1 While Loops

1.1 Motivation

Consider the following scenario:

```python
password = input("Please enter your password: ")
check = input("Please re-enter your password to verify: ")
```

What if the user makes a typo? We would like to allow the user to re-enter passwords until they get it right. This means we may need to repeat the line:

```python
check = input("Please re-enter your password to verify: ")
```

an unknown number of times.

Q. What statement have we already used to make a block of code happen several times?

A.

But how would you write a for-loop for this scenario?

1.2 Definition and Details

A while loop allows us to continue looping as long as some condition is True.

Form for a while loop:

```python
while <condition>:
    <while-body>
```

- while is the Python keyword
- condition is a boolean that must be True to enter/repeat the while loop.
- while-body is any set of Python statements (note they are indented).
  - Note that something should happen within the body of your while loop that will make the condition eventually evaluate to False.

How it works:

1. Check to see if the condition is True.

2. If the condition is False, we are done with the loop.

3. If it is True:
   - execute the entire body of the loop (even if the condition becomes False at some point).
   - go back to step 1.
Let’s write a **while** loop that keeps asking the user to verify their password until they get it right:

Q. What is the minimum number of times this loop executes?
   A.

Q. What is the maximum number of times?
   A.

Q. How are **while** loops different from **for** loops?
   A.

What if a loop runs forever? Let’s see if there’s a different way to implement the above **while**-loop.
1.3 Examples

Write the following functions using `while` loops.

```python
def count_letter(s, letter):
    """(str, str) -> int

    Precondition: len(letter) == 1

    Return the number of times the character letter occurs in s.
    (Note: Do NOT use any string methods.)
    
    >>> count_letter('banana', 'a')
    3
    """

def find_letter_n_times(s, letter, n):
    """ (str, str, int) -> str

    Precondition: len(letter) == 1

    Return the smallest substring of s starting from index 0 that contains
    n occurrences of letter. Return the empty string if no such substring exists.
    
    >>> find_letter_n_times('Computer Science', 'e', 2)
    'Computer Scie'
    
    >>> find_letter_n_times('Computer Science', 'p', 10)
    ''
    """
```