Repetition: the `for` loop

- Execution control structures
- `for` loop – iterating over a sequence
- `range` function
- Tracing code

Execution control structures

- Execution control (flow control) structures are statements that control which statements in a program are executed and in what order
  - An if/elif/else conditional statement specifies whether to execute or skip blocks of code
  - A for loop is an repetition (iteration) structure. For each item of a sequence, it assigns the item as the value of a control variable and then executes a block of code
- We are going to visualize the sequence of execution using pythontutor.com

`for` loop syntax

A **for** loop always has the following six elements:

```
for varName in sequence:
    codeBlock
```

1. It is introduced by the keyword ***`for`***
2. A control variable. You may choose any name for it. For example, *`i`* to go with a sequence of integers or *`day`* with a list of days of the week)
3. The keyword **`in`**
4. The name of a sequence (for example, a string, list or tuple)
5. A colon `(:)` (in Python, an indented block of code is always introduced by a colon)
6. An indented block of code

`for` loop execution

```
for varName in sequence:
    codeBlock
```

When a `for` loop is executed:

1. `varName` is assigned the value of the first element of sequence
2. the indented codeBlock is executed
3. Steps 1 and 2 are repeated for each element of sequence

For example:

```
word = "apple"
for letter in word:
    print(letter)
```

Instant exercise: type this example into pythontutor.com to visualize its execution. Then try the example using a variable name other than letter.

`for` loop – tuple example

```
for varName in aTuple:
    codeBlock
```

Example:

```
days = ('Mon','Tue','Wed','Thu','Fri','Sat','Sun')
for day in days:
    print(day)
```

Instant exercise: type this example into pythontutor.com to visualize its execution

`for` loop – list example

```
for varName in aList:
    codeBlock
```

Strings and tuples are immutable, but lists can be changed. What happens when codeBlock changes aList?

Example:

```
oneTwoThree = [1, 2, 3]
for i in oneTwoThree:
    oneTwoThree.append(i + 3)
    print('i = ',i)
    print('oneTwoThree = ', oneTwoThree)
```

Instant exercise: type this example into pythontutor.com to visualize its execution. Why is it a bad practice to modify the sequence you are iterating over?

Built-in function `range()`

A **for** loop is often used in combination with the built-in ***range*** function to execute a block of code a specified number of times.

The ***range*** function generates a sequence of integers.

- `range(n)` generates the sequence 0, …, n-1
- `range(i, n)` generates the sequence i, i+1, i+2, …, n-1
- `range(i, n, c)` generates the sequence i, i+c, i+2c, i+3c, …, n-1

Try each of these in pythontutor:

```
for i in range(5):
    print(i)
```

```
for i in range(3,5):
    print(i)
```

```
for i in range(1,6,2):
    print(i)
```

Exercise

Write for loops that will print the following sequences:

a) 0, 1, 2, 3, 4, 5, 6, 7, 8 , 9, 10
b) 1, 2, 3, 4, 5, 6, 7, 8, 9
c) 0, 2, 4, 6, 8
d) 1, 3, 5, 7, 9
e) 20, 30, 40, 50, 60