For Loops in Python

For loops
Range
Breaks

For loops

Iteration
For loops allow us to iterate over each element of a set or sequence

Syntax:
for <var> in <set>:
do . . .
do . . .

Examples

▶ Iterating over a list:
words = ['got', 'me', 'looking', 'so',
    'crazy', 'right', 'now']
for w in words:
    print(w)

▶ Iterating over a string:
phrase = 'looking so crazy in love'
for w in phrase:
    print(w)

The range iterator

Iteration
Python has a built-in function, range, which allows us to iterate over the numbers specified in the range.

What do these do?
l = list(range(0, 10))
l = list(range(10))
l = list(range(1, 11, 2))
l = list(range(0, -10, -1))

Note: in Python 3, range is an iterator, so it generates numbers on the fly
▶ the list function compresses them into a list

Range in loops

for number in range(0, 10):
    print(number)

a = 'uh oh uh oh uh oh uh oh oh no no'
for number in range(len(a)):
    print(number, a[number])

for vs. while

a = 'uh oh uh oh uh oh uh oh no no'
i = 0
while i < len(a):
    print(i, a[i])
    i += 1

for i in range(len(a)):
    print(i, a[i])
Breaking out of loops

- **break**: stops the execution of the loop, independent of the test
  ```python
  for x in range(0,1000):
    if (x % 7) == 0:
      print('first number divisible by 7: ', x)
      break
  ```
- **continue**: skip the rest of the loop body, but continue with the loop
  ```python
  for x in range(0,1000):
    if (x % 7) == 0:
      print(x)
    else:
      continue
      print(x, "is divisible by 7")
  ```

List comprehensions again

- List comprehensions
  ```python
  sasha = [1,2,3,4,5]
  fierce = [x**2 for x in sasha]
  ```
- List comprehensions with conditions
  ```python
  fierce = [x**2 for x in sasha if x%2==0]
  ```

Unpacking a list comprehension

```python
fierce = []
for x in sasha:
  if x%2 == 0:
    fierce.append(x**2)
```

Loop else

**Iteration**
else can follow a loop, then it is executed in case the loop is NOT exited by break.

```python
for x in range(100):
  if x > 101:
    print(x)
  else:
    print("not found")
```

Only if the break condition is met will the else not run.