Data science for everyone

Prof. Jones-Rooy & Prof. Policastro

March 25, 2020

8.2: Statistics

ANNOUNCEMENTS

1. Prof. Policastro takes over the course Monday, March 30
2. All grades and feedback through (and including) the midterm are available
   - Please check them all, contact your TAs with questions
   - Deadline for regrade requests & checking exceptions: **Friday, March 27 EDT**
3. No Lab 5 this week (free 2% for everyone)
4. Next (and last) homework: Homework 3/4, out April 6, due April 27
5. Feedback for future assignments will come through email and Gradescope
   - We will still fetch and submit assignments on nb-grader
6. Be working on project! (See lecture 8.1 for all mini-deadlines)
   - April 6: Have dataset decided on!
   - We *strongly* recommend you get your TAs approval and feedback on your dataset before proceeding with the project!
   - Your TAs will communicate with you about getting feedback on and approval of your dataset

Outline

1. Law of Large Numbers (from 8.1)
2. LLN intuition through simulation
3. Loops!

SEE SLIDES FROM LECTURE 8.1

Outline

1. Law of Large Numbers (from 8.1)
2. LLN intuition through simulation
3. Loops!

LAW OF LARGE NUMBERS

- If we repeat an experiment many, many times, the proportion of times we see a given outcome (e.g., a 3 or a 6) ***empirically*** will converge to the ***theoretical*** value we would expect (1/6, or 16.6%)
- How large is large enough?
  - It depends on your tolerance for risk/error/uncertainty
  - And on the variance of the phenomenon we are interested in
    - It takes less time for the empirical distribution of coin flips to look like the theoretical, expected distribution compared to dice rolls
    - To get some intuition, let's compare coin flips to dice rolls to (drumroll … drumroll…) a deck of cards!!

SEE 8.2 EXAMPLE CODE!

52 cards in a deck

The probability of pulling any given card is 1/52

The theoretical distribution is ***uniform*** with a 1.9% chance of pulling any one card (in a fair deck)

Be thinking about: How long does it take our simulated ***empirical distribution*** to ***converge*** to the expected ***theoretical distribution***?

Coins: ~100	Dice: ~1,000	Cards: ~100,000

Outline

1. Law of Large Numbers (from 8.1)
2. LLN intuition through simulation
3. Loops!

CONTROL FLOW STATEMENTS

- In computer science, these are statements that result in a choice being made as to which of two or more paths to follow
- We've already seen one kind of these: ***Conditional statements*** (if …else)
- Now we'll learn another very common (and useful) kind: ***Loops***

Reminder: Conditional statements

if

```
if test expression:
     statement(s)
```

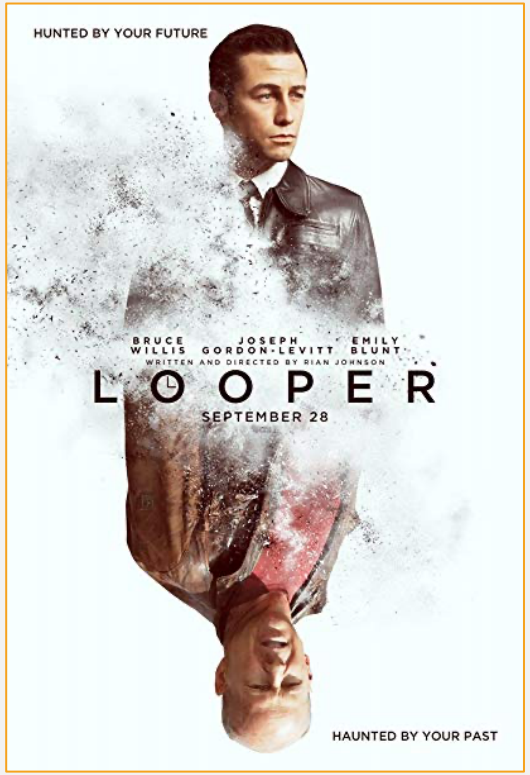
if…else

```
if test expression:
     Body of if
else:
     Body of else
```

if…else if…else

```
if test expression:
     Body of if
elif test expression:
     Body of elif
else:
     Body of else
```

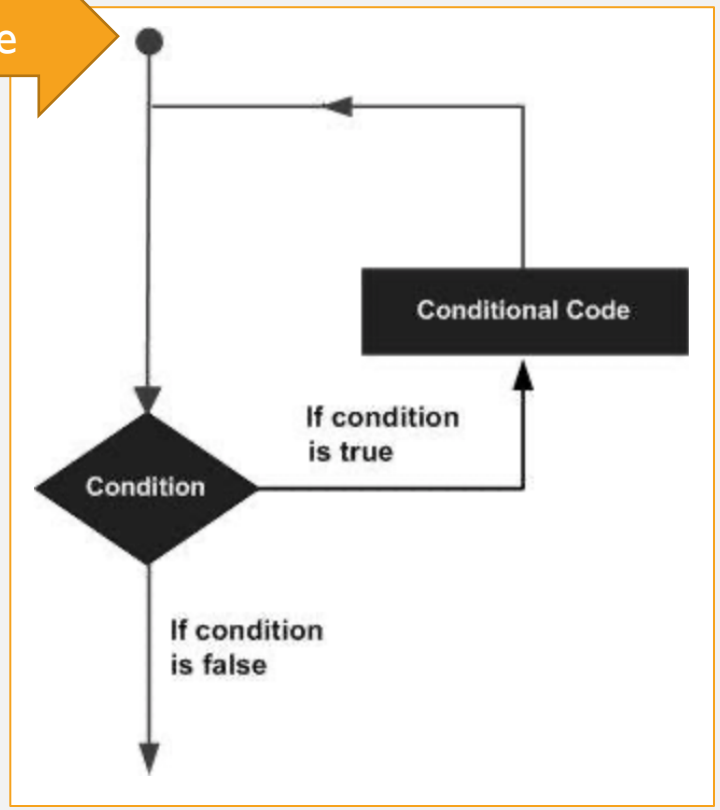
LOOPS



- Generally: Loops are ways to execute repeated, or looping, lines of code
- More formally: A loop statement allows us to execute a statement or group of statements multiple times
- Why we might want such a thing:
  - Most statements are executed sequentially in computer programs
  - But there may be situations where we want to execute a block of code multiple times
  - Programming languages offer some way of allowing for more complicated execution paths

LOOPS IN PYTHON

Previous code



- For
- While
- Nested
- Infinite

Further optional resources here!

FOR LOOPS

- A **For Loop** iterates over a sequence multiple times
- For Loops are useful when we know the number of iterations we want
  - In a moment, we will see While Loops, which are useful when we don't know how many iterations will be required until a condition is met
- **For Loop** syntax:

```
for [iterating variable] in [sequence]:
[do something]
```

- The thing being done will be executed until the sequence is over

```
for i in range(10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
a = [1, 6, 8, 10, 34]
for i in a:
    print(i)
```

```
1
6
8
10
34
```

```
a = [1, 6, 8, 10, 34]
for i in a:
    print(2*i)
```

```
2
12
16
20
68
```

FOR LOOPS EXAMPLES

It works for strings

```
mystring = ("Let's iterate")
for i in mystring:
    print(i)
```

```
L
e
t
'
s

i
t
e
r
a
t
e
```

And arrays

```
myarray = np.array([1, 2, 3, 4])
for i in myarray:
    print(i)
```

```
1
2
3
4
```

Note: it doesn't have to be `i`!

```
myarray = np.array([1, 2, 3, 4])
for z in myarray:
    print(z)
```

```
1
2
3
4
```

```
myarray = np.array([1, 2, 3, 4])
for r in myarray:
    print(r)
```

```
1
2
3
4
```

RANGES ARE BACK, TOO!

Refresh on ranges

```
x = np.arange(10)
x
```

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

```
x = np.arange(2, 10)
x
```

```
array([2, 3, 4, 5, 6, 7, 8, 9])
```

```
1 x = np.arange(2, 10, 2)
2 x
```

```
array([2, 4, 6, 8])
```

```
for i in range(10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
for x in range(10):
    print(x)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
for x in range(2, 10):
    print(x)
```

```
2
3
4
5
6
7
8
9
```

```
for x in range(2, 10, 2):
    print(x)
```

```
2
4
6
8
```

COMBINING WITH CONDITIONAL STATEMENTS

```
for x in range(10):
  print(x)
else:
  print('And now my range is done')
```

```
0
1
2
3
4
5
6
7
8
9
And now my range is done
```

```
for x in range(10):
  print(x)
else:
  print('And now his range is ended')
```

```
0
1
2
3
4
5
6
7
8
9
And now his range is ended
```

CAN INVOLVE MORE THAN ONE VARIABLE

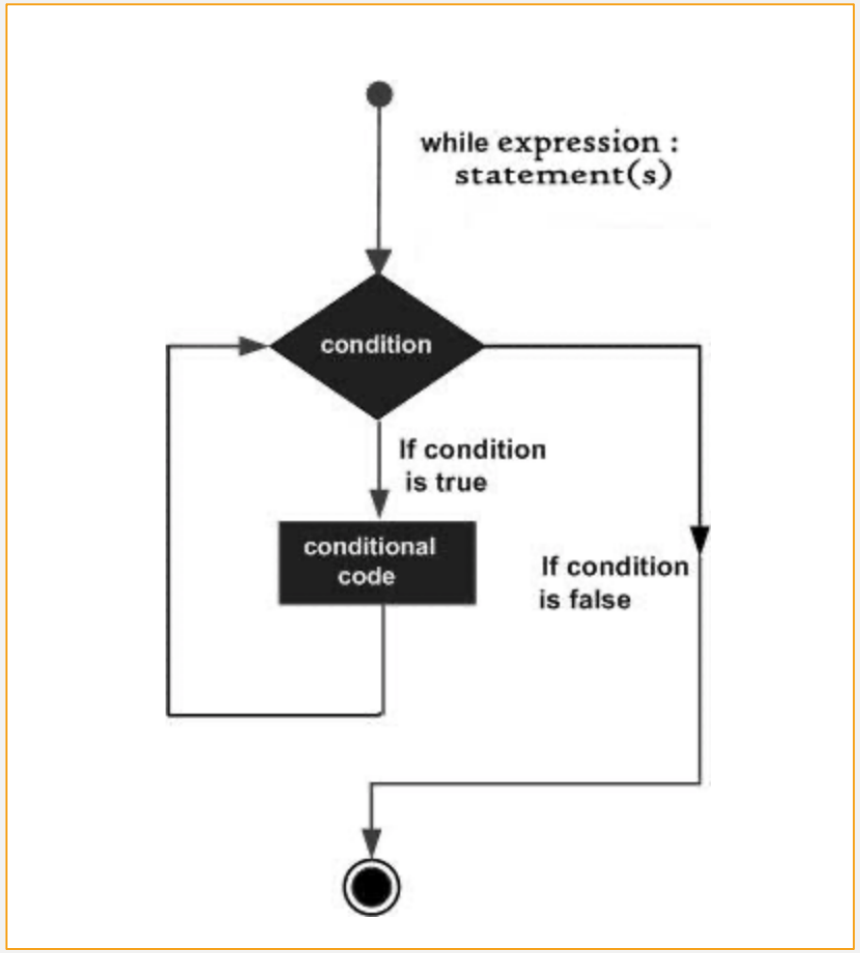
```
adj = ['average', 'tall', 'short']
seinfeld = ['Jerry', 'Kramer', 'George', 'Elaine']

for x in adj:
    for y in seinfeld:
        print(x, y)
```

```
average Jerry
average Kramer
average George
average Elaine
tall Jerry
tall Kramer
tall George
tall Elaine
short Jerry
short Kramer
short George
short Elaine
```

WHILE LOOPS

- A while loop is used to execute a block of statements repeatedly until a given condition is satisfied
- Repeats a statement or group of statements until a given condition is TRUE
  - Boolean alert!
- Syntax:

```
while expression:
    statements
```



SOME RIVETING EXAMPLES

```
x = 0
while x<9:
    print(x)
    x = x+1
```

```
0
1
2
3
4
5
6
7
8
```

```
x = 0
while(x<9):
    print(x)
    x = x+1
```

```
0
1
2
3
4
5
6
7
8
```

```
x = 0
while(x<=9):
    print(x)
    x = x+1
```

```
0
1
2
3
4
5
6
7
8
9
```

```
x = 0
while(x==9):
    print(x)
    x = x+1
```

(nothing!)

```
x = 9
while(x==9):
    print(x)
    x = x+1
```

```
9
```

MORE RIVETING EXAMPLES

```
x = 0
while (x < 9):
   print('The value for x is:', x)
   x = x + 1

print('Zai jian!')
```

```
The value for x is: 0
The value for x is: 1
The value for x is: 2
The value for x is: 3
The value for x is: 4
The value for x is: 5
The value for x is: 6
The value for x is: 7
The value for x is: 8
Zai jian!
```

```
x = 0
while (x > 9):
   print('The value for x is:', x)
   x = x + 1

print('Zai jian!')
```

```
Zai jian!
```

P.S.

zai jian = 再见 = 👋 = goodbye!

NESTED LOOPS

- We can write one loop inside another loop
- Syntax:

```
for iterating_var in sequence:
   for iterating_var in sequence:
      statements(s)
   statements(s)
```

```
while expression:
   while expression:
      statement(s)
   statement(s)
```

NESTED LOOPS

```
for i in range(1, 10):
    for j in range(1, 2):
        print(i, j)
```

```
1 1
2 1
3 1
4 1
5 1
6 1
7 1
8 1
9 1
```

```
for i in range(1, 10):
    for j in range(1, 2):
        print(i + j)
```

```
2
3
4
5
6
7
8
9
10
```

ANOTHER NESTED LOOP

```
i = 2
while(i < 10):
    j = 2
    while(j <= (i/j)):
        if not(i%j): break
        j = j + 1
    if (j > i/j) : print(i, 'is prime')
    i = i + 1

print("Ciao!")
```

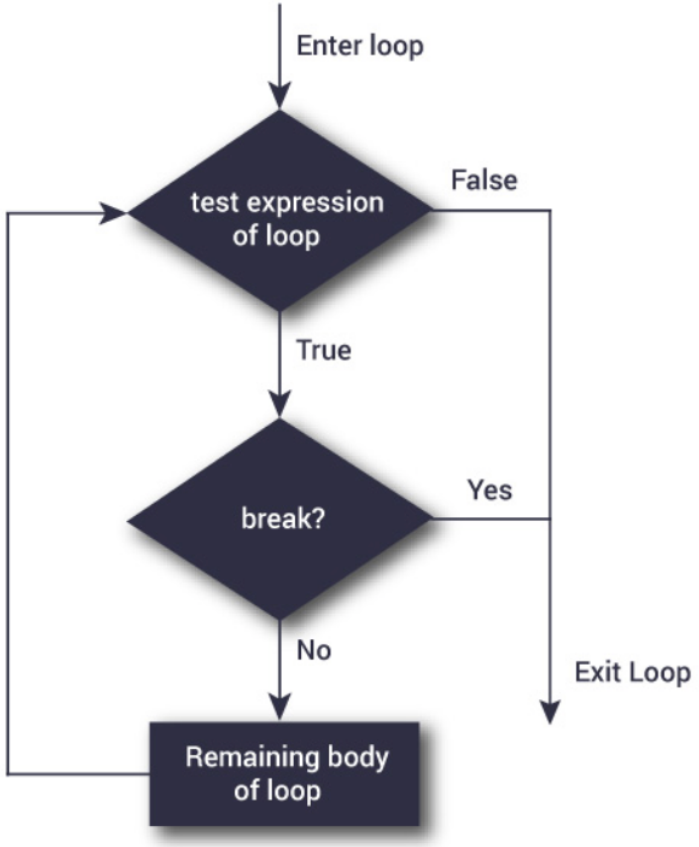
```
2 is prime
3 is prime
5 is prime
7 is prime
Ciao!
```

Useful statements (including because you may see these out there in the world)

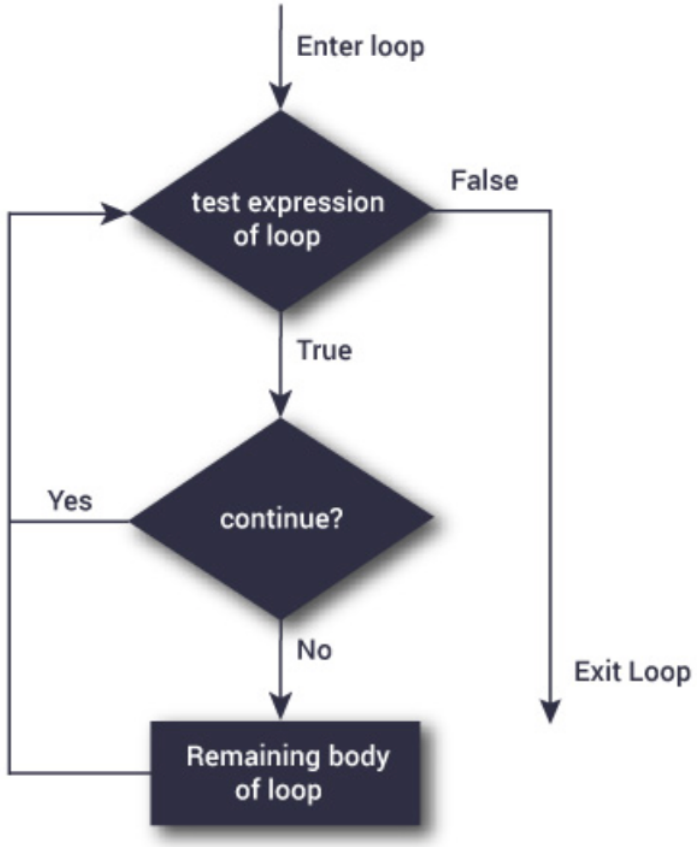
`break`: if you want to stop performing these steps before the loop completes

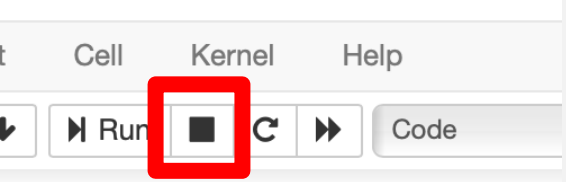
`continue`: rejects all remaining statements in current iteration of the loop and moves control back to the top of the loop

Flowchart of break



Flowchart of continue





INFINITE LOOP

```
In [*]:
```

```
x = 1
while (x > 0):
    print('The value for x is:', x)
    x = x + 1

print('Zai jian!')
```

```
The value for x is: 1
The value for x is: 2
The value for x is: 3
The value for x is: 4
The value for x is: 5
The value for x is: 6
The value for x is: 7
The value for x is: 8
The value for x is: 9
The value for x is: 10
The value for x is: 11
The value for x is: 12
The value for x is: 13
The value for x is: 14
The value for x is: 15
```

A Loop becomes an infinite Loop if it never becomes false

If you write one, you'll have to manually stop it from running (unless you want to use it to keep a program running….)

```
The value for x is: 586215
The value for x is: 586216
The value for x is: 586217
The value for x is: 586218

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (m
<ipython-input-25-603faebd4812> in <module>
      1 x = 1
```

```
x = 0
while x<9:
    print(x)
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

Outline

1. Law of Large Numbers (from 8.1)
2. LLN intuition through simulation
3. Loops!