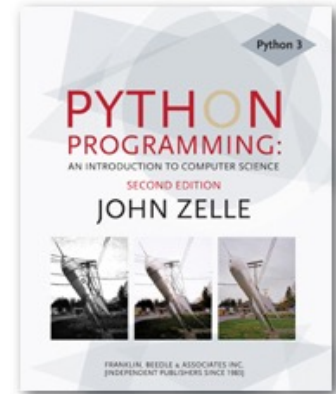




CS 177

---



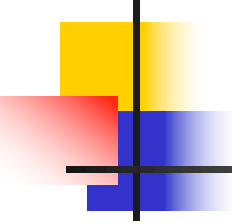
# List Comprehensions



# Lists, Lists, Lists... so many lists

---

- List Structures
- A List of Lists
- Loops and Lists
  - Iterations and Comparisons
- Building Lists
- List Comprehension



# Lists are useful, but only if we plan their structures carefully

---

It's important to consider not only what we're going to store in our List, but how it will be organized

```
fnames = ['Will', 'John', 'Yolanda', 'Zeb']
lnames = ['Carson', 'Wilhelm', 'Brown', 'Indiano']
for j in range(len(fnames)):
    print(lnames[j] + ', ' + fnames[j])
```



# Loops are useful for iteration, sequencing and comparisons

Creating a Lists can be done with for loops, while loops or ranges

```
list1=[]  
for j in range(1,100,2):  
    list1.append(j)
```

```
list3=list(range(1,100,2))
```

```
list2=[]  
j=1  
while j<100:  
    list2.append(j)  
    j+=2
```



# Practice: Use a loop to create the following:

---

1. A List with even integers from 50 to 200
2. A List with odd integers from 101 to 311
3. A List of UPPER case characters that are not vowels
4. A List of Lists containing all the UPPER case vowels and their corresponding ORD value  
ie: [ [ 'A' , 65 ] , [ 'E' , 69 ] ]



# List Comprehension

---

[ expression for variable in sequence if condition ]

- A *List comprehension* is a programming construct which is useful for creating a List based on another sequence
- A powerful and popular feature in Python  
Generates a new list by applying a function to every member of some other sequence



# The familiar components of *List* Comprehensions

---

[ expression for variable in sequence if condition ]

- The syntax looks like a **for** loop, an **in** operation, and an **if** statement
- All three of these keywords (**for**, **in**, and **if**) are also used in the syntax of forms of List comprehensions
- The **if** **condition** portion is optional



# List Comprehensions

---

[ expression for variable in sequence if condition ]

- Where expression is some calculation or operation acting upon the variable.
- For each member of the sequence that meets the condition:
  1. Set variable equal to that member,
  2. Calculate a new value using expression
  3. Add the new value to a List
- Finally, **return** the List





# List Comprehension Example

[ expression for variable in sequence ]

```
>>> [ x*2 for x in range(6,15,3) ]  
[12, 18, 24]
```

- The range function provides the values 6, 9 and 12
- [12, 18, and 24] are returned in the List



# Filtered List Comprehension

[ expression for variable in sequence if condition ]

```
>>> myList = [ 3, 6, 2, 7, 1, 9 ]
>>> [ elem*2 for elem in myList if elem > 4 ]
[ 12, 14, 18 ]
```

- Only 6, 7, and 9 satisfy the condition
- Only [12, 14, and 18] are returned as a result



# Practice: Use List comprehensions to create the following:

[ expression for variable in sequence if condition ]

1. A List with even integers from 50 to 200
2. A List with odd integers from 101 to 311
3. A List of UPPER case characters that are not vowels
4. A List of Lists containing all the UPPER case vowels and their corresponding ORD value  
ie: [ [ 'A', 65 ], [ 'E', 69 ]