CSE 115
Introduction to Computer Science I
Road map

▶︎ Review ◀

exercise from last time

code template / pattern

exercise

lists / arrays revisited

exercise
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.
for var in sequence:
    statement_0
    statement_1
    ...
    statement_N

A range is one of many kinds of sequences in Python.
Various JS loops
JavaScript/Python comparison

```javascript
function printSequence_0(seq) {
    for (var i = 0; i < seq.length; i = i + 1){
        console.log(seq[i]);
    }
}

printSequence_0(['a', 'b', 'c']);
```

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

printSequence(['a', 'b', 'c'])
```
Various JS loops

JavaScript/Python comparison

JS

for..in loops over indices

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

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

Python

for..in loops over elements

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

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

printSequence(['a', 'b', 'c']);
```
Various JS loops

JavaScript/Python comparison

for..in loops over indices

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

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

for..in loops over elements

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

printSequence(['a', 'b', 'c'])
```
Various JS loops

JavaScript/Python comparison

```javascript
function printSequence_3(seq) {
    for (var i of seq) {
        console.log(i);
    }
}
printSequence_3(['a', 'b', 'c']);
```

```python
def printSequence(seq):
    for x in seq:
        print(x)
printSequence(['a', 'b', 'c'])
```

for..of has similar behavior
Road map

Review

▶️ exercise from last time ◀️

code template / pattern

exercise

lists / arrays revisited

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.
Possible solutions

```javascript
function everyOther(s) {
    var result = "";
    for (var i = 0; i < s.length; i = i + 2){
        result = result + s[i];
    }
    return result;
}

console.log('"' + everyOther("abc") + '"');
console.log('"' + everyOther("abcd") + '"');
console.log('"' + everyOther("") + '"');
```

```python
def everyOther(s):
    result = ""
    for i in range(0,len(s),2):
        result = result + s[i]
    return result

def everyOther(s):
    result = ""
    for i in range(0,len(s),2):
        result = result + s[i]
    return result

print('"' + everyOther("abc") + '"')
print('"' + everyOther("abcd") + '"')
print('"' + everyOther("") + '"')
```
Road map

Review

exercise from last time

- code template / pattern

exercise

lists / arrays revisited

exercise
function everyOther(s) {
    var result = "";
    for (var i = 0; i < s.length; i = i + 2){
        result = result + s[i];
    }
    return result;
}

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

```javascript
function everyOther(s) {
    var result = "";
    for (var i = 0; i < s.length; i = i + 2){
        result = result + s[i];
    }
    return result;
}

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

An accumulator variable is initialized.
JS

function everyOther(s) {
    var result = ";
    for (var i = 0; i < s.length; i = i + 2){
        result = result + s[i];
    }
    return result;
}

JS

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

A loop of some kind executes repeatedly.
function sumTo(max) {
    var sum = 0;
    for (var i=1; i<=max; i=i+1){
        sum = sum + i;
    }
    return sum;
}

function everyOther(s) {
    var result = "";
    for (var i = 0; i < s.length; i = i + 2){
        result = result + s[i];
    }
    return result;
}
Common pattern

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

function everyOther(s) {
    var result = "";
    for (var i = 0; i < s.length; i = i + 2){
        result = result + s[i];
    }
    return result;
}
```

Final value in accumulator variable is returned.
Another example

function everyOther(s) {
    var result = "";
    for (var i = 0; i < s.length; i = i + 2){
        result = result + s[i];
    }
    return result;
}

function factorial(n) {
    var result = 1;
    for (var i = 1; i <= n ; i = i + 1){
        result = result * i;
    }
    return result;
}

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

Accumulator variable is initialized to the identity element of...
Another example

```javascript
function everyOther(s) {
    var result = "";
    for (var i = 0; i < s.length; i = i + 2){
        result = result + s[i];
    }
    return result;
}

function factorial(n) {
    var result = 1;
    for (var i = 1; i <=n ; i = i + 1){
        result = result * i;
    }
    return result;
}

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

Another example
Accumulator variable
is initialized to the
identity element of...

string concatenation: ""
multiplication: 1
addition: 0

the accumulation
operator of the loop.
```
Road map

Review

exercise from last time

code template / pattern

▶︎ exercise ◀

lists / arrays revisited

exercise
exercise 1

Write a function named 'implode' that takes an array (JS) or list (Python) of strings and returns a single string consisting of the individual characters of all of the argument strings, in order.

For example, `implode(["a", "b", "c"])` must return "abc".

Come up with additional test cases.

What is the accumulation operation?

What is the identity element of that operation?
Possible solutions

test cases:
implode(['a','b','c']) => 'abc'
implode(['a']) => 'a'
implode([]) => ''

Operation: + (string concatenation)
Initial result value: "" (empty string)

```js
function implode(x) {
    var result = "";
    for (var v of x) {
        result = result + v;
    }
    return result;
}
```

```python
def implode(x):
    result = ""
    for v in x:
        result = result + v
    return result
```
Road map

Review

exercise from last time

code template / pattern

exercise

▶ lists / arrays revisited ◀

exercise
some available operations

```javascript
var x = [];

x.push()
Adds one or more elements to the end of the array x and returns the new length of the array.

x.pop()
Removes the last element from the array x and returns that element.

x.length
Reflects the number of elements in the array x.
```

```python
x = []

x.append(y)
Add an item to the end of the list x.

x.pop[i]
Remove the item at the given position in the list x, and return it. If no index is specified, a.pop() removes and returns the last item in the list.

len(x)
Returns the number of elements in the list x.
```
Road map

Review

exercise from last time

code template / pattern

exercise

lists / arrays revisited

▶︎ exercise ◀︎
exercise 2

Write a function named 'explode' that takes a string and returns an array (JS) or list (Python) consisting of the individual characters of the argument string.

For example, explode("abc") must return ["a", "b", "c"].

Come up with additional test cases.

What is the accumulation operation?

What is the initial value for the accumulation variable?

[CURVEBALL: in this case it's not the identity of the operation]
Possible solutions

test cases:
explode('abc') => ['a','b','c']
explode('a') => ['a']
explode('') => []

explode(implode(['a','b','c'])) => ['a','b','c']
implode(explode('abc')) => 'abc'

Operation: push (JS) or append (Python)
Initial result value: [] (empty array/list)

JS

```javascript
function explode(str) {
    var x = [];
    for (var v of str) {
        x.push(v);
    }
    return x;
}
```

Python

```python
def explode(str):
    x = []
    for v in str:
        x.append(v)
    return x
```
fun application

Open the JavaScript console in your browser.

for (var elem of document.all) {
    elem.style.fontFamily = "courier";
}