

Loops in Python

CS 8: Introduction to Computer Science, Winter 2019
Lecture #7

Ziad Matni, Ph.D.
Dept. of Computer Science, UCSB

Administrative

- My office hours: rescheduled today:
2:30 pm – 3:30 pm (just for today...)
- Hw04 – due next week on **MONDAY in class**
- Lab03 – due next week on **MONDAY by midnight**
- You can check old homework on GradeScope
- Midterm Exam #1 is next **Wednesday!!!**
 - **Study Guide will be ON OUR WEBSITE by tomorrow**

Midterm #1 Exam

- **Feb. 6th 9:30 AM – 10:45 AM**
- In **THIS** classroom (unless you are a DSP student)
- Come **10 MINUTES EARLY** as there is **pre-assigned seating**
- **CLOSED BOOK!** But you can bring **1 page of notes**
 - Single-side only, 8.5" x 11"
 - Hand-written *or* computer printed is OK!
 - Must turn it in *with the exam* when done
 - No calculators / cell phones / any type of computer
- Bring your **UCSB ID** with you. **NO EXCEPTIONS.**

Midterm #1 Exam

WHAT'S ON IT?!

- **Everything**
 - Review ALL lectures
 - Review ALL readings
 - Review ALL labs
 - Review ALL homework

Midterm #1 Exam

SAMPLE QUESTIONS?!?!!?!!?!

- Yes! See Study Guide on the class website!

Lecture Outline

- Loops

Class Exercise

Get together with 2 or 3 other people around you and answer this question.

You can use your notes from last time. You can use your computers:

- a) Write a short Python code that asks a user their age. Once you do that, decide whether to print out “**Your age is an even number!**” or “**Your age is an odd number!**” depending on their answer.

- b) Now modify your code so that it can detect if someone entered a number less than 1 as their age. If so, print out a rejection message and quit.
Challenge: do this twice: once by using the **and** operator and once *without* using **and** (using nested-if statements)

Class Exercise

```
age = int(input("How old are you? "))

if (age % 2 == 0):
    print("Your age is an even number!")

else:
    print("Your age is an odd number!")
```

Class Exercise

```
age = int(input("How old are you? "))

if (age % 2 == 0) and (age > 0):
    print("Your age is an even number!")
elif (age % 2 != 0) and (age > 0):
    print("Your age is an odd number!")
else:
    print("You have entered an illegal age!")
```

Class Exercise

```
age = int(input("How old are you? "))

if (age > 0):
    if (age % 2 == 0):
        print("Your age is an even number!")
    else:
        print("Your age is an odd number!")

else:
    print("You have entered an illegal age!")
```

Loops

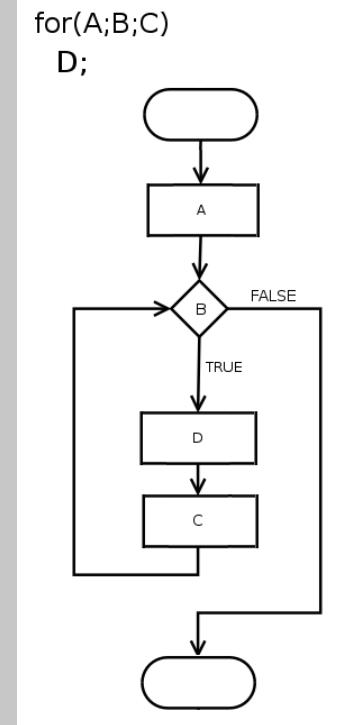
- Sometimes we want to be able to **repeat** a part of the program a certain number of times *without* being repetitive
 - Called a “loop”
- So instead of saying:

```
print("hello")
print("hello")
print("hello")
```

I can say:

do the following 3 times:

```
print("hello")
```
- A popular way to do this is with the **for** and the **while** commands.



Repetition with a `for` loop

- `for ref in some list:`
block of instructions – ref refers to current object in list
note that the block is all indented
 - `for, in, :` – mandatory parts
 - `ref` – a name for referring to objects in the list
- Example:
`for numbers in (0, 1, 2, 3, 4, 5):`
 `print (numbers)`
This will print out the numbers 1 thru 5 in sequence

Other Examples

```
for x in (9, 22, -77, 1):  
    y = x + 10  
    print (y)
```

WHAT DO YOU THINK THESE
LOOPS PRINT OUT?

```
for y in ("Hello", "Mother", "Hello", "Father"):  
    print (x, "!!")
```

```
n = 0  
for item in ["UCSB Location", (34.4140, -119.8489)]:  
    n = n + 1  
    print(n, item)
```

Using `range` with `for` loops

- The `range()` built-in function provides a handy list
- Simplest use: `range(n)`
 - Creates a list with `n` items `[0, 1, 2, ...n-1]`
- Example:

```
for numbers in range(6):  
    print (numbers)
```

This will print out the numbers 1 thru 5 in sequence
(just like the last example)

Other Examples

```
for x in range(7):  
    print (x)
```

WHAT DO YOU THINK THESE
LOOPS PRINT OUT?

```
for y in range(2, 9):  
    print (y - 2)
```

```
for item in range(5, -1, -1):  
    if item == 0:  
        print(item, "Blast off!!")  
    else:  
        print(item)
```

Repetition with a `while` loop

- `while condition`:
executes over and over until a condition is `False`
- Used for **indefinite iteration**
 - When it isn't possible to predict how many times a loop needs to execute, unlike with `for loops`
- We use `for` loops for **definite iteration**
(e.g., the loop executes exactly `n` times)

Repetition with a `while` loop

- `while condition`:
*# executes over and over until a condition is **False***
- While loops **won't run at all** if *condition* starts out as false
- While loops **run forever** if *condition* never becomes false
(i.e. if it always stays true)
- So care must be done in designing these sort of loops.

Applying while

Can be used for counter-controlled loops:

```
n = 500
counter = 0                      # (1) initialize
while counter < n:                # (2) check condition
    print(counter * counter)
    counter = counter + 1          # (3) change state
```

- But NOTE that this is a definite loop – easier to use a `for` loop:

```
for counter in range (500):      ...etc...
```

Applying while

This is a better application example – unlimited data entry:

```
AllGrades = 0          # (1) initialize
grade = int(input("enter grade or q to quit: "))
while grade != "q":    # (2) check condition
    AllGrades = AllGrades + grade      # process grade
    grade = int(input("enter grade or q to quit: ")) # ask again

# While loop has ended (no indents after here),
# now you can do other stuff...
print("Total grades is:", AllGrades)
print("You're all done now!")
```

YOUR TO-DOS

- Finish reading **Chapter 5**
 - We'll be discussing loops on Wednesday
- Start on **HW4** (**due next MONDAY**)
- Do **Lab3** (lab tomorrow ; turn it in by **Friday**)

- Don't bike angry!

</LECTURE>