-front + b-front + a-back + b-back
def front_back(a, b):
    #++ your code here ++

    frontHalf = math.ceil(len(a) / 2)
    backHalf = math.ceil(len(b) / 2)

    return a[0 : frontHalf] + b[0 : backHalf] + a[frontHalf : ] + b[backHa

#     if frontLen % 2 == 0:
#         if backLen % 2 == 0:
#             return a[0 : halfFront] + b[0 : halfBack] + a[halfFront : ] +
#         else:
#             return a[0 : halfFront] + b[0 : halfBack + 1] + a[halfFront :
#     else:
#         if backLen % 2 == 0:
#             return a[0 : halfFront + 1] + b[0 : halfBack] + a[halfFront +
#         else:
#             return a[0 : halfFront + 1] + b[0 : halfBack + 1] + a[halfFro

print()
print('front_back')
test(front_back('abcd', 'xy'), 'abxcdy')
test(front_back('abcde', 'xyz'), 'abcxydez')
test(front_back('Kitten', 'Donut'), 'KitDontenut')
```

```
front_back
 OK  got: 'abxcdy' expected: 'abxcdy'
 OK  got: 'abcxydez' expected: 'abcxydez'
 OK  got: 'KitDontenut' expected: 'KitDontenut'
```

Question 4

In [5]:

```
# Define a procedure weekend which takes a string as its input, and
# returns the boolean True if it's 'Saturday' or 'Sunday' and False otherwi
def weekend(day):
    return True if day == 'Saturday' or day == 'Sunday' else False

print(weekend('Monday'))
#>>> False

print(weekend('Saturday'))
#>>> True

print(weekend('July'))
#>>> False
```

```
False
True
False
```

Question 5

In [5]:

```
# By Ashwath from Udacity forums

# A leap year baby is a baby born on Feb 29, which occurs only on a leap ye

# Define a procedure is_leap_baby that takes 3 inputs: day, month and year
# and returns True if the date is a leap day (Feb 29 in a valid leap year)
# and False otherwise.

# A year that is a multiple of 4 is a leap year unless the year is
# divisible by 100 but not a multiple of 400 (so, 1900 is not a leap
# year but 2000 and 2004 are).

def is_leap_baby(day,month,year):
    # Write your code after this line.

    if month == 2 and day == 29:
        if year % 4 == 0 or year % 400 == 0:
            if year == 2000:
                return True

            if year % 100 == 0:
                return False

            return True
    else:
        return False

# The function 'output' prints one of two statements based on whether
# the is_leap_baby function returned True or False.

def output(status,name):
    if status:
        return "%s is one of an extremely rare species. He is a leap year b
    else:
        return "There's nothing special about %s's birthday. He is not a le

# Test Cases

print(test(output(is_leap_baby(29, 2, 1996), 'Calvin'),"Calvin is one of ar
print(test(output(is_leap_baby(19, 6, 1978), 'Garfield'),"There's nothing s
print(test(output(is_leap_baby(29, 2, 2000), 'Hobbes'),"Hobbes is one of ar
print(test(output(is_leap_baby(29, 2, 1900), 'Charlie Brown'),"There's noth
print(test(output(is_leap_baby(28, 2, 1976), 'Odie'),"There's nothing speci
# print(test(output(is_leap_baby(29, 2, 2004), "Justin"), "Justin is one of
 OK  got: 'Calvin is one of an extremely rare species. He is a leap year
baby!' expected: 'Calvin is one of an extremely rare species. He is a lea
p year baby!'
None
 OK  got: "There's nothing special about Garfield's birthday. He is not a
leap year baby!" expected: "There's nothing special about Garfield's birt
hday. He is not a leap year baby!"
None
 OK  got: 'Hobbes is one of an extremely rare species. He is a leap year
baby!' expected: 'Hobbes is one of an extremely rare species. He is a lea
p year baby!'
None
 OK  got: "There's nothing special about Charlie Brown's birthday. He is
not a leap year baby!" expected: "There's nothing special about Charlie B
rown's birthday. He is not a leap year baby!"
None
 OK  got: "There's nothing special about Odie's birthday. He is not a lea
p year baby!" expected: "There's nothing special about Odie's birthday. H
e is not a leap year baby!"
None
```