CSE 503
Introduction to Computer Science for Non-Majors

Dr. Eric Mikida
epmikida@buffalo.edu
208 Capen Hall

Day 11
Finding Common Patterns

Announcements

- Lab #1 due Monday @ 11:59PM
- Lab #2 will be released Monday

Exercises from Last Time

1. Write a function, **`sumTo`**, that sums all numbers up to (and including) a given number. ie: **`sumTo(3)`** would sum 1, 2, and 3.
2. Write a function, **`sumSquaresTo`**, that sums all perfect squares up to a given number. ie: **`sumSquaresTo(19)`** would sum 1, 4, 9, and 16.
3. Write a function, **`countChars`**, that takes a string and a character, and counts how many times the character appears in that string. ie: **`countChars("Hello World!", "l")`** would return 3.

A Few Minor Details

JavaScript

```
function printSequence(seq) {
  for(let x of seq) {
    console.log(x);
  }
}
```

The **`for`**…**`of`** loop in JavaScript is used to iterate over elements of a sequence.
(Just like **`for`**…**`in`** *in Python)*

A Few Minor Details

JavaScript

```
function printSequence(seq) {
  for(let x of seq) {
    console.log(x);
  }
}
```

The **`for`**…**`of`** loop in JavaScript is used to iterate over elements of a sequence.
(Just like **`for`**…**`in`** *in Python)*

JavaScript also allows you to iterate over the ***indices*** of a sequence using **`for`**…**`in`**.
(Yes…it's a bit confusing)

A Few Minor Details

JavaScript

```
function printSequence(seq) {
  for(let x of seq) {
    console.log(x);
  }
}
```

The **`for`**…**`of`** loop in JavaScript is used to iterate over elements of a sequence.
(Just like **`for`**…**`in`** *in Python)*

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

JavaScript also allows you to iterate over the ***indices*** of a sequence using **`for`**…**`in`**.
(Yes…it's a bit confusing)

A Few Minor Details

JavaScript

These two functions do the same thing

```
function printSequence(seq) {
  for(let x of seq) {
    console.log(x);
  }
}
```

The **`for`**…**`of`** loop in JavaScript is used to iterate over elements of a sequence.
(Just like **`for`**…**`in`** *in Python)*

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

JavaScript also allows you to iterate over the ***indices*** of a sequence using **`for`**…**`in`**.
(Yes…it's a bit confusing)

Python	Construct/Concept	JavaScript
bool int float str range	Types	Boolean Number String
True False 1 2 17 3.5 4.99 "Hello", 'Goodbye'	Literals	true false 1 2 17 3.5 4.99 "Hello" 'Goodbye'
+ - * / // % < <= > >= == != and or not	Operators	+ - * / % ** < <= > >= == != and or not

Must be assigned before use. ie:
x = 10
y = x * 2
	Variables	Must be declared before use. ie:
		let x;
		x = 10;
		let y = x * 2;

True False def return if elif	Keywords	true false let function return

A Common Pattern

How would we write a function that takes a string, and returns a string with every other letter?

A Common Pattern

How would we write a function that takes a string, and returns a string with every other letter?

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

A Common Pattern

How would we write a function that takes a string, and returns a string with every other letter?

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

This function looks a bit familiar…

A Common Pattern

How would we write a function that takes a string, and returns a string with every other letter?

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

This function looks a bit familiar…

```
function sumTo(n) {
  let s = 0;
  for(let i=0;i<=n;i=i+1) {
    s = s + i;
  }
  return s;
}
```

It turns out it is a pretty common pattern.

A Common Pattern

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

function factorial(n) {
  let f = 1;
  for(let i=1;i<=n;i=i+1) {
    f = f * n;
  }
  return f;
}
```

```
function sumTo(n) {
  let s = 0;
  for(let i=0;i<=n;i=i+1) {
    s = s + i;
  }
  return s;
}
```

A Common Pattern

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

```
function factorial(n) {
  let f = 1;
  for(let i=1;i<=n;i=i+1) {
    f = f * n;
  }
  return f;
}
```

```
function sumTo(n) {
  let s = 0;
  for(let i=0;i<=n;i=i+1) {
    s = s + i;
  }
  return s;
}
```

1. Declare a variable to accumulate into
   a. *(assigned the identity of your operation)*

A Common Pattern

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

```
function sumTo(n) {
  let s = 0;
  for(let i=0;i<=n;i=i+1) {
    s = s + i;
  }
  return s;
}
```

```
function factorial(n) {
  let f = 1;
  for(let i=1;i<=n;i=i+1) {
    f = f * n;
  }
  return f;
}
```

1. Declare a variable to accumulate into
   a. *(assigned the identity of your operation)*
2. Loop some number of iterations

A Common Pattern

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

```
function factorial(n) {
  let f = 1;
  for(let i=1;i<=n;i=i+1) {
    f = f * n;
  }
  return f;
}
```

```
function sumTo(n) {
  let s = 0;
  for(let i=0;i<=n;i=i+1) {
    s = s + i;
  }
  return s;
}
```

1. Declare a variable to accumulate into
   a. *(assigned the identity of your operation)*
2. Loop some number of iterations
3. Accumulate into your declared variable

A Common Pattern

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

```
function factorial(n) {
  let f = 1;
  for(let i=1;i<=n;i=i+1) {
    f = f * n;
  }
  return f;
}
```

```
function sumTo(n) {
  let s = 0;
  for(let i=0;i<=n;i=i+1) {
    s = s + i;
  }
  return s;
}
```

1. Declare a variable to accumulate into
   a. *(assigned the identity of your operation)*
2. Loop some number of iterations
3. Accumulate into your declared variable
4. After the loop you have the accumulated value

More on Arrays and Lists

Lists (Python)

```
x = []
```

`x.append(y)`
Adds an item, y, to the end of list x.

`x.pop([i])`
Removes the item from position i of list x and returns it. If i is not given, it removes and returns the last item.

`len(x)`
Returns the length of list x.

Arrays (JavaScript)

```
let x = [];
```

`x.push(y)`
Adds an item, y, to the end of array x.

`x.pop()`
Removes and returns the last item in array x.

`x.length`
Returns the length of array x.

Exercise

Write a function named **`implode`** which takes an array (in JS) or a list (in Python) of strings and returns a single string consisting of the characters of of the argument strings in order.

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

What is the accumulation operation? What is the identity element?