Week 8
CSC111 - Spring 2015

Dominique Thiébaut
dthiebaut@smith.edu
Midterm Exams Returned **Wed** 3/26
1:00 p.m.
FH356
Review Loops & Booleans (Chapter 8)
For loops in context

Applications

While Loops for Robustness

Break & Continue
For-Loops

Items: [ dog, cat, horse, hen, pig ]
For-Loops

Items: [ dog, cat, horse, hen, pig ]
For-Loops

Items: [ dog, cat, horse, hen, pig ]

```python
list = [ dog, cat, horse, hen, pig ]
for x in list:
    process( x )
```
For-Loops

Items: [ dog, cat, horse, hen, pig ]
For-Loops

Items: [ dog, cat, horse, hen, pig ]

Index Generator
For-Loops

Items: [ dog, cat, horse, hen, pig ]

```python
list = [ dog, cat, horse, hen, pig ]
for i in range( 1, len(list), 3):
    x = list[i]
    process( x )
```
For loops in context

Applications

While Loops for Robustness

Break & Continue
Applications
Count Matching Items

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, cat, pigeon, hen, sheep ]
Count Matching Items

Items1: [ dog, cat, horse, hen, pig ]
Items2: [ dog, cat, pig, hen, sheep ]

Exact Place Matching
items1 = [ dog, cat, horse, hen, pig ]
items2 = [ dog, cat, pig, hen, sheep ]

count = 0
for x in items1 and y in items2:
    if x == y:
        count += 1
items1 = [ dog, cat, horse, hen, pig ]
items2 = [ dog, cat, pig, hen, sheep ]

count = 0
for x in items1 and y in items2:
    if x == y:
        count += 1

Why?
items1 = [ dog, cat, horse, hen, pig ]
items2 = [ dog, cat, pig, hen, sheep ]

count = 0
for i in range( len(items1) ):
    if items1[i]==items2[i]:
        count += 1
What if the lists do not have the same length?

```python
#           0  1  2  3  4
items1 = [ dog, cat, horse, hen, pig ]
items2 = [ dog, cat, pig ]

count = 0
for ???:
    if items1[i]==items2[i]:
        count += 1
```
What is the risk?
What could go wrong?
What kind of error?
What is the risk?
What could go wrong?
What kind of error?
What if the lists do not have the same length?

```python
#           0    1    2      3    4
dog, cat, horse, hen, pig ]
items2 = [ dog, cat, pig ]

count = 0
for i in range( len( items1 ) ):
    if items1[i]==items2[i]:
        count += 1
```
items1 may not be the longest list

```python
#           0    1    2    3    4
items1 = [ dog, cat, horse]
items2 = [ dog, cat, pig, hen, pig ]

count = 0
for i in range( len( items1 ) ):
    if items1[i]==items2[i]:
        count += 1
```
## 2. Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here:

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>abs()</code></td>
<td><code>dict()</code></td>
<td><code>help()</code></td>
<td><code>min()</code></td>
<td><code>setattr()</code></td>
</tr>
<tr>
<td><code>all()</code></td>
<td><code>dir()</code></td>
<td><code>hex()</code></td>
<td><code>next()</code></td>
<td><code>slice()</code></td>
</tr>
<tr>
<td><code>any()</code></td>
<td><code>divmod()</code></td>
<td><code>id()</code></td>
<td><code>object()</code></td>
<td><code>sorted()</code></td>
</tr>
<tr>
<td><code>ascii()</code></td>
<td><code>enumerate()</code></td>
<td><code>input()</code></td>
<td><code>oct()</code></td>
<td><code>staticmethod()</code></td>
</tr>
<tr>
<td><code>bin()</code></td>
<td><code>eval()</code></td>
<td><code>int()</code></td>
<td><code>open()</code></td>
<td><code>str()</code></td>
</tr>
<tr>
<td><code>bool()</code></td>
<td><code>exec()</code></td>
<td><code>issubclass()</code></td>
<td><code>ord()</code></td>
<td><code>sum()</code></td>
</tr>
<tr>
<td><code>bytearray()</code></td>
<td><code>filter()</code></td>
<td><code>issubclass()</code></td>
<td><code>pow()</code></td>
<td><code>super()</code></td>
</tr>
<tr>
<td><code>bytes()</code></td>
<td><code>float()</code></td>
<td><code>iter()</code></td>
<td><code>print()</code></td>
<td><code>tuple()</code></td>
</tr>
<tr>
<td><code>callable()</code></td>
<td><code>format()</code></td>
<td><code>len()</code></td>
<td><code>property()</code></td>
<td><code>type()</code></td>
</tr>
<tr>
<td><code>chr()</code></td>
<td><code>frozenset()</code></td>
<td><code>list()</code></td>
<td><code>range()</code></td>
<td><code>vars()</code></td>
</tr>
<tr>
<td><code>classmethod()</code></td>
<td><code>getattr()</code></td>
<td><code>locals()</code></td>
<td><code>repr()</code></td>
<td><code>zip()</code></td>
</tr>
<tr>
<td><code>compile()</code></td>
<td><code>globals()</code></td>
<td><code>map()</code></td>
<td><code>reversed()</code></td>
<td><code>_import_()</code></td>
</tr>
<tr>
<td><code>complex()</code></td>
<td><code>hasattr()</code></td>
<td><code>max()</code></td>
<td><code>round()</code></td>
<td></td>
</tr>
<tr>
<td><code>delattr()</code></td>
<td><code>hash()</code></td>
<td><code>memoryview()</code></td>
<td><code>set()</code></td>
<td></td>
</tr>
</tbody>
</table>

**abs(x)**

[https://docs.python.org/3.4/library/functions.html](https://docs.python.org/3.4/library/functions.html)
items1 = [ dog, cat, horse]
items2 = [ dog, cat, pig, hen, pig ]

count = 0
len1 = len( items1 )
len2 = len( items2 )
for i in range( min( len1, len2 ) ):
    if items1[i]==items2[i]:
        count += 1
Applications, #2
Count Matching Misplaced Items

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ cat, pig, pigeon, hen, dog ]
Count Matching
Misplaced Items

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]
Algorithm

0 1 2 3 4

Items1: [ dog, cat, horse, hen, pig ]

0 1 2 3 4

Items2: [ dog, pig, pigeon, hen, cat ]
Algorithm

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]
Algorithm

count: 1
Algorithm

i

0 1 2 3 4

Items1: [ dog, cat, horse, hen, pig ]

j

0 1 2 3 4

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

Items1: [dog, cat, horse, hen, pig]

Items2: [dog, pig, pigeon, hen, cat]

count: 1
Algorithm

Items1: [dog, cat, horse, hen, pig]

Items2: [dog, pig, pigeon, hen, cat]

count: 1
Algorithm

\[ i \]

0 1 2 3 4

Items1: [ dog, cat, horse, hen, pig ]

\[ j \]

0 1 2 3 4

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 \\
\end{array}
\]

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 \\
\end{array}
\]

\text{Items1: [ dog, cat, horse, hen, pig ]}

\text{Items2: [ dog, pig, pigeon, hen, cat ]}

\text{count: 1}
Algorithm

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

```
i
0 1 2 3 4
Items1: [ dog, cat, horse, hen, pig ]
```

```
0 1 2 3 4
Items2: [ dog, pig, pigeon, hen, cat ]
```

count: 1
Algorithm

items1: [ dog, cat, horse, hen, pig ]
items2: [ dog, pig, pigeon, hen, cat ]

count: / 2
items1 = [ dog, cat, horse, hen, pig]
items2 = [ dog, pig, pigeon, hen, cat]
count = 0

for i in range( len( items1 ) ):
    for j in range( len( items2 ) ):
        if items1[i]==items2[j]:
            count += 1
For loops in context

Applications

**While Loops for Robustness**

Break & Continue
Robust Input
With While Loops

• Example 1: While quantity not valid

• Example 2: While item not in list
# get a positive number from user
x = int( input( "> " ) )

...
What if user enters negative number?

```python
# get a positive number from user
x = int( input( "> " ) )
...
```
Solution: keep on asking until input is ok

```python
# get a positive number from user
x = int(input("> "))
while x < 0:
    x = int(input("Invalid number\n> "))
```
Solution: keep on asking until input is ok

```python
# get a positive number from user
x = int(input("> "))
while x < 0:
    x = int(input("Invalid number\n> "))

# or

while True:
    x = int(input("> "))
    if x > 0:
        break
    print("Invalid input")
```
# get a positive number from user

def getPositiveInt():
    x = int(input("> "))
    while x < 0:
        x = int(input("Invalid number
> "))
    return x

x = getPositiveInt()
Solution 2

```python
# get a positive number from user
def getPositiveInt():
    while True:
        x = int(input("> "))
        if x > 0:
            return x
        print("Invalid number")

x = getPositiveInt()
```

# get a positive number from user

def getPositiveInt():
    while True:
        x = int(input(" > " ))
        if x > 0:
            return x
        print("Invalid number")

x = getPositiveInt()
Robust Input
With While Loops

- Example 1: While quantity not valid
- Example 2: While item not in list
# get a YES/NO answer from user

def getAnswerYesNo():
    x = input( "Continue (Yes/No)? " )

    while ???:
        print( "Invalid input, must be YES or NO" )
        x = input( "Continue (Yes/No)? " )

    return x

ans = getAnswerYesNo()
# get a YES/NO answer from user

def getAnswerYesNo():
    x = input("Continue (Yes/No)? ").upper()

    while ( x in ["YES", "NO"] ) == False:
        print("Invalid input, must be YES or NO")
        x = input("Continue (Yes/No)? ").upper()

    return x

ans = getAnswerYesNo()
def getAnswerYesNo():
    x = input( "Continue (Yes/No)? " ).upper()

    while not ( x in [ "YES", "NO" ] ):
        print( "Invalid input, must be YES or NO" )
        x = input( "Continue (Yes/No)? " ).upper()

    return x

ans = getAnswerYesNo()
For loops in context

Applications

While Loops for Robustness

Break & Continue
Break and Continue
Monopoly = loop
break = ?
continue = ?

http://en.wikipedia.org/wiki/File:German_Monopoly_board_in_the_middle_of_a_game.jpg
Monopoly = loop
break = ?
continue = ?
Monopoly = loop
break = ?
continue = ?
Monopoly = loop
break = ?
continue = ?
Monopoly = loop
break = ?
continue = ?
Monopoly = loop
break = ?
continue = ?
Example
Count number of ints in range (50,75) and (75,100)
Stop when getting -1

# initialize counters
count1 = 0
count2 = 0

# assume user enters numbers between 0 and 100, 
# and enters -1 to stop
while True:
    x = int( input( "> " ) )
    if x == -1:
        break
    if x < 50:
        continue
    if 50 <= x <= 75:
        count1 += 1
    else:
        count2 += 1
print( count1, count2 )
# initialize counters
count1 = 0
count2 = 0

# assume user enters numbers between 0 and 100,
while True:
x = int(input("> " ))
if x == -1:
    break
if x < 50:
    continue
if 50 <= x <= 75:
    count1 += 1
else:
    count2 += 1
print(count1, count2)
Count number of ints in range (50,75) and (75,100)  
Stop when getting -1

```python
# initialize counters
count1 = 0
count2 = 0

# assume user enters numbers between 0 and 100,
while True:
    x = int( input( "> " ) )
    if x == -1:
        break
    if x < 50:
        continue
    if 50 <= x <= 75:
        count1 += 1
    else:
        count2 += 1

print( count1, count2 )
```

The order is important! Why?
We stopped here last time...
A side step: Importing Modules

A Game of Rock, Paper, Scissors

Top-Down

Goal

Listing all Details

Coding starting at the top
Importing Modules

```python
# utils.py

def robustF1():
    ....
    ....

def main():
    print( "Welcome to utils.py!" )
    ....

main()
```

```python
# myApp.py

import utils

def main():
    x = utils.robustF1()

main()

Welcome to utils.py!
...
```
Importing Modules

**utils.py**

```python
def robustF1():
    .......
    .......

def main():
    print("Welcome to utils.py!")
    .......
main()
```

**myApp.py**

```python
import utils
def main():
    x = utils.robustF1()
main()
```
Importing Modules

**utils.py**

```python
def robustF1():
    ....
    ....

def main():
    print("Welcome to utils.py!")
    ....
main()
```

**myApp.py**

```python
import utils

def main():
    x = utils.robustF1()

main()
```

Welcome to utils.py!

...
### Importing Modules

```python
# utils.py

def robustF1():
    ........
    ........

def main():
    print( "Welcome to utils.py!" )
    ........

if __name__=="__main__":
    main()

# myApp.py

import utils

def main():
    x = utils.robustF1()
    print( "Let's play!" )

main()
```

Let's play
def robustF1():
    ...
    ...

def main():
    print("Welcome to utils.py!"
    ...

if __name__=='__main__':
    main()

---

import utils

def main():
    x = utils.robustF1()
    print("Let's play!"

main()
if __name__=='__main__':
    main()

If this module is imported, don't call its main function…
Rock

Paper

Scissors
But first, Let’s write a short guessing game...
Top-Down Coding
Goal

• 2 players
• Human + Computer
• Computer play is random
• Human plays via keyboard input
• Go for 10 rounds and announce winner
• Human may make mistakes
We're going to need the random module/library

we're going to need a loop (while, for?)

counters for the two players

if statements to test the two plays
Plan of Attack

• Just concentrate on an endless loop that gets the two players pick.

• Make the computer always pick the same letter ==> simpler to test this way

• Rely on a function to figure out who wins. In a first approach, make it always return the same thing: Again, it makes testing easier this way...
Let's code!
We stopped here last time...
Midterm Exam

Histogram of Midterm Grades

D. Thiebaut, Computer Science, Smith College
Studying Skills for Intro. to Programming

• Practice! Practice! Practice!

• Read the chapters. Challenge your knowledge with the self-test questions and exercises

• When the lab/hw solutions become available, compare your programs to the solutions. Update your programs if necessary.

• Review the programs we do in class. Try to rewrite them on your own.

• Verify that you can write the program from scratch on your own.

• 20-minutes a day…
Python is a foreign language
Review Lab 8
Problems
• getPositive()
• get Yes/No answer
• reading file until some key word
• converging quantity
Man in Hole Stories
An Introduction by Kurt Vonnegut (Visiting Prof. at Smith College, 2001)

https://www.youtube.com/watch?v=oP3c1h8v2ZQ
• Motherboard article: "There are only six basic book plots, according to computers," 3 Feb 2015

• Paris Review article: "Man in Hole," 4 Feb 2015

• Prof. Matthew Jokers, U. Nebraska, English Dept., released software tool on github to generate book plots
Neutralizing the last third of *Portrait*

![Graph showing emotional valence over narrative time]

Matthew Jockers
taken from http://www.matthewjockers.net/
Can we do something like this with Python?