PSTAT 5A: Lab06 Solutions (generated by Ethan)

Task 1

In [1]:
```
x = ['success', 'failure', 'failure', 'success', 'failure', 'failure', 'fail
```

In [2]:
```
for k in x:
    print(k == 'success')
```

```
True
False
False
True
False
False
False
True
```

Remember that the variable name (which I called `k` above) is not unique.

Task 2

Here is what the table looks like:

FIRST ITERATION	
Start of Iteration	• `k` : `'success'`
End of Iteration	• `k` : `'success'`
SECOND ITERATION	
Start of Iteration	• `k` : `'failure'`
End of Iteration	• `k` : `'failure'`
THIRD ITERATION	
Start of Iteration	• `k` : `'failure'`
End of Iteration	• `k` : `'failure'`
FOURTH ITERATION	
Start of Iteration	• `k` : `'success'`

End of Iteration	• `k : 'success'`
FIFTH ITERATION	
Start of Iteration	• `k : 'failure'`
End of Iteration	• `k : 'failure'`
SIXTH ITERATION	
Start of Iteration	• `k : 'failure'`
End of Iteration	• `k : 'failure'`
SEVENTH ITERATION	
Start of Iteration	• `k : 'failure'`
End of Iteration	• `k : 'failure'`
EIGHTH ITERATION	
Start of Iteration	• `k : 'success'`
End of Iteration	• `k : 'success'`

Task 3

In [3]:
```
count = 0

for k in x:
    if k == 'success':
        count += 1

count
```

Out[3]: 3

Task 4

In [4]:
```
import numpy as np
```

In [5]:
```
count = 0

for k in np.arange(0, len(x)):
    if x[k] == 'success':
        count += 1

count
```

Out[5]:
```
3
```

Task 5

Using `arange()`: We need to specify the *step size*; i.e. the space between the numbers, which in this case is 0.1.

In [6]:
```
np.arange(1, 2.1, 0.1)
```

Out[6]:
```
array([1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2. ])
```

Using `linspace()`: We need to specify the *total number of elements*, which in this case is 11.

In [7]:
```
np.linspace(1, 2, 11)
```

Out[7]:
```
array([1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2. ])
```

Task 6

Rolling a fair six-sided die 100 times and recording the outcome on each roll is equivalent to picking 100 numbers (at random and with replacement) from the set `[1, 2, 3, 4, 5, 6]`. As such, we run

In [8]:
```
import random as rnd
x = rnd.choices([1, 2, 3, 4, 5, 6], k = 100)
```

We can examine the first 10 elements of `x` using indexing:

In [9]:
```
x[0:9]
```

Out[9]:
```
[6, 2, 6, 2, 6, 1, 2, 4, 2]
```

By the Way: If we wanted to be fancy, we could have generated the list `[1, 2, 3, 4, 5, 6]` using `arange()` or `linspace()`. Because the number of sides on the die for this problem (6) is relatively small this may be overkill, but we can imagine that for a

multifaceted die (e.g. a 12-sided die, or a 36-sided die) using `arange()` or `linspace()` might be a good idea.

In [10]:
```
x = rnd.choices(np.arange(1, 7), k = 100)
```

In [11]:
```
x[0:9]
```

Out[11]:
```
[5, 3, 6, 3, 3, 4, 4, 3, 3]
```