In [3]:
```
#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!'

def hello_name(name):


          return "Hello " + name + "!"
hello_name('Bob')

```

Out[3]: 'Hello Bob!'

In [4]:
```
#Given two strings, a and b, return the result of putting them together in t


#make_abba('Hi', 'Bye') → 'HiByeByeHi'
#make_abba('Yo', 'Alice') → 'YoAliceAliceYo'
#make_abba('What', 'Up') → 'WhatUpUpWhat'

def make_abba(a, b):
        return a+2*b+a
make_abba('Hi','Bye')

```

Out[4]: 'HiByeByeHi'

In [5]:
```
#The web is built with HTML strings like "<i>Yay</i>" which draws Yay as ita


#make_tags('i', 'Yay') → '<i>Yay</i>'
#make_tags('i', 'Hello') → '<i>Hello</i>'
#make_tags('cite', 'Yay') → '<cite>Yay</cite>'



def make_tags(tag, word):
       return "<"+tag+">"+word+"</"+tag+">"
make_tags('i','Yay')

```

Out[5]: 'Yay'

In [6]:
```
#Given an "out" string length 4, such as "<<>>", and a word, return a new st


#make_out_word('<<>>', 'Yay') → '<<Yay>>'
#make_out_word('<<>>', 'WooHoo') → '<<WooHoo>>'
#make_out_word('[[]]', 'word') → '[[word]]'

def make_out_word(out, word):
    return out[:2] + word + out[2:]
make_out_word('<<>>','Yay')

```

Out[6]: '<<Yay>>'

In [7]:
```
#Given a string, return a new string made of 3 copies of the last 2 chars of


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

def extra_end(str):
      return str[-2:]*3
extra_end('Hello')

```

Out[7]: 'lololo'

In [8]:
```
#Given a string, return the string made of its first two chars, so the Strin


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


def first_two(str):
      return str[:2]
first_two('Hello')

```

Out[8]: 'He'

In [27]:
```
#Given a string of even length, return the first half. So the string "WooHoo


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

def first_half(str):
   return str[:len(str)//2]
first_half('HelloThree')

```

Out[27]: 'Hello'

In [ ]:
```
#Given a string, return a version without the first and last char, so "Hello


#without_end('Hello') → 'ell'
#without_end('java') → 'av'
#without_end('coding') → 'odin'
```

In [21]:
```
#Given 2 strings, a and b, return a string of the form short+long+short, wit


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

def without_end(str):
      return str[1:-1]
without_end('Hello')



```

Out[21]: 'ell'

In [22]:
```
#Given 2 strings, return their concatenation, except omit the first char of


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


def combo_string(a, b):
  if len(a)<len(b):

    return a+b+a
  else:
    return b+a+b
combo_string('Hello','hi')

```

Out[22]: 'hiHellohi'

In [23]:
```
#Given a string, return a "rotated left 2" version where the first 2 chars a


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

def non_start(a, b):
  return a[1:]+b[1:]
non_start('Hello','There')
```

Out[23]: 'ellohere'

In [24]:
```
#Given a string, return a "rotated left 2" version where the first 2 chars a


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

def left2(str):
    return str[2:]+str[:2]
left2('Hello')

```

Out[24]: 'lloHe'

In [ ]:
```

```