Loops

(while and for)

CSE 1310 – Introduction to Computers and Programming
Book reference

• Book chapter for Repetition/Loop statements:
  – Chapter 2:
    • 2.1.4
    • 2.1.4, 2.1.6
    • 2.2.10 - 2.2.15
      – 2.2.13 – covers the range object. See the slides ‘for loop and range’ for details.
      – We did not cover the use of an ‘else’ clause for a ‘while’ or a ‘for’ loop.
Motivation

• Suppose we want to write a program that does this:
  – Ask the user to input an integer N.
  – Prints out all integers between 0 and N.

• The elements of Python that we have covered so far are not sufficient for writing this program.

• What is missing: the ability to repeat some instructions as many times as we want (N times).
Loops

• Loop statements implement repetition:
• There are 2 types of loops in Python:
  – `while` loops
    • The programmer specifies all the components that control the loop.
  – `for` loops
    • Iterate over a collection.
while loop

• Syntax:

```python
while boolean_expression:
    line 1
    line 2
    ...
    line n
```

• Notice the similarity in syntax with the if statement.
while loop

• Syntax:

```python
while boolean_expression:
    line 1
    line 2
    ...
    line n
```

- **header** of the while loop.
- **body** or the **suite** of the while loop.

• Notice the similarity in syntax with the `if` statement.
while loop execution

```python
while boolean_expression:
    line 1
    line 2
    ...
    line n
```

first line after loop

• Loop execution:
  – Step 1: evaluate `boolean_expression`.
  – Step 2: If `boolean_expression` is false, go to the first line after the loop.
  – Step 3: If `boolean_expression` is true, execute the body of the while loop, and go back to step 1 (evaluate the boolean expression).
**while loop with sentinel**

- Example: keep asking for user input until they enter 0.

```python
N = 1
while N != 0:
    N = int(input("enter an integer: "))
print("done with the while loop")
```
**while** loop with counter

- Example: print all the numbers between 0 and a value N entered by the user.

```python
N = int(input("enter an integer: "))

i = 0
while (i <= N):
    print(i)
    i = i+1

print("done with the while loop")
```
while loop with counter

- Example: print all the numbers between 0 and a value N entered by the user.

```python
N = int(input("enter an integer: "))

i = 0  # Variable i is called a counter. Initialize the counter.
while (i <= N):
    print(i)
    i = i+1  # Update the counter.

print("done with the while loop")
```
while loops: indentation matters

N = int(input("enter an integer: "))

i = 0
while (i <= N):
    print(i)
    i = i+1

print("done with the while loop")

What does this program do?
Designing a \texttt{while} loop

• When you design a \texttt{while} loop, you need to make sure that the loop will terminate exactly when needed (not too early or too late).
  – You need to define a test (boolean expression), that determines when to stay in the loop and when to exit.
  – You need to update variables within the body of the loop, as needed.
The **break** statement

- **break** forces termination of the current loop.
- Syntax:

```python
while condition:
    ...
    if condition:
        ...
        break  # after this line it moves to the next instruction after while
        ...
    ...
    ...
```
The `break` statement

- Example: print the first number $\geq N$ that is divisible by 13.
The **break** statement

- Example: print the first number >= N that is divisible by 13.

N = int(input("enter an integer: "))

i = N
while True:
    if (i % 13 == 0):
        print("first number >=", N, "divisible by 13 is ", i)
        break  # after this line it moves to the next instr after while
    i = i+1
The **continue** statement

- **continue** skips the rest of the body of the loop and goes back to the header of the loop (which will cause it to execute the next iteration, or to terminate).

- Syntax:

```python
...  
while condition:  
  ...  
  if condition:  
    ...  
    continue  
  ...  
...  
```
The **continue** statement

- Example: print numbers between 1 and N that are divisible by 13.
The *continue* statement

- Example: print numbers between 1 and N that are divisible by 13.

```python
N = int(input("enter an integer: "))
i = 0

while i < N:
    i = i+1
    if (i % 13 != 0):  # not divisible by 13,
        continue
    print(i, "is divisible by 13")
```
Practice: basic iteration

• Algorithm and problem solving
• Loop control: infinite loop and loop that never executes
• Create a program that iterates
  – as long as the user wants to continue (asked after each iteration)
  – 10 times
  – N times (N entered by the user)
• Generate numbers
  – 1, 2, ..., N
  – N, N-1, ..., 3, 2, 1
• Even numbers:
  – Generate and print even numbers
  – Generate all numbers, but print only the even ones
  – Generate all numbers, take different action based on a property
    • Draw the flow of control diagram. Be aware of the sequence of numbers generated by each branch.
• Print N stars
• Strings:
  – Print string letters
  – Print string letters with a certain property
Compute cumulative operations

• Algorithm, typical setup (counter, accumulator)
  – Separate the process of generating the values from the process of computing the cumulative operation.

• Practice: compute the following values:
  – Sum: $1 + 2 + \ldots + N$
  – $N!$: $1 \times 2 \times 3 \times 4 \times \ldots \times N$
  – sum of even and product of odd numbers.
Practice: break and continue

• Pb1: Repeat a certain operation until the user enters a specific value:
  – As long as user enters values greater or equal to 0: show letter grade based on score
  – If strictly negative: finish

• Pb2: Same letter grade problem, but stop only when user enters -1. Skip scores out of range (maybe print a message), print letter grade for scores in range.
  – Variation: in addition to printing the letter grade, also concatenate it into a string
Deeper understanding

• Do you see what the *general problem* corresponding to each of the specific examples above?

• What general problems have we seen?
Problem type

• Check if a collection (of values/objects) has a certain property.

• Example:
  – The user enters x and y, with x < y. Print True if any (at least one) number in that range (including x and y) is divisible by 13. Print False otherwise (note: this is the difficult part!).
    • Extra: if x > y, swap their values.
Nested loops

• One loop in the body of another loop

""" What does the following program print? How many times does each line execute?"""

```python
i = 1
while i < 4:  # outer loop
    print("i =", i)
    k = 1
    while k < 6:  # inner loop
        print("    i =", i, " k =", k)
        k = k + 1
    i = i + 1
print("Bye")
```
More practice: combine problems

• Keep asking for user input (an integer) until the number they enter would be a leap year.
The **for** loop
for loop

• The *for* loop iterates over the elements of a collection.
• Syntax:

```python
for variable in collection:
    line 1
    line 2
    ...
    line n
```

• *collection* can be, among other things, a string, a range or a list. (We will cover lists later in the course.)
The for loop iterates over the elements of a collection.

Syntax:

```
for variable in collection:
  line 1
  line 2
  ...
  line n
```

- `collection` can be, among other things, a string, a range or a list. (We will cover lists later in the course.)
Example: `for` loop over a string

```python
for my_var in "Hello there!":
    print(my_var)
print("---")
```

- What will this code do?
- What will `my_var` be? (piece of syntax, data, type)
Example: for loop over a string

```python
for my_var in "Hello there!":
    print(my_var)
print("---")
```

• What will this code do?
• What will `my_var` be? (piece of syntax, data, type)
  – Variable
  – Each one of the characters in the string “Hello there!”.
  – `my_var` is a string, that has only one character (a different one at each iteration)
• What is an iteration?
Example 2: **for** loop with a string

```python
# count the number of vowels in text entered by the user.

text = input("enter some text: ")
vowel_counter = 0

for i in text:
    if (i in 'aeiouyAEIOUY'):
        vowel_counter = vowel_counter + 1

print("\nThe text contains", vowel_counter, "vowels.")
```