ECS 10

2/11

for loop

- On list

```
for animal in [“cow”, “goat”,  “mule”]:
        print(animal)
```

- On string

```
for char in “3,497”:
        if char != “,”:
                print(char, end=“”)
```

for loop

- On range

```
for count in range(1,6):
        print(count)
```

- On file

```
inFile = open(“obesity.tsv”,”r”)
for line in inFile:
        print(line)
```

Behavior depends on data type

- Example: for loop

```
for x in M:
   …
```

  - If M is a list, x is an element
  - If M is a string, x is a character
  - If M is a file, x is a string (a line of the file)
  - If M is a range object (iterator), x is an integer

Behavior depends on data type

- Example: indexing

```
M[2]
```

  - If M is a list, this is the third element
  - If M is a string, this is the third character

Let user pick file to run on

- But.. program has no way of checking whether the user typed the name of a file in same folder except by trying to open it.
- It will crash if the file is not there!
- But programs should not crash!
- Similar to problem we had with getting numbers from the user.

Exceptions

- Python mechanism for handling user input that might crash the program:

```
try:
    inFile = open(inFileName, "r")
except:
    # Gets here if we cannot open the file
    print( "Cannot find file",inFileName )
```

Exceptions

```
try:
    # Command that might cause a crash
except:
    # Executes this block if a crash would have
    # happened!
```

The value None

```
x = None
```

- None is the value you put into a variable to indicate that the variable exists, but it is empty.
- All we can do is test it for equality other values.
- Here we test to see if it is equal to “a” or “b”

When to Use Exceptions

- Use exceptions to handle input you cannot control.
- We have seen the most common scenarios: file names, and converting user-input strings to numbers. You **may not** use exceptions for anything else in this course.
- Most crashes are because there is a something wrong with your program. **Fix the bug**, don’t put it inside an exception.

Converting Strings

```
popStr = input(‘Enter the population: ‘)
try:
    pop = float(popStr)         # Try conversion
except:
    # Conversion failed!
    print (‘Not a number.’)
    pop = None
```

Getting input to functions

```
def canBeFloat(s):
```

- s is the parameter of canBeFloat; what the input is called inside the function.
- Might be called anything outside the function, when it is called:

```
if not canBeFloat(reply):
```

- reply is the argument of the function canBeFloat in this particular line of a program using the function.

Getting output from functions

```
return False
```

- The value produced by the function follows the return command.
- If there is nothing following the return command, then the value is None!

canBeInt

```
def canBeInt(s):
  try:
    int(s)
  except:
    return False
  return True
```

import

```
from inputCheck import canBeInt
```

- Import tells Python to put the function canBeInt into your program, from file inputCheck.