CSE 115
Introduction to Computer Science I
Road map

▶︎ Review ◀

range(stop) and range(start, stop, step)

for in Python

live coding / exercise
Arrays

An array is a collection of variables, with related names.

```javascript
var x = ['zero', 'one', 'two'];
```

The array consists of three variables, whose names are `x[0]`, `x[1]`, `x[2]`. The value of `x[0]` is 'zero', the value of `x[1]` is 'one', and the value of `x[2]` is 'two'.

`x.length` reflects the length of the array, which is 3.
<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

"zero"
"one"
"two"
Control flow
for statement

for ( statement\textsubscript{INIT} ; expression ; statement\textsubscript{UPDATE} ) {
    statement\textsubscript{0};
    statement\textsubscript{1};
    ...
    statement\textsubscript{N};
}

A loop in the flow of control!
Control flow
visualizing repetition (for)

function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}

Suppose we call sumTo(4)
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}
Control flow
visualizing repetition (for)

```javascript
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}
```

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>4</td>
</tr>
<tr>
<td>sum</td>
<td>0</td>
</tr>
<tr>
<td>i</td>
<td>1</td>
</tr>
</tbody>
</table>
function sumTo(max) {
  var sum = 0;
  for (var i=1; i<=max; i=i+1) {
    sum = sum + i;
  }
  return sum;
}
Control flow
visualizing repetition (for)

```javascript
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}
```

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>4</td>
</tr>
<tr>
<td>sum</td>
<td>1</td>
</tr>
<tr>
<td>i</td>
<td>1</td>
</tr>
</tbody>
</table>
Control flow
visualizing repetition (for)

function sumTo(max) {
  var sum = 0;
  for (var i=1; i<=max; i=i+1) {
    sum = sum + i;
  }
  return sum;
}

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>4</td>
</tr>
<tr>
<td>sum</td>
<td>1</td>
</tr>
<tr>
<td>i</td>
<td>2</td>
</tr>
</tbody>
</table>
Control flow
visualizing repetition (for)

function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>4</td>
</tr>
<tr>
<td>sum</td>
<td>1</td>
</tr>
<tr>
<td>i</td>
<td>2</td>
</tr>
</tbody>
</table>
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1) {
        sum = sum + i;
    }
    return sum;
}
Control flow
visualizing repetition (for)

```javascript
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}
```

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>4</td>
</tr>
<tr>
<td>sum</td>
<td>3</td>
</tr>
<tr>
<td>i</td>
<td>3</td>
</tr>
</tbody>
</table>

because $i \leq max$

The diagram illustrates the flow of the function `sumTo(max)` with a loop iterating from 1 to `max` inclusive, updating `sum` by adding `i` in each iteration, and finally returning the accumulated `sum`.
Control flow
visualizing repetition (for)

function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1) {
        sum = sum + i;
    }
    return sum;
}
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}
Control flow
visualizing repetition (for)

function sumTo(max) {
  var sum = 0;
  for (var i=1; i<=max; i=i+1) {
    sum = sum + i;
  }
  return sum;
}

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>4</td>
</tr>
<tr>
<td>sum</td>
<td>6</td>
</tr>
<tr>
<td>i</td>
<td>4</td>
</tr>
</tbody>
</table>
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1) {
        sum = sum + i;
    }
    return sum;
}
Control flow
visualizing repetition (for)

function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}
Road map

Review

- range(stop) and range(start, stop, step)

for in Python

live coding / exercise
**range**

The `range` type represents an immutable sequence of numbers and is commonly used for looping a specific number of times in `for` loops.

```python
class range(stop)
class range(start, stop[, step])
```

The arguments to the range constructor must be integers [...] If the step argument is omitted, it defaults to 1. If the start argument is omitted, it defaults to 0.

For a positive step, the contents of a range `r` are determined by the formula:

\[ r[i] = start + step \times i \text{ where } i \geq 0 \text{ and } r[i] < stop. \]

For a negative step, the contents of the range are still determined by the formula:

\[ r[i] = start + step \times i, \text{ but the constraints are } i \geq 0 \text{ and } r[i] > stop. \]

Lightly edited from https://docs.python.org/3/library/stdtypes.html#range
range examples

range(5) consists of the values 0, 1, 2, 3, and 4.

range(3, 7) consists of the values 3, 4, 5, and 6.

range(3, 7, 2) consists of the values 3 and 5.

Keep in mind: a range does not include its "stop" value.
Road map

Review

range(stop) and range(start, stop, step)

▶ for in Python ◀

live coding / exercise
for

https://docs.python.org/3/reference/compound_stmts.html#for

for var in sequence:
    statement_0
    statement_1
    ...
    statement_N
for
Python example

def sumTo(max):
    sum = 0
    for i in range(1, max + 1):
        sum = sum + i
    return sum

A range does not include its "stop" value.
range(1, 5) includes the values 1, 2, 3, and 4.
for
JavaScript/Python comparison

```javascript
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}
```

```python
def sumTo(max):
    sum = 0
    for i in range(1, max + 1):
        sum = sum + i
    return sum
```
for

https://docs.python.org/3/reference/compound_stmts.html#for

for var in sequence:
    statement₀
    statement₁
    ...
    statementₙ

A range is one of many kinds of sequences in Python.
sequences and for
Python example

```
def printSequence(seq):
    for x in seq:
        print(x)

printSequence("this is a string")
printSequence(range(5))
printSequence(['a', 'b', 'c'])
```

The following are sequences:
- range, as in `range(3, 8)`
- str, as in "this is a string"
- list, as in `['a', 'list', 'of', 'strings']`

N.B. A list is not an array, but is similar in some ways.
sequences and for
JavaScript/Python comparison

```javascript
function printSequence(seq) {
    for (var i = 0; i < seq.length; i = i + 1) {
        console.log(seq[i]);
    }
}

printSequence("this is a string");

// JS doesn't have a native range
printSequence(['a', 'b', 'c']);
```

```python
def printSequence(seq):
    for x in seq:
        print(x)

printSequence("this is a string")

printSequence(range(5))
printSequence(['a', 'b', 'c'])
```
JavaScript for (...in...)
i takes on index values, not contents, of sequence

```
function printSequence_1(seq) {
    for (var i in seq) {
        console.log(seq[i]);
    }
}

printSequence_1(['a', 'b', 'c']);
```

```
function printSequence_2(seq) {
    for (var i in seq) {
        console.log(i);
    }
}

printSequence_2(['a', 'b', 'c']);
```
Road map

Review

range(stop) and range(start, stop, step)

for in Python

▶ live coding / exercise ◀
exercise

Both Python and JavaScript allow the bracket notation to access elements of a string:

"abc"[0] has value "a"
"abc"[1] has value "b"

Define a function which accepts a string as an argument and which returns a new string consisting of every other character from the argument string, starting with the character at position 0.