Sequences II and Mutability

Warm-up

1. Rewrite the code inside `mystery` function so that it does not utilize a `break` statement.

```
def mystery(lst: list) -> list:
    i = 0
    while i < len(lst):
        if lst[i] != 'bye':
            i += 1
        else:
            break
    return lst[:i]
```

```
def mystery(lst: list) -> list:
    i = 0
    while i < len(lst) and lst[i] != 'bye':
        i += 1
    return lst[:i]
```

2. Write a function called `check_nums` that takes a list as its parameter, and contains a while loop that only stops once the element of the list is the number 7. What is returned is a list of all of the numbers up until it reaches 7.

```
def check_num(lst: list) -> list:
    i = 0
    while i < len(lst) and lst[i] != 7:
        i += 1
    return lst[:i]
```

3. Write a function called `escape_spaces` that takes in a phrase and returns the phrase with the space between words replaced by underscores.

```
def escape_spaces(phrase: str) -> str:
    return_str = ''
    i = 0
    for c in phrase:
        if c == ' ':
            return_str += '_'
        else:
            return_str += c
    return return_str
```

4. Evaluate the following Python expressions in the order that they are listed:

Python code	Output
`s = 'abc'`	
`lst = list(s)`	
`lst[0] = s`	
`print(lst)`	['abc','b','c']
`del lst[0]`	
`print(lst)`	['b','c']
`del s[0]`	Runtime error: str is immutable
`lst += ['a','b','c']`	
`print(lst)`	['b','c','a','b','c']
`del lst[1::2]`	
`print(lst)`	['b','a','c']
`lstc = ['b','a','c']`	
`lst == lstc`	True
`lst is lstc`	False
`id(lst) == id(lstc)`	False
`lsta = lst`	
`lst == lsta`	True
`lst is lsta`	True
`id(lst) == id(lsta)`	True
`lstb = lst[:]`	
`lst == lstb`	True
`lst is lstb`	False
`id(lst) == id(lstb)`	False
`lst.append('d')`	False
`lst += 'e'`	False
`print(lst)`	['b','a','c','d','e']
`lst += 1`	TypeError: int is not a sequence type
`lst == lsta`	True

Assume that the following statements have already been executed:

```
a = ['red']
b = 'mcgregory'
c = ['Alabama', 'Alaska', 'Arkansas']
d = a.copy()
```

5. Write **one line** of code that matches the comment provided. You should use list and string methods. You are expected to use list and string methods when appropriate.

a) Create a variable, called `e`, that is equal to `'McGregory'`. (You cannot simply write `e = 'McGregory'`)

e = b[:2].capitalize()+b[2:].capitalize()

b) Create a variable, called `f` that is an alias of `a`.

f = a

c) Create a variable, called `g` that is an clone of `a`.

g = a.copy() or g = a[:]

d) Remove the last element of the list `d` and print it.

print(g.pop())

e) Print a slice of `c` containing the last two items.

print(c[1:]) or print(c[-2:])

f) Use a and c to create a new list called h that looks like `['Alabama', 'Alaska', 'Arkansas', 'red']`

h = c+a

g) Add the string `'blue'` to the end of list a.

a.append('blue')

Trace the code below. For each line of code, write what it prints if it's a print statement or indicate the type of the relationship between variables by circling alias or copy for statements that are not print statements. It might be helpful to draw diagrams that keep track of the values that different variables point to.

```
def func(x, y):
    y[0] = x[0] + y[0]
    x = x[:]
    print(x)            # Point 1
    print(y)
    y = x

a=[1,2,3]
b=a.copy()
func(a,b)
print(a)                # Point 2
print(b)
```

6. At `Point 1` in the code, circle **all** the true statements. It is possible that no statements are true.

a) b is an alias of a

b) b is a copy of a

c) x is an alias of a

d) y is an alias of b

D is the only one that is true.

7. At `Point 2` in the code, circle **all** the true statements. It is possible that no statements are true.

a) b is an alias of a

b) b is a copy of a

c) x is an alias of a

d) y is an alias of b

None are correct

8. Write down what the program prints:

```
[1,2,3]
[2,2,3]
[1,2,3]
[2,2,3]
```