Class 2 - Functions and Conditionals
Recap of Class 1

- **Print statements**
  - Printing text eg. `print("Hello World")` => ‘Hello World’ in the shell
  - Printing the value of variables eg. `x = "Hello"; print(x)` => ‘Hello’ in the shell
  - Doing simple math eg. `print(2 + 3 * 5)` => ‘17’ in the shell
- **Taking input from the user**
  - Eg. `name = input()` => allows the user to type in something that gets stored in the variable ‘name’ after the user hits enter.
- **Basic data types**
  - Integer, float, string and boolean.
  - `type()` function - allows the user to know the type of an expression.
Some math operations

1. \( a + b \)  # Summation
2. \( a - b \)  # Subtraction
3. \( a \times b \)  # Multiplication
4. \( a / b \)  # Float division
5. \( a \div b \)  # Integer division (rounds down)
6. \( a \mod b \)  # Modulus/Remainder operator
7. \( a^{**}b \)  # Power operator
Comparison Operators

- `==` (Equality)
  - Note: Equality is `==` because `=` is used for assigning to variables
- `<` (Less than)
- `<=` (Less than or equal to)
- `>` (Greater than)
- `>=` (Greater than or equal to)
- `!=` (Inequality)

These operators compare two things and evaluate to a boolean ie. either True or False.
Learning goals:

1. Comparison operators
2. Working with Booleans

```python
main.py

1. `print(3 == 3)`
2. `print(3 == 5)`
3. `a = 18; print(a)`
4. `print(a < 21)`
5. `print("Hello" == "Hello")`
6. `print("Hello" == "Hello")`
7. `print(2+5*3 >= 20)`
8. `print(7 != 42)`
9. `print(True == False)`

<table>
<thead>
<tr>
<th>Shell</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>18</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>
```
Pop Quiz 1:

What is the output of the following program:

1. `print("True" == True)`
2. `print("3+3" == 6)`
3. `print(int(False) == 0)`
Let $X$ and $Y$ be boolean expressions.

- $X \text{ and } Y$ evaluates to True if $X$ evaluates to True and $Y$ evaluates to True. It evaluates to False otherwise.
- $X \text{ or } Y$ evaluates to True if either $X$ evaluates to True or $Y$ evaluates to True. It evaluates to False otherwise.
- $\text{not } X$ negates the value of $X$. 

Boolean Operators (and, or, not)
Demo

Learning goals:

1. Boolean operators
2. Using comparison and boolean operators together
Pop Quiz 2:

Which of the following would result in an output of "True" for this program (possible multiple answers):

```python
print((a < b) or (b > c and a > c) or a == d)
```
Control Flow

Sequential control flow [previous lecture]

- Statement 1
- Statement 2
- Statement 3
- Statement 4

Conditional control flow [this lecture]

- Statement 1
- If 1
  - True -> Statement 2
  - False -> Statement 3
If-Statements

If statements are responsible for allowing certain lines of code to be run only when a certain condition is met.

For example, code lines 1 and 2 will only be run when ‘condition’ evaluates to True.

Note: The indentation (1 tab / 4 spaces) of the code lines under the if statement is important!

Indented lines denote lines that are guarded by the if statement.

Syntax:

if (condition):
    Code line 1
    Code line 2
    ...

If-Else Statements

if (condition):
    Code line 1
else:
    Code line 2

If-else statements can interpreted as follows:

If condition is true, then run code line 1.
Otherwise, run code line 2.

Again, indentation here is important!
```python
if (10 > 3):
    print("10 is bigger than 3")
if (8 < 2):
    print("8 is smaller than 2")
print("We love math")
```

Output:
```
10 is bigger than 3
We love math
```
```python
if (5 > 4):
    print("5 is bigger than 4")
else:
    print("5 is smaller than 4")
print("We love math")

if (8 == 2):
    print("8 is 100% equal to 2")
    print("Anything is now possible")
else:
    print("Math is working as expected")
```
If-Elif-Else Statements

if (condition 1):
    Code line 1
elif (condition 2):
    Code line 2
else:
    Code line 3

If-elif-else statements can interpreted as follows:

If condition 1 is true, then run code line 1. Otherwise, if condition 2 is true, then run code line 2. Otherwise, run code line 3

Again, indentation here is important!

Note: You can have as many elifs as you want!
```python
if (4 > 4):
    print("4 must be bigger than 4")
    print("My genius frightens me sometimes")
elif (4 < 4):
    print("4 must be smaller than 4")
    print("IMO Gold medal here I come")
else:
    print("4 is equal to 4")
    print("Makes sense I guess")
print("We love math")
```

4 is equal to 4
Makes sense I guess
We love math
Pop Quiz 3:

Which of the following computes whether a variable $a$, which is always the integer 0, 1, 2 or 3, is even or odd:

**Code 1**

```python
if not(a == 1 or a == 3):
    print("odd")
else:
    print("even")
```

**Code 2**

```python
if a == 0 or a == 2:
    print("even")
else:
    print("odd")
```

**Code 3**

```python
if a != 1 or a != 3:
    print("even")
else:
    print("odd")
```

**Code 4**

```python
print("even" if a == 0 or a == 2 else "odd")
```
Grouping code into functions

```python
1. def simpleFunction(x):
2.     print(x)
3.     print(x)
4.  simpleFunction("Hello!")
```
Anatomy of a function

def func(parameters):

body

return X

parameters are variables that will be provided when the function is called

contains the actions (statements) that the function performs
returns a value (optional)
Simple example of a function

We define a function called double that takes one parameter and returns its double.

```python
def double(x):
    # This function doubles a number
    print("Doubling a number!")
    return 2 * x

print(double(4))
```

To use the function, we use this syntax, similar to how a function in math is used.
```python
def f(x, y, z):
    return x + y + z

print(f(1, 3, 2))  # Prints 6
print(f(1, 2))     # WRONG! Needs 2 parameters

def g():
    return 42

print(g())         # Prints 42

def h(x, y):
    if x > y:
        print("x is greater than y")
    elif y > x:
        print("y is greater than x")
    else:
        print("x and y are the same!")

h(4, 5)
h(5, 4)
h(5, 5)
```
Variable scope

def f(x):
    y = 5
    return x + y

print(f(4))  # Prints 9
print(x)     # Crashes!
print(y)     # Crashes!

Variables defined in the body of a function definition are only defined inside the indented block!

In the code on the left, the two last print statements will crash because we never defined a variable x or y in that scope.
Pop Quiz 4:

What is the output of the following program:

```python
def max(a, b):
    if a > b:
        return b
    b = 2
    return a

a = 5
b = 10
print(2 + max(2, 3))
print(a)
```
Another example

```python
def f(x):
    print("In f, x =", x)
    x += 7
    return x - 1

def h(x):
    x += 3
    return f(x+4)

x = 5
print(x)
print(f(x))
print(h(x))
print(x)
```

What’s the output of the code on the left?

Visualize here: [https://shorturl.at/etNS2](https://shorturl.at/etNS2)
What's next?

Homework will be posted on Piazza by tomorrow!

Class 3: Loops

How to write code that repeats instructions
How to iterate through the input