Coding Class

Definitions

What is a definition?

Definitions are mainly used to reduce the copying of code

Like this:

```
def basicFunction(paramater1, paramater2):
  if bool(paramater1) == True:
    print(paramater2 + " is true.")
  if bool(paramater1) == False:
    print(paramater2 + " is false.")
```

Definitions are mainly used to reduce the copying of code

Its easier to do this:

```
def add(num1, num2):
  return num1 + num2

x = add(3, 4)
x = add(x, 1)
x = add(add(6432, x), x)
print(x)
```

Than this:

```
x = 3 + 4
x = x + 1
x = (6432 + x) + x
print(x)
```

This may look easier, but if you wanted to change all of the “+” to “-“ it would be hard.

```
x = 3 + 4
x = x + 1
x = (6432 + x) + x
print(x)
```

But for this all you would really have to change is the `return num1 + num2` -> `return num1 - num2`

```
def add(num1, num2):
  return num1 + num2

x = add(3, 4)
x = add(x, 1)
x = add(add(6432, x), x)
print(x)
```

Text Manipulation

Definitions are often used to manipulate text.
Let's say that you want to remove all the spaces in a string.

```
def removeSpaces(string):
  string = string.replace(" ", "")
  return string
```

This would be the code that would Make this work.

The `replace` function takes what every is in the first parameter Which is the “ “ and replaces it with the second parameter Which is “”

So if we passed this: `print(removeSpaces("Hello, how are you?"))`

It would return: `Hello,howareyou?`

Text Manipulation

If we changed the code up a bit so that it would remove words that we don't want.

```
def removeSpaces(string):
  string = string.replace("Hello", "Hey")
  return string
```

This would be the updated code

And if we ran the same code as before:

```
print(removeSpaces("Hello, how are you?"))
```

It would return: `Hey, how are you?`

This is useful because your program can find words and change them to others.

Text Manipulation

Loops are also helpful with definitions.

For example this code will double numbers, starting with the `startNum` value

```
def loopAmount(loopTimes, startNum):
  count = 1
  while count < loopTimes:
    print(startNum)
    startNum = startNum + startNum
    count = count + 1
  return startNum
```

Try to guess the output when I run this:

```
print(loopAmount(10, 1))
```

Answer:

```
1
2
4
8
16
32
64
128
256
512
```

Try to guess for this one:

```
print(loopAmount(8, 20))
```

Answer:

```
20
40
80
160
320
640
1280
2560
```

We can use definitions like this to minimize and speed up our math in programming.