More about Loops and Strings

February 24, 2021

Administrative Notes

Homework #1 grades were pretty good - keep it going

Sample exam is on Blackboard

Academic Integrity Quiz

Test #1 next Wednesday

Topics for Today’s lecture

The join statement - strings

Nested loops

Iterators that are not integers

The opposite of split is join

Remember the split statement

- Takes a string, and returns a list
- Breaks on whatever character you specify; breaks on whitespace by default

“Join” is the opposite

- Input is a list
- Result is a string, with the elements of the lists combined into a string
- Elements separated by the character(s) you identify in the statement
- Syntax:

```
str = “sep”.join(l)
```

.join - Notes

- The separator “sep” can be any string you want
  - Blank spaces, commas, and periods are fairly common
- If you want no separator, specify an empty string and the list elements will be run together
- Note: this ONLY works if the list is made up of strings
  - If you have a list of ints, you cannot join them!!

Using .split and .join to edit a string

s = “alpha,beta,gamma,delta,sigma”

replace all of the commas with blank spaces - using split and join

Nested loops

```
for i in range(10):
        for j in range(5):
                print( i, “ * “, j, “ is “, i*j)
                Input (“hit Enter to continue”)
```

```
i = 0
while i < 10:
    j = 0
    while j < 5:
      print( i, “ * “, j, “ is “, i*j)
      input (“hit Enter to continue”)
      j += 1
    i += 1
```

Note how this works. The inner loop - where j is the iterator - goes through its entire sequence for each different value of i

- You can’t use the same variable as the iterator in both loops

Let’s play around with the formatting of that previous example

```
i = 0
while i < 10:
    j = 0
    while j < 5:
      print( i, “ * “, j, “ is “, i*j)
      input (“hit Enter to continue”)
      j += 1
    i += 1
```

```
i = 0
while i < 10:
    j = 0
    while j < 5:
      print( i, “ * “, j, “ is “, i*j, end = “ “)
      input (“hit Enter to continue”)
      j += 1
    print(“\n”)
    i += 1
```

For i loops - why must the range be an integer?

Can’t be a string

- Python has no way of knowing what’s between each string
- “Alabama”, “Alaska” - what comes in between them? Depends -
  - Mississippi, Arkansas, Kansas, Colorado, Utah, Idaho, Washington…

Floating point numbers?

- Roundoff error