String-1

March 23, 2021

Q1) hello_name

Given a string name, e.g. “Bob”, return a greeting of the form “Hello Bob!”.

hello_name(‘Bob’) ‘Hello Bob!’ hello_name(‘Alice’) ‘Hello Alice!’ hello_name(‘X’) ‘Hello X!’

[2]:
```
def hello_name(name):
  greeting = "Hello {}!".format(name)
  return greeting
```

Q2) make_abba

Given two strings, a and b, return the result of putting them together in the order abba, e.g. “Hi” and “Bye” returns “HiByeByeHi”.

make_abba(‘Hi’, ‘Bye’) ‘HiByeByeHi’ make_abba(‘Yo’, ‘Alice’) ‘YoAliceAliceYo’ make_abba(‘What’, ‘Up’) ‘WhatUpUpWhat’

[3]:
```
def make_abba(a, b):
  combine_a_b = a + b + b + a
  return combine_a_b
```

Q3) make_tags

The web is built with HTML strings like “Yay” which draws Yay as italic text. In this example, the “i” tag makes and which surround the word “Yay”. Given tag and word strings, create the HTML string with tags around the word, e.g. “Yay”.

make_tags(‘i’, ‘Yay’) ‘Yay’ make_tags(‘i’, ‘Hello’) ‘Hello’ make_tags(‘cite’, ‘Yay’) ‘Yay’

[4]:
```
def make_tags(tag, word):
  astr = '<{}>{}</{}>'.format(tag,word,tag)
  return astr
```

Q4) make_out_word

Given an “out” string length 4, such as “«»”, and a word, return a new string where the word is in the middle of the out string, e.g. “<>”.

make_out_word(‘«»’, ‘Yay’) ‘<>’ make_out_word(‘«»’, ‘WooHoo’) ‘<>’ make_out_word(‘[[]]’, ‘word’) ‘[[word]]’

[5]:
```
def make_out_word(out, word):
  astr = out[:2] + word + out[2:]
  return astr
```

Q5) extra_end

Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.

extra_end('Hello') 'lololo' extra_end('ab') 'ababab' extra_end('Hi') 'HiHiHi'

[6]:
```
def extra_end(str):
  last_2char = str[-2:]
  return last_2char * 3
```

Q6) first_two Given a string, return the string made of its first two chars, so the String "Hello" yields "He". If the string is shorter than length 2, return whatever there is, so "X" yields "X", and the empty string "" yields the empty string "".

first_two('Hello') 'He' first_two('abcdefg') 'ab' first_two('ab') 'ab'

[7]:
```
def first_two(str):
  if len(str) < 2:
    return str
  else:
    return str[:2]
```

Q7) first_half Given a string of even length, return the first half. So the string "WooHoo" yields "Woo".

first_half('WooHoo') 'Woo' first_half('HelloThere') 'Hello' first_half('abcdef') 'abc'

[8]:
```
def first_half(str):
    first_half = str[0: len(str) / 2]
    return first_half
```

Q8) without_end

Given a string, return a version without the first and last char, so "Hello" yields "ell". The string length will be at least 2.

without_end('Hello') 'ell' without_end('java') 'av' without_end('coding') 'odin'

[9]:
```
def without_end(str):
  return str[1: len(str) - 1]
```

Q9) combo_string

Given 2 strings, a and b, return a string of the form short+long+short, with the shorter string on the outside and the longer string on the inside. The strings will not be the same length, but they may be empty (length 0).

combo_string('Hello', 'hi') 'hiHellohi' combo_string('hi', 'Hello') 'hiHellohi' combo_string('aaa', 'b') 'baaab'

[10]:
```
def combo_string(a, b):
  if len(a) > len(b):
    long = a
    short = b
  else:
    long = b
    short = a
  word = short + long + short
  return word
```

```
```

Q10) non_start Given 2 strings, return their concatenation, except omit the first char of each. The strings will be at least length 1.

non_start('Hello', 'There') 'ellohere' non_start('java', 'code') 'avaode' non_start('shotl', 'java') 'hotlava'

[13]:
```
def non_start(a, b):
  return a[1:] + b[1:]
```

Q11) left2 Given a string, return a "rotated left 2" version where the first 2 chars are moved to the end. The string length will be at least 2.

left2('Hello') 'lloHe' left2('java') 'vaja' left2('Hi') 'Hi'

[1]:
```
def left2(str):
  return   str[2:] + str[:2]
```

[ ]:
```
```