

Welcome back to Coding!

Lesson 2: Arrays, Loops, and Memories

Brought to you by the University of Maryland Balloon
Payload Program

Recap

- Data Types
- Variables
- Conditional Statements
- Libraries

Data Types

- Integer: 1, 2, 3, 4, 5, ...
- Float: 1.5, 20.2, 100.12354, 3.1415, ...
- Character: 'a', 'b', 'c', ...
- String: 'Hello World', 'abcdefg', ...
- Boolean: True, False

Variables

Ways for us to store information that we want to use later

```
X = "pony"
```

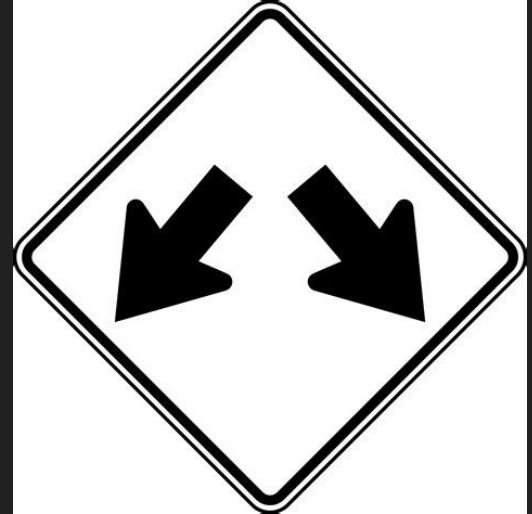
```
X = 10
```

```
X = X + 10
```

Conditional Statements

A way for us to get the computer to make decisions

- If something is true
 - Run this code
- Else
 - Run this code



Libraries

Snippets of code that other people wrote that we use ourselves

- Simplifies what we need to do

```
random.randint(low, high)
```

Picks a number between and including the low and high bounds

Arrays - what are they?

- A way of representing a list of things in code
- Example shopping list:
 - Eggs
 - Milk
 - Bread
 - Apples
- Example shopping list *array*

```
shopping_list = ["eggs", "milk", "bread", "apples"]
```

So an array is just a list of things -> this is why python calls them *lists* (list = human, array = computer)



Arrays - how to write them down

- Commonly used syntax:

Starting bracket —————→ [

Data separated by commas —————→ 1, 2, 3, 4, 5

Ending bracket —————→]

Combining it all together:

[1, 2, 3, 4, 5] —————→ an array!

Sometimes, arrays use parenthesis —————→ (thing1, thing2, thing3, ...)

Sometimes, array use curly brackets —————→ {thing1, thing2, thing3, ...}

Most often, they use brackets!!! —————→ [thing1, thing2, thing3]

Arrays - uses n'stuff

Arrays allow us to store multiple points of data in a specific order - just like a list of things!

Python arrays (called *lists*) can store different types of data:

- **Strings:** ["a", "b", "c", "defg", "h"]
- **Floats/Doubles:** [124.832, 23.11, 2.431]
- **Integer:** [1, 54, 46, 9, 0]
- **Booleans:** True/False

In fact, a string is just an *array* of *character*!

```
[ "i", " ", "a", "m", " ", "r", "o", "m", "e", "o" ] = "i am romeo"
```

Using and saving data in a array

When you want to change a value in your list, you cross it out or erase it - same with code!

Let's say I have an array:

```
my_things = ["ball", "shoe", "pants", "hat"]
```

Get the shoe:

```
my_shoe = my_things[1]
```

Add an item to the end of the array:

```
my_things.append("phone")
```

 here, I've added a new item called "phone"

Change the item at a specific place in the array:

```
my_things[2] = "jeans"
```

 here, I've changed "pants" to "jeans"

Using data stored in a array

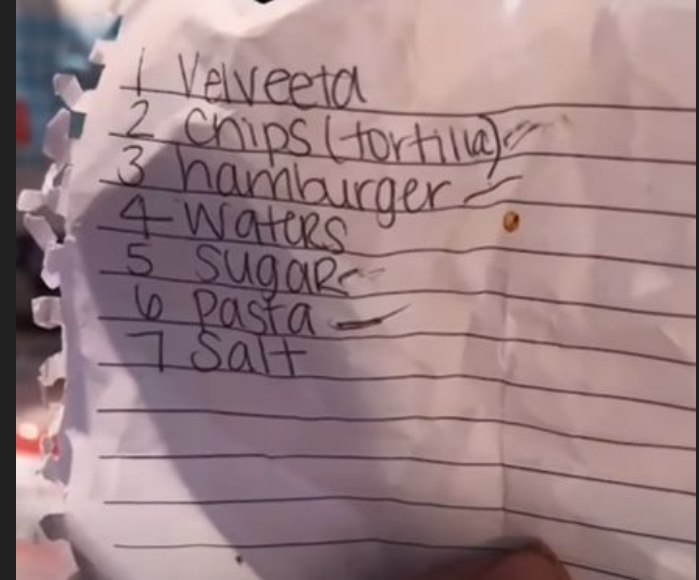
Normal paper lists, when numbered, start at 1, but arrays start at 0, and represent the index (location in the list) of a value:

	0	1	2	3
	↓	↓	↓	↓
my_list =	["a", "b", "c", "d"]			
print(my_list[0])	→ "a"			
print(my_list[2])	→ "c"			

Try it!

Normal shopping list

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.



2d arrays

You can also store arrays inside of arrays, and make a *matrix* of data -> pretty much like a table of stuff instead of a list of stuff:

Example:

My table for the weather:

1)	Time of Day		Temperature	Row 1
2)	12pm		60°	Row 2
3)	1pm		65°	.
4)	2pm		68°	.
5)	3pm		70°	.
6)	4pm		71°	
	Column 1		Column 2	

2d arrays

You can also store arrays inside of arrays, and make a *matrix* of data -> pretty much like a table of stuff instead of a list of stuff:

Example, my 2d array representing the weather:

```
my_weather_table = [ [12, 60],  
                      [ 1, 65],  
                      [ 2, 68],  
                      [ 3, 70],  
                      [ 4, 71] ]
```

—————> Basically just arrays inside of arrays!

To access the data, you need to index twice! First number is the **ROW**, second number is the **COLUMN**:

```
print(my_weather_table[2][1]) —————> 68
```

Using Arrays to make Rock, Paper, Scissors Better

```
choice = 1
```

```
rps = ["rock", "paper", "scissors"]
```

```
selected_choice = rps[choice - 1] # Why subtracting 1?
```

```
print("selected_choice")
```

Numpy

Numpy is a python package that allows you to create 1D and 2d arrays with much more ease and use them:

```
Import numpy
```

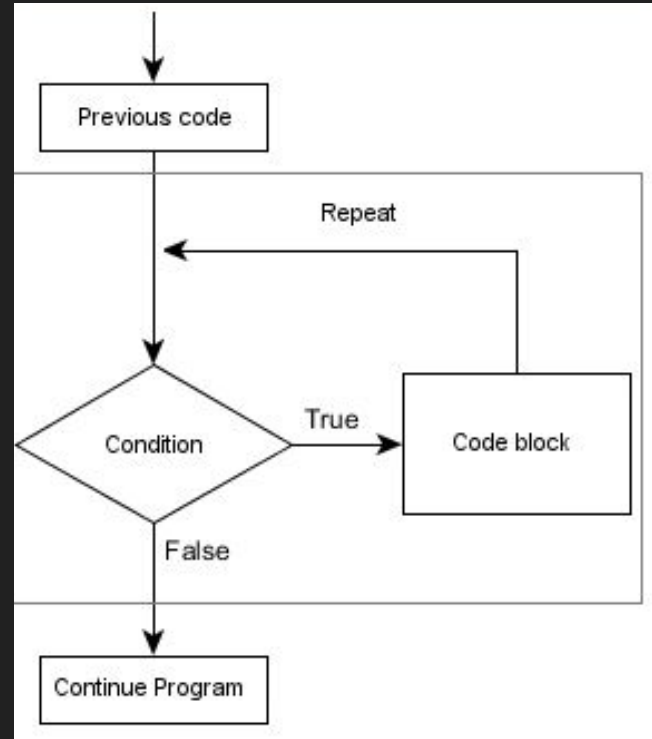
Or, people commonly do:

```
Import numpy as np
```

This allows you to type **np** instead of **numpy** when using the language.

Loops

- **Loops:** sets of instructions that are continuously repeated *until something happens*
 - Types: 'for' loops, 'while' loops
- Incredibly useful for making the computer do things hundreds, millions, or even billions of times
 - Yeah, the computer is that fast!



The “while” Loop

Syntax:

```
while (condition):
```

```
    (code you want to run)
```

```
    (you can add as much as you want)
```

This will run the code you want to run until the condition becomes FALSE.

The “while” Loop - Example

```
x = 0
```

```
while (x < 10):
```

```
    print(x)
```

```
x = 0
```

```
while (x < 10):
```

```
    print(x)
```

```
    x = x + 1
```

The “while” Loop - Example

```
word = “hello”  
  
while (len(word) < 10)  
    word = word + “o”  
  
print(word)
```

```
word = “hello”  
  
while ( word != “helloooo” )  
    word = word + “o”  
  
print(word)
```

DAN ADD A LEETCODE CS50 TYPE LOOP EXAMPLE
TO SHOW HOW YOU CAN GET COOL STRUCTURE
WITH NESTED LOOPS OR SOMETHING

The “for” Loop: Arrays & Loops

Syntax:

```
for ( i in range(#, #) ):
```

```
    (code you want to run)
```

Syntax:

```
for ( element in list ):
```

```
    (code you want to run)
```

```
if (condition):
```

```
    break (this stops the loop)
```

The “for” Loop - Example

Syntax:

```
list = ["rock", "paper", "scissors"]
```

```
for ( i in range(0, 2) ):
```

```
    print( list[i] )
```

```
print(shoot)
```

Syntax:

```
list = ["rock", "paper", "scissors"]
```

```
for ( element in list ):
```

```
    print(element)
```

```
print(shoot)
```

Breaking Loops

Let's say you have a loop that's working great - but you want it to stop if something specific happens.

For example: *you want to find out if a word is in the dictionary*

- You are **looking through every word in the dictionary**
- You look at each word, and **check if it's the word you want**
- When you **find the word you want, you stop the loop**

Breaking Loops

```
for ( element in list ):
```

```
    (code you want to run)
```

```
    if (condition):
```

```
        break (this stops the loop)
```

Let's try this with an index variable!

```
word_i_want = "orange"
```

```
dictionary = ["apple", "banana", "coconut", ...]  
(imagine every word is in this list)
```

```
for ( word in dictionary ):
```

```
    if (word_i_want == word):
```

```
        print("Your word is in the dictionary")
```

```
        break
```

```
print("code complete")
```

Some useful tools:

```
word = "HELLO"
```

```
word = word.lower()
```

```
print(word)
```

This turns your words to lower-case, which might be useful in case you accidentally use caps!

```
i = 1
```

```
i = i + 1
```

```
print(i)
```

```
i += 1
```

```
print(i)
```

These two statements are equivalent

Computer Memory

Where is all of this stuff being stored?

- Our code, variables, etc.

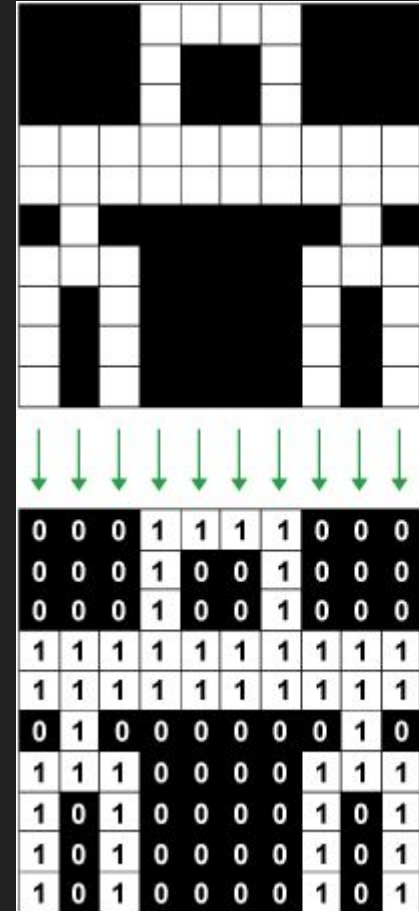
The computer contains a grid of ones and zeros.

- Everything we do happens in these ones and zeros

Memory Addresses

How does the computer know where things are?

- Each piece of memory has an address
- This address is stored in our variable
- Data is sequential
 - The data at address 10 is after 9 and before 11



How is memory useful?

Arrays, long lines of sequential memory

-

Project Setup

VSCode sorts projects into folders

- Let's make a folder to store our code

We're going to use libraries...

- Libraries are other people's code
- Their code is updated, which might make our code stop working

```
python -m venv myenv
```

Today's Project: The 5x5 Treasure Hunt

Rules:

1. There is a **5x5 grid**
2. The treasure is hidden at a **random place**
 - a. Place = Row/Column
3. We **guess where the treasure is**
 - a. If we are right, we win!
 - b. If we are wrong, we put an X there
4. (Optional) If the treasure chest is right next to our guess -> put a "?" there

	0	1	2	3	4
0					
1					
2					
3					
4					

Upcoming Topics

Lesson 1: Coding Basics & Logic

Lesson 2: Arrays, Loops, & Logic

Lesson 3: Graphics, UI, Functions

Lesson 4: Python Local HTML Website, Objects

Access this lesson and extra materials online!

Visit our Wiki Page:

<https://ter.ps/STEMworkshop>

Coding on our computers

Visual Studio Code

- Where we type all our code
- code.visualstudio.com



Python

- Lets our computer run our written code
- python.org/downloads



What is Visual Studio Code (VSCode)?

A nicer way to write and edit code

- Technically, all code could be written in notepad
- VSCode has tools that make coding easier
 - Autocompletes variable names
 - Colors written code
 - User extensions for additional features
 - Highlights errors
 - More!

```
# Water (includes thermal scattering)  
water = openmc.Material(name='Water')
```

Functions

Define a function for addition

```
def add(a, b):  
    return a + b
```

Define a function for subtraction

```
def subtract(a, b):  
    return a - b
```

Define a function for multiplication

```
def multiply(a, b):  
    return a * b
```

Define a function for division

```
def divide(a, b):  
    if b == 0:  
        return "Error! Cannot divide by zero."  
    return a / b
```

Using the functions

```
x = 10  
y = 5
```

Call the functions and print the results

```
sum_result = add(x, y)  
print("Addition:", sum_result)
```

```
diff_result = subtract(x, y)  
print("Subtraction:", diff_result)
```

```
product_result = multiply(x, y)  
print("Multiplication:", product_result)
```

```
quotient_result = divide(x, y)  
print("Division:", quotient_result)
```