CSCI 141
Computer Programming I

Filip Jagodzinski
Midterm Exam

- 12 February (Friday)
- In-class, 50 minutes
- Closed Notes
- Closed Book

- A sample midterm exam has been posted to the course website
- February 10 (Wednesday) we’ll have an in-class midterm review

(We still have a few weeks before the midterm, so a few topics on the sample midterm we haven’t yet covered.)
From Last Time

```python
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")
```

In addition to the unary and binary selection, we’ve learned about nested selections and chained conditional

Task: be able to draw the flow diagram for this code
From Last Time

```python
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")
```

In addition to the unary and binary selection, we’ve learned about nested selections and chained conditional

Task: be able to draw the flow diagram for this code

A. Count number of selection (if statements)
B. Count number of statements of code
C. Draw shapes
D. Designate Start
E. Designate End
F. Draw arrows, labeled where necessary
A flow diagram specifies **which** selection and code statements are executed, and their possible **order** of execution.
numCows = 6
if (numCows == 1):
    print("solo cow")
elif (numCows > 2):
    if (numCows == 3):
        print("cow * 3")
    else:
        print("cow quorum")
elif (numCows == 6):
    print("lots of milk")

Q: What is the output of the above program?

A. solo cow
B. cow * 3
C. cow quorum
D. lots of milk
Warmup

```python
numCows = 6
if (numCows == 1):
    print("solo cow")
elif (numCows > 2):
    if (numCows == 3):
        print("cow * 3")
    else:
        print("cow quorum")
elif (numCows == 6):
    print("lots of milk")
```

Q: What is the output of the above program?

A. solo cow  
B. cow * 3  
C. cow quorum  
D. lots of milk

Be sure you understand why ...

---

CSCI 141  
Computer Programming I
numCows = 6
if (numCows == 1):
    print("solo cow")
elif (numCows > 2):
    if (numCows == 3):
        print("cow * 3")
    else:
        print("cow quorum")
elif (numCows == 6):
    print("lots of milk")

Even though there is an `elif` that evaluates to True, a previous `elif` already evaluated to True, therefore only the code block in the first True `elif` is executed.

Q: What is the output of the above program?

A. solo cow
B. cow * 3
C. cow quorum  
D. lots of milk

Be sure you understand why …
For loops
The for loop

Motivation: Computers (via computer programs) are reaaaalyy good at monotonous tasks. If asked to do the following:

For a starting bank Account balance of 100.00, and assuming that each year the account earns 10% interest that is added to the account’s value, what is the account’s value after five years?

Q: How would you do this “on paper”?
Motivation: Computers (via computer programs) are reaaaaly good at monotonous tasks. If asked to do the following:

For a starting bank Account balance of 100.00, and assuming that each year the account earns 10% interest that is added to the account’s value, what is the account’s value after five years?

But because you are learning python, you could write code...
The for loop

Motivation: Computers (via computer programs) are reaaaaaly good at monotonous tasks. If asked to do the following:

For a starting bank Account balance of 100.00, and assuming that each year the account earns 10% interest that is added to the account’s value, what is the account’s value after five years?

But because you are learning python, you could write code ...

```python
balance = 100.00
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
```

Which when executed ...
The for loop

Motivation: Computers (via computer programs) are reaaaaaly good at monotonous tasks. If asked to do the following:

For a starting bank Account balance of 100.00, and assuming that each year the account earns 10% interest that is added to the account’s value, what is the account’s value after five years?

But because you are learning python, you could write code ...

```python
balance = 100.00
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)

>>> 110.0
121.0
133.1
146.41
161.051
```
Motivation: Computers (via computer programs) are really good at monotonous tasks. If asked to do the following:

For a starting bank account balance of 100.00, and assuming that each year the account earns 10% interest that is added to the account’s value, what is the account’s value after 500 years?

But because you are learning python, you could write code...

```python
balance = 100.00
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
```

Easy! .. For THIS task, but what about ...

I hope you are good at cutting and pasting!

We want an easy way to easily do a task over and over again.
The for loop

Motivation: Computers (via computer programs) are reaaaaly good at monotonous tasks. If asked to do the following:

Several observations

The same code is repeated over and over again ... We want to be smarter ... We don’t want to be cut-and-paste monkeys!

But because you are learning python, you could write code ...

```
balance = 100.00
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
balance = balance + (0.10 * balance)
print(balance)
```
The for loop

```python
for x in range(0, 3):
    print("The value of x is", x)
```

Learning by example
The for loop

```python
for x in range(0, 3):
    print("The value of x is", x)
```

The keyword `for` specifies special functionality.

Q: Therefore can you have a variable name that is named `for`?
The for loop

The variable \( x \) is the **iterator** variable

Think of the iterator as a variable that is assigned a new value for EACH iteration of the body of the for loop

```python
for x in range(0, 3):
    print("The value of x is", x)
```
The for loop

```python
for x in range(0, 3):
    print("The value of x is", x)
```

The keyword in specifies that x can take on values ...
The for loop

```python
for x in range(0, 3):
    print("The value of x is", x)
```

... output/generated by the function `range`, which generates a range of numbers starting from 0 and going up to but not including 3.
The for loop

```python
for x in range(0, 3):
    print("The value of x is", x)
```

You’ve seen this before. The colon specifies that what is to follow is the code block for “this” for loop.
The for loop

```python
for x in range(0, 3):
    print("The value of x is", x)
```

This is the code block for this for loop. You’ve used the print function already ... you know what it does.

The code block is executed for each iteration of the for loop.

Q: How many iterations are there of this for loop?
The for loop

This is the code block for this for loop.
You’ve used the print function already ... you know what it does

The code block is executed for each iteration of the for loop

Q: How many iterations are there of this for loop?

The range function generates the list 0, 1, and 2, so there are three separate values for x, and the for loop will iterate three times
The for loop

```
for x in range(0, 3):
    print("The value of x is", x)
```

Q: How is this for loop executed?
The for loop

```python
for x in range(0, 3):
    print ("The value of x is", x)
```

Execution steps

1. range generates the “list” of numbers 0, 1, and 2

Just like `print` is a function that you’ve seen receive as input one or more “inputs”, `range` is a function that in this case is being provided 2 inputs.
The for loop

```python
for x in range(0, 3):
    print("The value of x is", x)
```

Execution steps

1. range generates the “list” of numbers 0, 1, and 2

Unlike `print`, the `range` function “returns” something explicitly. We’ll talk about functions in coming weeks. For the time being, think of the return of range as its “output.” If you think about the black box analogy, then ...

```
print("Hello World")  ────  print  ────  Hello World  (output is sent to “screen”)
```
The for loop

for x in \texttt{range}(0, 3):
  \texttt{print} (\texttt{"The value of x is"}, x)

Execution steps

1. \texttt{range} generates the “list” of numbers 0, 1, and 2

Unlike \texttt{print}, the \texttt{range} function “returns” something explicitly. We’ll talk about functions in coming weeks. For the time being, think of the return of \texttt{range} as its “output.” If you think about the black box analogy, then ...

\begin{align*}
\texttt{print("Hello World")} & \quad \rightarrow \quad \texttt{print} \quad \rightarrow \quad \texttt{Hello World} \quad \text{(output is sent to “screen”)} \\
\texttt{range}(0, 3) & \quad \rightarrow \quad \texttt{range} \quad \rightarrow \quad 0 \ 1 \ 2 \quad \text{(output is sent to “program”)}
\end{align*}
The for loop

```
for x in range(0, 3):
    print("The value of x is", x)
```

**Execution steps**

1. `range` generates the “list” of numbers 0, 1, and 2
2. `x` is assigned the value 0

The combination of the keywords `for` and `in` causes the iterator variable `x` to assume the list of numbers generated by `range`, one-by-one
The for loop

```python
for x in range(0, 3):
    print("The value of x is", x)
```

**Execution steps**

1. range generates the “list” of numbers 0, 1, and 2
2. x is assigned the value 0
3. the print statement is executed

The value of x is 0
The for loop

Execution steps

1. range generates the “list” of numbers 0, 1, and 2
2. x is assigned the value 0
3. the print statement is executed

When the print statement finishes, we say that the first iteration of the for loop has completed

Q: What happens next?
The for loop

```
for x in range(0, 3):
    print("The value of x is", x)
```

**Execution steps**

1. `range` generates the “list” of numbers 0, 1, and 2
2. `x` is assigned the value 0
3. the print statement is executed **The value of x is 0**
4. `x` is assigned the value 1
The for loop

```
for x in range(0, 3):
    print("The value of x is", x)
```

**Execution steps**

1. range generates the “list” of numbers 0, 1, and 2
2. x is assigned the value 0
3. the print statement is executed
   
   ```
   The value of x is 0
   ```
4. x is assigned the value 1
5. the print statement is executed
   
   ```
   The value of x is 1
   ```

When the print statement finishes, we say that the second iteration of the for loop has completed.
The for loop

```python
for x in range(0, 3):
    print("The value of x is", x)
```

Execution steps

1. range generates the “list” of numbers 0, 1, and 2
2. `x` is assigned the value 0
3. the print statement is executed
4. `x` is assigned the value 1
5. the print statement is executed
6. `x` is assigned the value 2
The for loop

```
for x in range(0, 3):
    print ("The value of x is", x)
```

**Execution steps**

1. range generates the "list" of numbers 0, 1, and 2
2. x is assigned the value 0
3. the print statement is executed
4. x is assigned the value 1
5. the print statement is executed
6. x is assigned the value 2
7. the print statement is executed

**Q: What happens next?**
The for loop

```python
for x in range(0, 3):
    print("The value of x is", x)
```

Execution steps

1. range generates the “list” of numbers 0, 1, and 2
2. `x` is assigned the value 0
3. the print statement is executed
   - The value of x is 0
4. `x` is assigned the value 1
5. the print statement is executed
   - The value of x is 1
6. `x` is assigned the value 2
7. the print statement is executed
   - The value of x is 2
8. the for loop terminates
On-the-fly question

Task 1 : write a python program that prints to the screen `heya` 6 times, each time on a separate line. Sample output shown right.

In-class coding demonstration
On-the-fly question

Task 1: write a python program that prints to the screen `heya` 6 times, each time on a separate line. Sample output shown right.

In-class coding demonstration

Task 2: Complete the python program shown on the right so that it outputs the following:

Do not create a new String

In-class coding demonstration
Task 3: Write a python program that prints to the screen the positive multiples of 389 up to 3000.

The first three multiples of 389 are 389, 778, and 1167

Q: How many ways can this be done?

In-class coding demonstration
Task 3: Write a Python program that prints to the screen the positive multiples of 389 up to 3000.

The first three multiples of 389 are 389, 778, and 1167.

Q: How many ways can this be done?

In-class coding demonstration
The for loop

Back to the original task ...

For a starting bank Account balance of 100.00, and assuming that each year the account earns 10% interest that is added to the account’s value, what is the account’s value after 500 years?

( Coding demonstration )
The for loop

Sample solution

```python
balance = 100.00
for x in range(0, 500):
    balance = balance + (0.10 * balance)

print(balance)
```

Q: Where is the code block for the for loop?
Q: How many times will the balance be printed?
Q: Why are we using the range 0 through 499?
The for loop

Sample solution

```
balance = 100.00
for x in range(0, 500):
    balance = balance + (0.10 * balance)
print(balance)
```

Q: Where is the code block for the for loop?
Q: How many times will the balance be printed?
Q: Why are we using the range 0 through 499?

```
>>> 4.969841967312273e+22
```
Up next

Much more on for loops