ECS10

1/30

Upcoming Events

- Assignment Friday
- Checkpoint Thursday 2/7
- Final version Thurs 2/14

while-break loop

```
while True:
    reply = input("Enter h or t:")
    if (reply == "h") or (reply == "t") :
        break
    else:
        print ("not valid")
# now reply is either h or t
```

while-break loop

```
while True:
    .......
    if......:
        break
    .......
```

- Loop continues until break statement, then jumps out
- return – ends a function
- break – ends a loop

How many flips for 10 heads?

- Need to count two things – number of flips, number of heads so far.
- Use a while loop. When should it stop?
- What should it do on tails?
- What should it do on heads?

while loop with continue

```
while heads < 10:
    .......
    if......:
        continue
    .......
```

- When it hits the continue statement, it jumps back to the top of the loop, and continues running
- Different from break, which stops the loop
- Continue just stops this iteration.

Iteration

- Means do something over and over again.
- While loops iterate the block under the while.
- Each time through the block is an iteration.

continue statement

```
while heads < 10:
    coin = randrange(0,2)
    # integer, either 0 or 1
    flips = flips+1
    if coin == 1:
        # tails - ignore it
        continue
    heads = heads+1
```

How many flips do we think we need?

- How many do we need to get one heads?
- So how many do we need for 10 heads?
- Let's see what we get….

Nested loops

```
while iterations < 10000:
    …….
    …….
    while True:
        ……..
        ……..
    …….
    …….
…….
```

Most popular value?

- Should be about 19.5 flips to get 10 heads
- Is it less as often as it is more?
- How often is it exactly 19? 20?

- How often does it take 10, 11, 12…?
- I could have a zillion variables, or I could have… a list!

Save data in a list

- New data type!

```
L = ["cow","horse","mule"]
i = 0
while i < 3:
    print(L[i])
    i = i+1
```

- A list of strings
- i is the index variable

Elements of a list

```
L[0]
```

- Expression for the element at position zero

```
L[0] = "otter"
```

- Changes the element at position zero
- List elements are essentially variables

Indexing a list

```
L = ["cow","horse","mule"]
I = 3
print(L[3])  # crashes!
```

- Have to be careful that the index < length of list

```
if I < len(L):
        print( L[i] )
```

Operators on lists

- Concatenate two lists:

```
L = ["pig"]+["cow"]
```

- List with all repeat elements:

```
L = ["pig"]*5
```

List of possible results of experiment

- Might go up to infinity, so lets stop at, say, 50….
- Extra variable for really big numbers, just in case

- 18 is more popular than 20….?
- Crank it up to 100,000 iterations