

CSCI 127: Joy and Beauty of Data

Lecture 6: Iteration

Reese Pearsall

Snowmester 2020

<https://reese.github.io/classes/127/main.html>

Announcements

- Lab 3 due **TONIGHT** 11:59 PM
- Program 1 due date moved to Thursday 12/10 @11:59 PM
Lab 4 also due on Thursday 12/10 @ 11:59 PM
- Lab 1 and 2 grades on D2L (solution videos posted)
- Some practice exam questions have also posted

Today

Intro to iteration

WHEN YOU FORGET TO CAPITALYZE THE
BOOL IN PYTHON



True != true

A few observations from the first couple of assignments

Variable Names

The beginning character of variables should always be lowercase

Boat_Name → boat_name
Boat_Name → boatName
Boat_Name → boat_Name
Boat_Name → boatname
Name → name

There are standards and conventions for naming things in Python. If we capitalize our variable names, it usually indicates that it is something else (something we will talk about later)

Same goes for naming functions!

A few observations from the first couple of assignments

File Naming

Remember to follow the format for naming your .py when you submit

`YourFirstName-YourLastName-LabX.py`

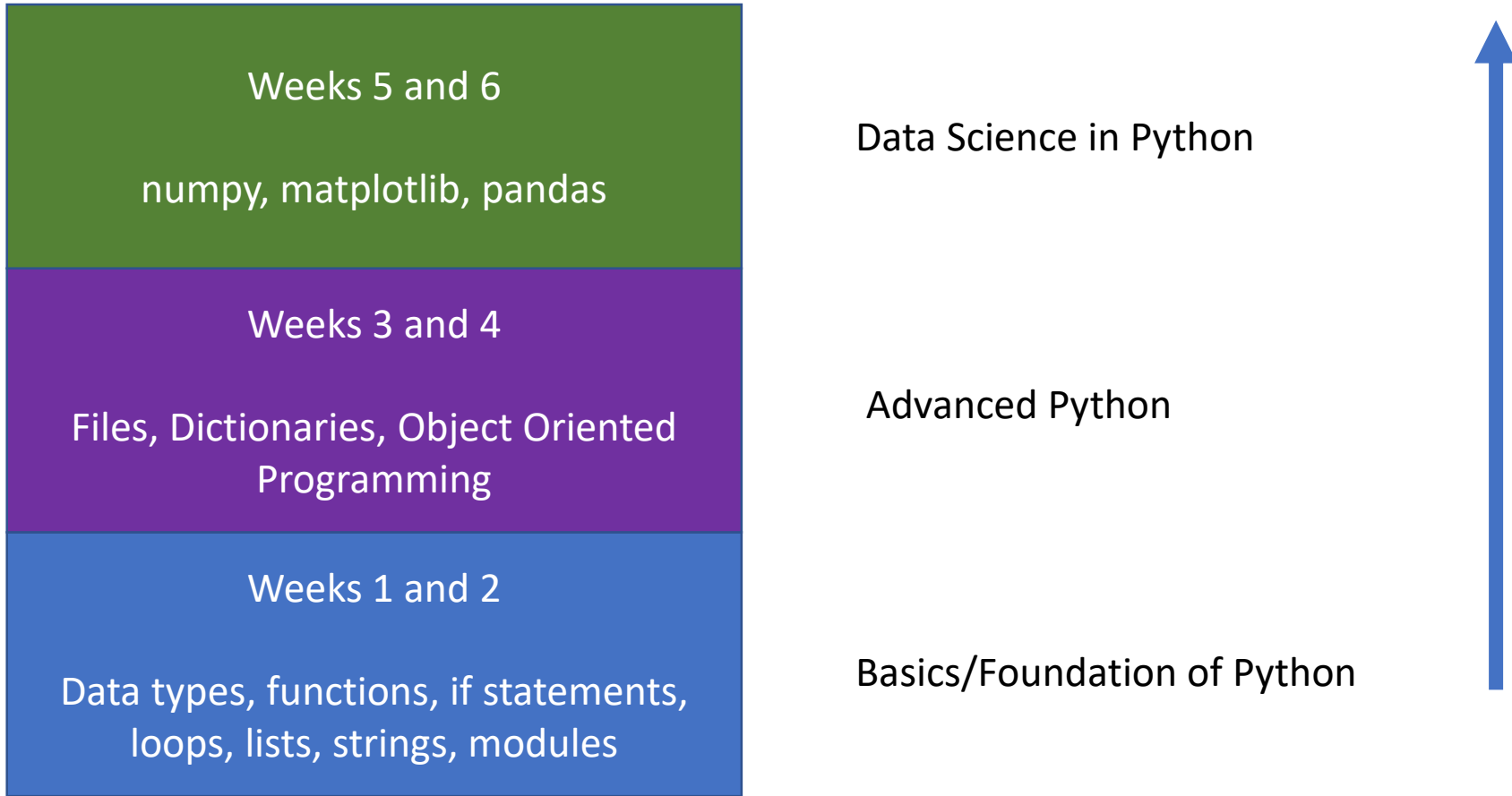
A few observations from the first couple of assignments

File Naming

Remember to follow the format for naming your .py when you submit

`YourFirstName-YourLastName-LabX.py`

Where we are in the class



Intro to Iteration

Often times, we want to repeat a certain block of code

Iteration (**loops**) allows us to repeat code and make our solution more efficient

May want to repeat a certain number of times (definite) or an unknown number of times (indefinite)

for loops

while loops

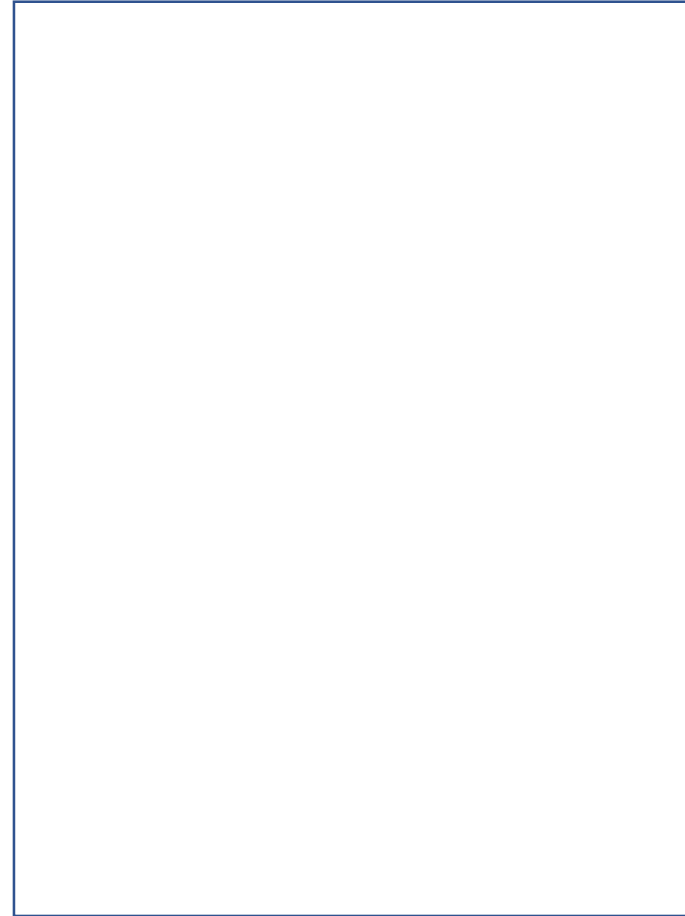
For example, we made a program that checks to a year is a leap year ?

What about a program that finds all leap years between 1900 to 2020 ?

A basic for loop

```
for i in range(5):  
    print("hello")
```

Output



A basic for loop

```
for i in range(5):  
    print("hello")
```

Output

```
hello  
hello  
hello  
hello  
hello
```

Control flow of a for loop

```
for i in range(5):  
    print("hello")  
    print("world")  
    print(i)
```

Output

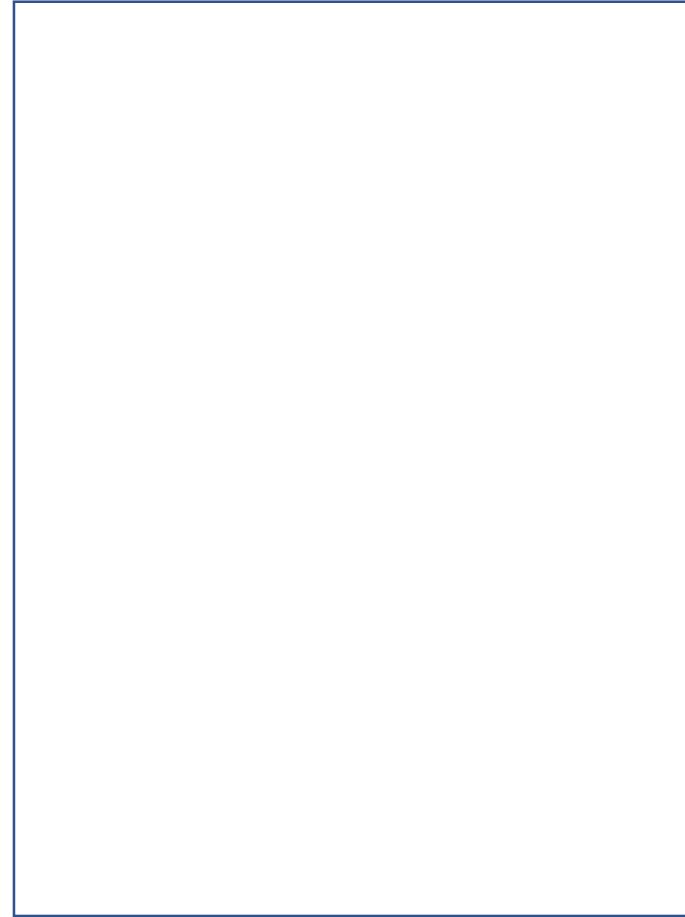


Control flow of a for loop

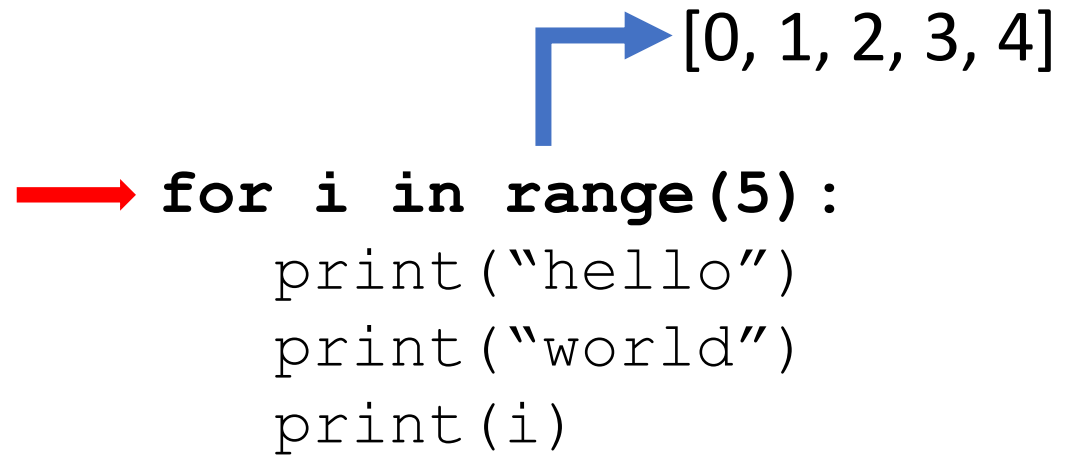
[0, 1, 2, 3, 4]

```
for i in range(5):  
    print("hello")  
    print("world")  
    print(i)
```

Output



Control flow of a for loop



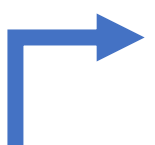
```
→ for i in range(5):  
    print("hello")  
    print("world")  
    print(i)
```

[0, 1, 2, 3, 4]

Output



Control flow of a for loop

Iteration 1 **i = 0**
 **[0, 1, 2, 3, 4]**

```
for i in range(5):  
→ print("hello")  
  print("world")  
  print(i)
```

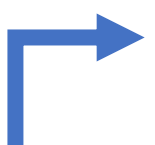
Output

```
hello
```

Control flow of a for loop

Iteration 1

$i = 0$

 `[0, 1, 2, 3, 4]`

```
for i in range(5):  
    print("hello")  
→ print("world")  
    print(i)
```

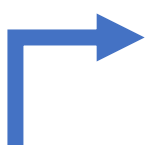
Output

```
hello  
world
```

Control flow of a for loop

Iteration 1

i = 0

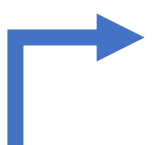
 **[0, 1, 2, 3, 4]**

```
for i in range(5):  
    print("hello")  
    print("world")  
→ print(i)
```

Output

```
hello  
world  
0
```

Control flow of a for loop

Iteration 1 $i = 0$
 `[0, 1, 2, 3, 4]`

```
for i in range(5):  
→ print("hello")  
  print("world")  
  print(i)
```

REPEAT!

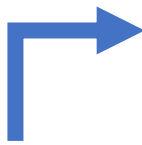
Output

```
hello  
world  
0
```


Control flow of a for loop

Iteration 5

$i = 4$

 `[0, 1, 2, 3, 4]`

```
for i in range(5):  
    print("hello")  
    print("world")  
→ print(i)
```

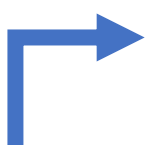
REPEAT ??

Output

```
hello  
world  
0  
hello  
world  
1  
hello  
world  
2  
hello  
world  
3  
hello  
world  
4
```

Control flow of a for loop

Iteration 5 $i = 4$

 [0, 1, 2, 3, 4]

```
for i in range(5):  
    print("hello")  
    print("world")  
    print(i)
```

REPEAT ??

No! we've looped 5 times
already (exhausted our range
of numbers)

Output

```
hello  
world  
0  
hello  
world  
1  
hello  
world  
2  
hello  
world  
3  
hello  
world  
4
```

Control flow of a for loop

See lecture for entire walkthrough

```
for i in range(5):  
    print("hello")  
    print("world")  
    print(i)
```

Program Done!

Output

```
hello  
world  
0  
hello  
world  
1  
hello  
world  
2  
hello  
world  
3  
hello  
world  
4
```

The range () function

range () format:

```
range(start, stop, step)
```

range (n)

Generates a list of integer number from 0 to n (not including n)

```
range(10) → [0,1,2,3,4,5,6,7,8,9]
```

range (m, n)

Generates a list of integer number from m to n (not including n)

```
range(3, 10) → [3,4,5,6,7,8,9]
```

range (m, n, s)

Generates a list of integer number from m to n (not including n) and increments by s each time

```
range(2, 10, 2) → [2,4,6,8]
```

Example: List of squares

(you might find this example helpful for lab 4 and program 1)

Write a program that will generate a list of squared numbers up from 1 to some user defined **n**

For example. `list_of_squares(6)` should print out something like this:

List of squares up to 6:

1. 1
2. 4
3. 9
4. 16
5. 25
6. 36

Examples

Write a function, `is_prime`, that takes a single integer argument and returns `True` when the argument is a *prime number* and `False` otherwise.

Rewrite the function `sumTo(n)` that returns the sum of all integer numbers up to and including n . This time use the accumulator pattern.

(You might find this example helpful for program 1...)

Write a program that will print out whether a number is even or odd for all number from 1 to 100

Announcements (Tuesday)

- Program 1 and Lab 4 due on Thursday 12/10 @11:59 PM
- Program 2 (Lists & Iteration) has been posted
-> Due on 12/16 @ 11:59 PM

Today

While loops, more on for loops, and more examples 😊

```
for i in range(8):
```



(meme made by reese)

(purple sus)

“Unnecessary Code”

On assignments such of programs you can get docked points for having unnecessary code

- Comments explaining your solution are **NOT** unnecessary. *You should be commenting your code*
- Commenting out large blocks of code and leaving it in your submission is unnecessary code
- Functions or lines of code that don't do anything or are never executed is unnecessary code

While loops

- Continually repeat code if a condition is True
- If the condition ever becomes false, stop looping
- Good to use when you don't know how long you need to loop for
- Can cause you some annoying problems (infinite loops, etc)

Control Flow of a While loop

```
x = 0
while (x < 5):
    print("still looping")
    print(x)
    x += 1
```

Output



Control Flow of a While loop

Current value of x:

x = 0

Output

→ x = 0

```
while (x < 5):  
    print("still looping")  
    print(x)  
    x += 1
```

A large empty rectangular box with a thin blue border, intended for displaying the output of the code.

Control Flow of a While loop

Current value of x:

x = 0

Output

```
x = 0  
→ while (x < 5):  
    print("still looping")  
    print(x)  
    x += 1
```



Control Flow of a While loop

Current value of x:

x = 0

Output

```
x = 0  
→ while (x < 5):  
    print("still looping")  
    print(x)  
    x += 1
```

Is $0 < 5$?

Check our looping condition!

Control Flow of a While loop

Current value of x:

x = 0

Output

```
x = 0  
→ while (x < 5):  
    print("still looping")  
    print(x)  
    x += 1
```

Is 0 < 10 ?

True

→ run the code in body of loop

Check our looping condition!

Control Flow of a While loop

Current value of x:

x = 0

Output

still looping

```
x = 0
while (x < 5):
    → print("still looping")
    print(x)
    x += 1
```

Control Flow of a While loop

Current value of x:

x = 0

```
x = 0
while (x < 5):
    print("still looping")
→ print(x)
    x += 1
```

Output

```
still looping
0
```


Control Flow of a While loop

Current value of x:

x = 0

x = 0 + 1

```
x = 0
while (x < 5):
    print("still looping")
    print(x)
    → x += 1
```

Output

```
still looping
0
```

Control Flow of a While loop

Current value of x:

x = 1

```
x = 0
while (x < 5):
    print("still looping")
    print(x)
→ x += 1
```

Output

```
still looping
0
```

Control Flow of a While loop

Current value of x:

x = 1

Output

```
still looping  
0
```

```
x = 0
```

```
→ while (x < 5):  
    print("still looping")  
    print(x)  
    x += 1
```

REPEAT

Is $1 < 5$?

Check our looping condition!

Control Flow of a While loop

Current value of x:

x = 1

Output

```
still looping  
0
```

```
x = 0
```

```
→ while (x < 5):  
    print("still looping")  
    print(x)  
    x += 1
```

REPEAT

Is $1 < 5$?

True

→ run the code in body of loop

Check our looping condition!

Control Flow of a While loop

See lecture for entire walkthrough

Current value of x:

x = 5

```
x = 0  
→ while (x < 5):  
    print("still looping")  
    print(x)  
    x += 1
```

Is $5 < 5$?

Check our looping condition!

Output

```
still looping  
0  
still looping  
1  
still looping  
2  
still looping  
3  
still looping  
4
```

Control Flow of a While loop

Current value of x:

x = 5

```
x = 0
while (x < 5):
    print("still looping")
    print(x)
    → x += 1
```

Is 5 < 5 ?

False

→ Stop looping !!!

Check our looping condition!

Output

```
still looping
0
still looping
1
still looping
2
still looping
3
still looping
4
```

Control Flow of a While loop

Current value of x:

x = 5

```
x = 0
while (x < 5):
    print("still looping")
    print(x)
    x += 1
```



All done!

Output

```
still looping
0
still looping
1
still looping
2
still looping
3
still looping
4
```

While loop examples

Write a Python program that generates a random number between 1 and 10. The program should repeatedly ask the user to guess what the number is until the user guesses correctly. When the user guesses correctly, the program should print a message that shows how many tries it took. For example, the message might be *Congratulations! That took 6 guesses.*

Write a python program that will simulate a game of rock paper scissors against a computer. The user is given 3 lives and if the user ever loses to the computer, they lose a life. The user gets a point every time they beat the computer. The game ends when the user loses all of their lives

break and continue

Two built-in python keywords for looping

break- exit current loop you are in

continue- exit current iteration and move to next iteration

Note from previous professor who created this class:

“Try to avoid using these two Python constructs. Typically, there is a better way to solve the problem.”

More on for loops

We will take about more things we can use for loops for later in the snowmester

For loops can be used to iterate through:

Strings:

```
s = "hello world"
for each_char in s:
    print(each_char)
```

Output

```
h
e
l
l
o

w
o
r
l
d
```

Lists:

```
s = ["Reese", "Jane", "Jeff"]
for each_name in s:
    print(each_name)
```

Output

```
Reese
Jane
Jeff
```

Generating random numbers

We will talk more about this on Friday....

1. Be sure to include `import random` at the top of your program
2. To generate a random integer number between X and Y (including X and Y)

```
rand_num = random.randint(X, Y)
```

Example: Generating a random number between 1 and 10

```
rand_num = random.randint(1, 10)  
print(rand_num)
```

while loop vs for loops

For loops

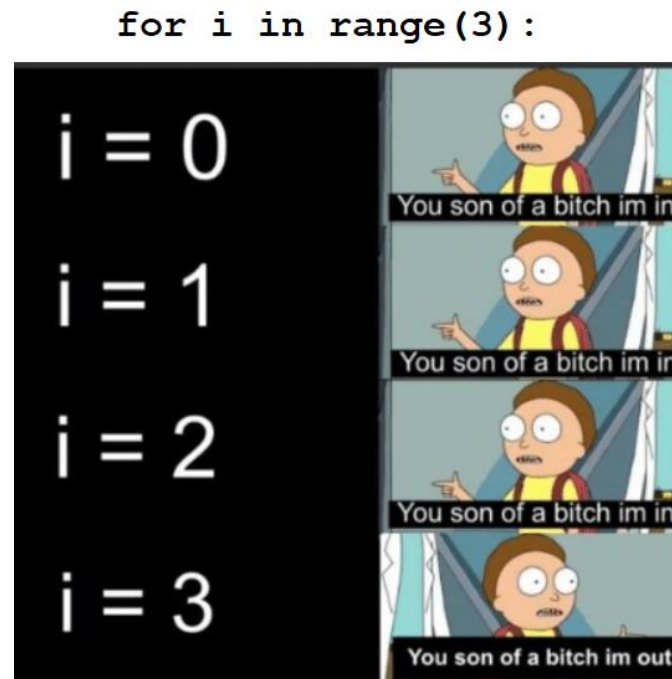
- Good if you know in advanced how many times you need to loop
- Good for iterating through lists, strings, dictionaries, etc.
- Don't have to worry about infinite looping**

While loops

- Good if you **don't** know in advanced how many times you need to loop
- Good for repeating based on a certain condition
- Possibility of infinite looping

** there are ways to achieve an infinite for loop, but you shouldn't have to worry about them

The End



when you forget to write an exit
condition for your while loop



More for loop examples

Write a Python program that asks the user to enter two integers: one for the number of rows and one for the number of columns. The program should then produce a text-based drawing where each position in the drawing is randomly determined with equal probability to be either a "*" or a "-". For example, with 4 rows and 6 columns, the drawing might look like this:

```
**_**  
--**--  
-*_-*-  
---*_*
```

Write a python function `count_p()` that will take in a word. The function will count how many P's are in that word