**ECS10**

10/24

---

**Midterm**

- Mostly programs looked good.
- Scores should appear on myUCDavis soon.
- If you got 4 or less on the program, you need to work more on the skills in Chapters 1 and 2.
- There is another (no credit) practice program you should do this week on the Web page.
- I will be in Lab hours on Thurs (1-3 in 2020 SLB).

---

**Next Project**

- Draw stars!
- Start using graphics.
- New data type – list
- New kind of loop - for

---

**Loop on string**

```
strIn = "530-752-4767"
i = 0
strOut = ""
while i < len(strIn):
    char = strIn[i]
    if char != ":
        strOut = strOut+char
    i = i+1
Eliminates the dashes.
```

---

**Lists**

- A string is a sequence of characters, numbered starting from 0.
- A list is a sequence of anything, numbered starting from zero.
- Index into a list just like indexing into a string.

```
intList = [5,0,3,5,6,8,9,0,4,4,4]
intList[4] # This is the integer 6
```

---

**List of strings**

```
beastList = ["cow","sheep","duck"]
i = 0
while i < len(beastList):
    print "A",beastList[i],"is an animal"
    i = i+1
```

- Length function len() works just the same as it does on strings – len(beastList) has the value 3.
- Same structure as loop on strings.
Sequences

- Lists and strings are two kinds of sequences.
- The len() function works on sequences (not on floats, or ints, or Booleans).
- You can index sequences to get elements.
- There is another kind of sequence in Python, the tuple, which is very like a list (does fewer things, but does them faster). We will mostly ignore tuples in this course.

Very common structure

- You have a big list, and you want to do the same thing for every item in the list.
- Add up scores for all students.
- Draw all lines in star.
- Convert all strings to integers.
- ....

For loop

```python
beastList = ["cow","sheep","duck"]
for beast in beastList:
    print "A",beast, "is an animal"
```

- Shorter and sweeter, but exactly the same effect.
- Goes through items in list, starting with `beastList[0]`, then `beastList[1]`, `beastList[2]`....
- String variable `beast` takes on each of these values in turn
- No index variable!

Loop on string

```python
strIn = "530-752-4767"
strOut = ""
for char in strIn:
    if char != "-":
        strOut = strOut+char
```

- Same effect as previous version.
- Prettier, shorter.
- char takes on values 5, then 3, then 0, then -, ...

Loop on integers

```python
count = 1
while count < 11:
    print count
    count = count+1
```

- Shorter and sweeter, but exactly the same effect
- Standard way to do something a fixed number of times.
- `range(start,stop)` is a built-in function
- Produces the list `[1,2,...,9,10]`
**while loop**

**effective interest rate**

```
while loop

balance = 100.0
annualRate = 7.0
monthlyRate = annualRate/12.0
month = 0
while month < 12:
    print "month ",month
    balance = balance+monthlyRate/100.0*balance
    month = month+1
```

**for loop**

**effective interest rate**

```
for loop

balance = 100.0
annualRate = 7.0
monthlyRate = annualRate/12.0
for month in range (0,12):
    print "month ",month
    balance = balance+monthlyRate/100.0*balance
```

- Two lines shorter than while version….

### for vs while

- Anything you can do with a for loop, you could also do with a while.
- for loop will be a little shorter and tidier.
- While loops are much more versatile.
- For loops are only good if you know how many times they will run before you start (12 months, length of list,…)

### Example: doubling time

```
principal = 100.0
annualRate = 7.0
monthlyRate = annualRate/12.0
balance = principal
month = 0
while balance < 2* principal:
    balance = balance + balance*monthlyRate/100
    month = month+1
```

- Can’t do this with a for loop!