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

In [ ]:
```
def hello_name(name):
    print(f"Hello {name}!")

hello_name("Bob")
```

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. "<>".

In [ ]:
```
def make_out_word(out, word):
    print(out[:2] + word + out[2:])

make_out_word("<<>>", "word")
```

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

In [ ]:
```
def first_teks(s):
    half_length = len(s) // 2
    print(s[:half_length])

first_teks("WooHoo")
```

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

In [ ]:
```
def first_concat(str1, str2):
    if len(str1) < 1 or len(str2) < 1:
        print("")
        return

    result = str1[1:] + str2[1:]
    print(result)

first_concat("Hello", "World")
```

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

In [ ]:
```python
def concatenation(a, b):
    result = a + b + b + a
    return result

result = concatenation("Hello" , "World")
print(result)
```

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.

In [ ]:
```python
def repeat_two_chars(s):
    repeat_two_chars = s[-2:]
    result = repeat_two_chars * 3
    return result

result = repeat_two_chars("Python")
print(result)
```

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

In [ ]:
```python
def first_and_last(s):
    if len(s) >= 2:
        return s[1:-1]

result = first_and_last("Hello")
print(result)
```

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.

In [ ]:
```python
def rotate_left(s):
    if len(s) >= 2:
        return s[2:] + s[:2]

result = rotate_left("Hello")
print(result)

result = rotate_left("World")
print(result)
```

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*".

In [ ]:
```
def make_html(tag, word):
    return f"<{tag}>{word}</{tag}>"

result = make_html("i", "Yay")
print(result)

result = make_html("b", "Bold")
print(result)

result = make_html("em", "Emphasized")
print(result)
```

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 "".

In [ ]:
```
def first_chars(s):
    if len(s) < 2:
        return s
    else:
        return s[:2]

print(first_chars("Hello"))
print(first_chars("X"))
print(first_chars(""))
```

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).

In [ ]:
```
def short_long_short(a, b):
    if len(a) < len(b):
        short = a
        long = b
    else:
        short = b
        long = a
    return short + long + short

print(short_long_short("Hello", "hi"))
print(short_long_short("hi", "Hello"))
print(short_long_short("aaa", "b"))
print(short_long_short("", "abc"))
print(short_long_short("abc", ""))
```

In [ ]: