Python Bootcamp

Workshop 2

8/4

Youth In Code

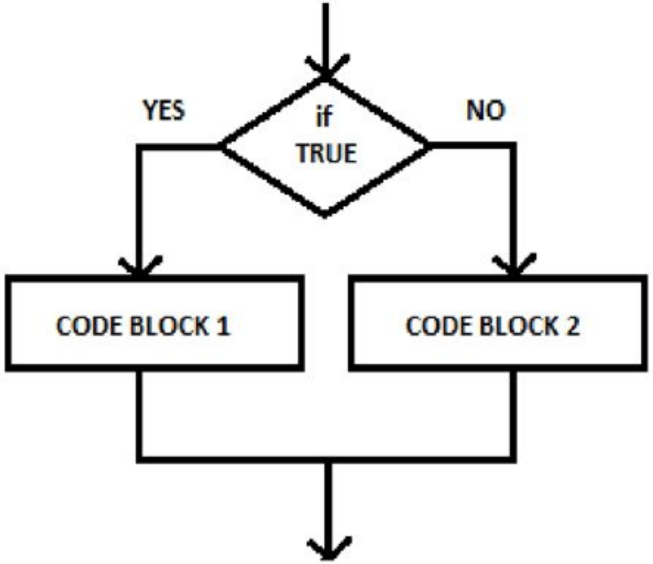
Justin Liu

Roadmap

- Flow Control
- Boolean Operators
- Indentation
- Conditionals - if, elif, and else
- *break* and *continue* Statements
- While loops, For loops, and range()
- The Random module
- sys.exit()
- Build a Rock, Paper, Scissors Game!

Flow Control

- Flow control statements often start with a part called the condition and are always followed by a block of code called the clause
- Constitute a deterministic, robust system
- Allows you as the programmer to account for every possible situation
- Composed of “blocks”



Flow Control

- There are only 2 Boolean values: *True* or *False*
- Booleans are useful for comparisons — conditional statements
- '==' checks if the values to its left and right are equivalent → returns a boolean
  - Not to be confused with the assignment operator
  - != is the same as ==, except it checks if two values are the *not the same* → boolean

```
>>> 'hello' == 'hello'
True
>>> 'hello' == 'Hello'
False
>>> 'dog' != 'cat'
True
>>> True == True
True
>>> True != False
True
>>> 42 == 42.0
True
>>> 42 == '42'
False
```

Boolean Operators

- Boolean operators:
  - “and”
  - “or”
- “in” operator
- “is” operator
- “not” operator
  - !
  - >>> not not not not True
  - True
  - >>> not True
  - False

Expression	Evaluates to . . .
True and True	True
True and False	False
False and True	False
False and False	False

Expression	Evaluates to . . .
True or True	True
True or False	True
False or True	True
False or False	False

Indentation and if Statements

- So far, indentations and spaces don’t matter in Python
- print("Hi "   +   "everyone   !"   ) works
- However, this isn’t always the case
- Consider an ***if statement***
  - An if statement could be read as, “If this condition is true, execute the code in the clause.” Note the condition is followed by a colon

```
# Works:
if name == ‘Rob’:
    print('Hi, Rob.')
# Raised Error (nothing in the clause):
if name == ‘Rob’:
print('Hi, Rob.')
```

else Statements

- An if clause can optionally be followed by an else statement
- else clause is executed only when the preceding if statement’s condition==False
- An else statement could be read as, “If this condition is true, execute this code. Otherwise, execute that code.”
- Note else statements are also followed by a colon, and an indented block of code on the next line(s) → the else clause

```
if name == ‘Rob’:
    print('Hi, Rob.')
else:
    print('I don’t know you.')
```

elif Statements

- Sometimes you want to consider multiple possible if statements
- The elif statement is an “else if” statement that follows an if/elif statement
- It provides another condition that is checked only if all of the previous conditions were False
- Same syntax applies for elif statements as do for if and else statements

```
if name == ‘Rob’:
    print('Hi, Rob.')
elif name == ‘Josh’:
    print(‘Hi Josh’)
else:
    print('I don’t know you.')
```

while Loops

- Suppose you want a block of code to repeat over and over, so long as it follows a condition -- a while loop does just that
- Followed by colon and indents
- Be wary of infinite loops in your programs
    - When the condition in your loop is never evaluated as False, so the loop runs indefinitely (it can crash your computer!)

```
# Can anyone decipher what this program is doing? Hint: Friedrich Gauss...
count, i = 0, 1
while (i <= 100):
        count = count + i
        i = i + 1
print(count)
```

break and continue Statements

- *break*: A shortcut to make your program “break out” of the while loop it’s currently executing
- *continue*: used inside loops and causes a program to jump back to the top of the loop depending on the conditional the continue keyword is nested within. continue statements are less useful, at least in my personal opinion

```
while True:
    print('Who are you?')
    name = input()
  ❶ if name != 'Joe':
      ❷ continue
    print('Hello, Joe. What is the password? (It is a fish.)')
  ❸ password = input()
    if password == 'swordfish':
      ❹ break
❺ print('Access granted.')
```

- Note that if you don’t enter the correct username and password when prompted by the console, the program will get stuck in an infinite loop
- *Hitting Ctrl + C on the terminal will cause the infinite loop to stop after sending a KeyboardInterrupt error

for Loops and range()

- The while loop keeps going until its condition is broken (hence its name)
- What if you want to execute a block of code only a known, finite number of times? For loops!
- Know that any code you write with a for loop can be written with a while loop and vice versa
- break and continue statements work in for loops as well
- TL;DR: use for loops when you know how many times you want a loop to execute; otherwise use a while loop
- range() returns a sequence/special type of list

```
# note that this is the same code written by our while loop earlier
count = 0
for i in range(101):
  count += i
print(count)
```

More on range()

- The range() function takes a maximum of 3 parameters: start, stop, & step size
- You normally only will be specifying 1 parameter, which is the “stop” - defaulting start to 0 and step size to 1
- Note that the returned sequence is not inclusive of the stop parameter, but it is of the start parameter

```
for i in range(0, 10, 2):
    print(i)
```

Console:

```
0
2
4
6
8
```

More on range()

- By specifying a negative step size and when start > stop, you can go backwards:

```
for i in range(5, 0, -1):
    print(i)
```

Console:

```
5
4
3
2
1
0
```

Random

- We won’t cover modules and/or libraries in this bootcamp, but know that *importing* the *random* module is necessary for random functionalities in Python
- The random module gives us access to its randint() function

```
import random
for i in range(5):
    print(random.randint(1, 10))
```

Sample Output:

```
8
7
2
5
2
```

sys.exit()

- Causes a program to terminate even before it reaches the last line of code

Your Turn!

- Use the programming knowledge you've gained already to build a traditional rock, paper, and scissors game
- Print out all actions, player instructions, and scores to the console
- Delineate a plan first, maybe even write out some logic on paper
    - How will I set it up so that the player can play against the computer?
    - When and where would I apply the things I've learned thus far?
    - For loops or while loops?
    - Hint: use the random module
    - Ask questions in class!