Lesson 3: Loops and Lists

Recap of Last Lesson:

- *Why do we need if statements/conditional logic?*
    - Almost all software/code has to do certain things only sometimes - i.e. a video game only increases the score when something happens. We need a way to run some code when some conditions are met.
- *Conditions:*
    - These are the expressions we write which can be evaluated to True or False. We use operators like ==, <=, >=, !=, is, etc. in these expressions (i.e. cash >= cost).
- *If Statements:*
    - An if statement is something with a condition and some code. If the condition is True, the code inside the statement will get executed. If the condition is False, the computer will skip all the code in the statement.
- *Else-If Statements:*
    - Else-if statements are like if statements, but they only get checked if the if statements (or other elif statements) above them are False.
- *Else Statements:*
    - Else statements can be put at the end of a block of if/elif statements, and it contains code that should be executed if all the if/elif statements above are False.
- *Nested If Statements:*
    - We are allowed to write if statements inside of other if statements. We tell Python what blocks are inside of other blocks by using tabs.

Homework from Last Lesson:

If the user enters 1, the program runs "Apple Picking" mode, **if they instead enter a 2,** the program runs a "Tax Calculator" mode**, if they enter a 3,** it runs a "Rectangle Area Calculator" mode.

In Apple Picking mode, you ask the user for a number of apples, and print how much money they can make from the apples (assume they could sell 1 apple for $1.50). If they can make more than $50, tell them "You are rich!", if they can make more than $25 but less than $50, tell them "You can make a decent amount", otherwise don't print anything besides how much money they can make.

In Tax Calculator mode, you ask the user for the price of a product, and print how much tax they will pay, and what the grand total cost would be (assume a 13% tax rate).

In Rectangle Area Calculator mode, you run the program you wrote last week - ask the user for length and width, and give them the area and perimeter.

Motivation for Loops

In code, **we want to be able to repeat things**. Some examples are:

- Video games - until the game is over, updating the screen view, keeping score, handling different game events needs to happen.
- Social media - the page/app loads when we launch it, but if there's new content, the page/app needs to repeat that process and load the new content on the screen.
- Algorithms - algorithms like path finding (i.e. giving directions on Google Maps) need to repeat certain steps to calculate the solutions

Another simple example is, **what if I asked you to write a program that prints something 10 times on the screen?**

This solution works, but there are some issues. It's not easy to change. If I asked you to change the program so it prints it 1,000 times, you'll spend a lot of time copy and pasting. If I told you to make it print 1,000,000 times, the python file will now take up GB's of space (which is bad).

```
print("Hello World!")
print("Hello World!")
print("Hello World!")
print("Hello World!")
print("Hello World!")
print("Hello World!")
print("Hello World!")
print("Hello World!")
print("Hello World!")
print("Hello World!")
```

While Loops

In Python there are two types of loops, the first one is called **while loops**. Here's an example:

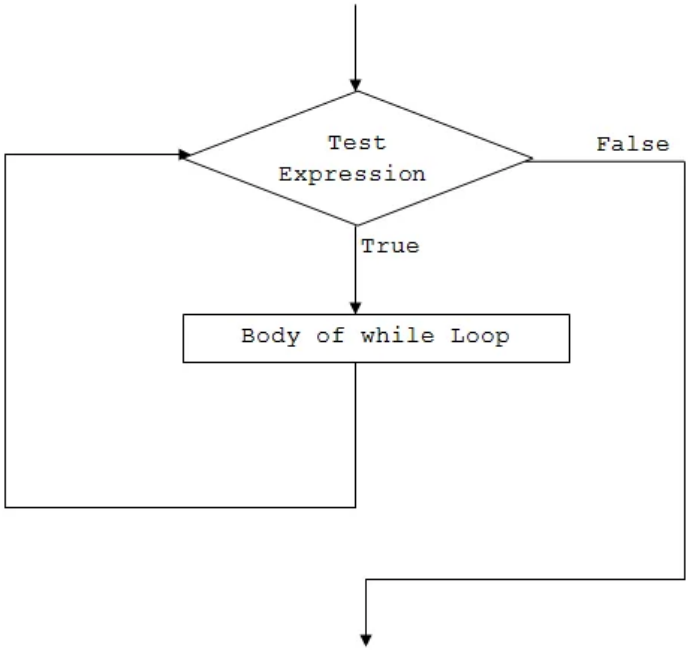
```
number = 1

while number <= 10:

    print("Hello World!")

    number += 1
```

This looks a lot like an if statement, except we used the word 'while' instead of 'if'. Just like an if-statement, **the while loop has a condition**. The computer will **test the condition** and if it is True, run the code inside of the loop. When it hits the end, it will **jump to the top and retest the condition.**



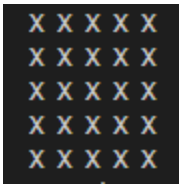
Side note: the `**number += 1**` is just a short-form of saying `**number = number + 1**`.

Questions:

- What will happen when this code is run (what will be put on the screen)?
  - Prints hello world 10 times
- What if I wanted to change this code so that it prints 100 times?
  - Prints it 100 times
- What would happen if I change the first line to `**number = 1000**`?
  - Nothing is printed
- What would happen if I change the condition to `**number >= 0**`?
  - **The loop never stops. This is called an INFINITE LOOP**.

Activity:

- Write a program that prints a square of X's on the screen:
  - i.e. you have a variable called length, and if we set length to 5, the screen should show:

- 
- If we set length to 20:
- 
- **Hints:** If we write **print("X " * length)**, X will be printed length times in 1 line.

Nested Loops

Just like if statements, we can put loops inside of loops. We can even put if statements inside of loops, inside of other loops.

```
row, length = 1, 5

while row <= length:

    row += 1

    column = 1

    while column <= length:

        column += 1

        print("X", end=" ")

    print("")
```

Questions:

- How many times does the outer loop run?
    - 5 times.
- For each time the outer loop runs, how many times will the inner loop run?
    - 5 times.
- How many times in total will the inner loop run?
    - 5 x 5 = 25
- What does the whole code do?
    - It prints a 5x5 square
    - The **print("X", end=" ")** prints 1 X, without moving to a new line.
    - The **print("")** moves the cursor to a new line.

Lists

Question: So far what types of variables do we know?

- Integers - whole numbers → x = 0
- Floats - decimal point numbers → x = 0.5
- Strings - letters → x = "blah"

We need to introduce another type of variable in Python called **lists**. This is a **special type of variable** that contains more than 1 piece of information. Just like a to-do list or a grocery list, a list is something that contains 0 or more things.

```
# Create a list

my_first_list = [1, 2, 3]

# Access the elements with [] operator

print(my_first_list[0])

# Change the contents

my_first_list[1] = 5

# Add new elements

my_first_list.append(4)
```

NOTE: Lists are **"zero indexed"** which means in code **the 1st thing in the list is at index 0**.

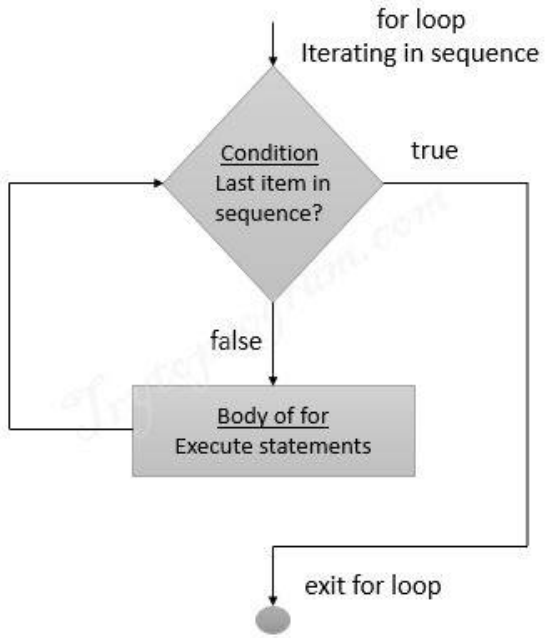
For Loops

We can now introduce the other type of loop in Python. For loops are loops that run for each element in a list. In Python we use special words **for** and **in** to define the for loop.

```
grocery_list = [
    "Apples",
    "Bananas",
    "Chicken",
    "Bread",
    "Ice Cream"
]


for item in grocery_list:
    print(item)
```

As this loop runs, **item will get set to the values in grocery_list**. The code will print the list.



Related Information (may need for homework)

If we have a string in Python, we can **split the string and turn it into a list.**

```
sentence = "Hello, my name is Josh."

words = sentence.split(" ")
```

To test if a number is even, we can use the **modulus operator (%)** which returns the **remainder after division**.

```
x = 1 % 2 # 1

x = 2 % 2 # 0

x = 3 % 2 # 1

x = 4 % 2 # 0

x = 5 % 2 # 1

x = 6 % 2 # 0
```

Activities/Homework:

1. Write a program that prints a shape like this if **size = 9**. If **size = 5** it would only have 5 rows. Hint: Think about how we can calculate how many stars to print based on what row we are on… i.e. when we're at row 2, how do we know to print 2 stars? What about row 7? Does your formula change when size changes?

```
*

**

***

****

*****

****

***

**

*
```

Challenge: Try to see if you can do it using only 1 loop.

2. Even or Odd:

Write a program that asks the user to enter an integer. The program counts up the numbers between 1 and the number (prints them on the screen). If a number is even, it will also print "which is even" and if its odd, will print "which is odd"

3. Calculator:

Write a program that asks the user to enter a bunch of numbers (separated by spaces). The program calculates the **sum** of the number and the **product** of the number, and prints those. Try to do this one way using while loops, and another using for loops.

Input: "5 6"

4. ** Challenge ** Calculator:

Write a program that asks the user to enter an expression to calculate where the numbers/operators are separated by spaces (i.e. "5 + 5" or "6 + 6"). The program calculates the answer and prints it. **NOTE:** You don't need to worry about the order of operations, or the user entering something invalid (i.e. "5 +") since that makes the program a lot harder.

Input: "5 + 6 + 10 + 2 + 3"

If you really want a challenge: the expressions could contain addition (+) subtraction (-) multiplication (x) or division (/).

Input: "5 + 6 x 10 / 2 - 3"