First-Class Functions

First-Class Functions:

- "A programming language is said to have first-class functions if it treats functions as first-class citizens."

First-Class Citizen (Programming):

- " A first-class citizen (sometimes called first-class objects) in a programming language is an entity which supports all the operations generally available to other entities. These operations typically include being passed as an argument, returned from a function, and assigned to a variable."

Python Vs JavaScript

function
- passed on argument
- returned from a function
- assign a function to a variable

Python

```
def square(x):
    return x * x

f = square(5)

print(square)
print(f)
```

5 passed in into the square function

Returns ->> <function square at 0x11b4bd6a8>
25

If a function accepts other functions as arguments or returns functions as their result this what we call an higher-order function?

JavaScript

```
function square(x) {
  return x * x;
}

var f = square(5)

console.log(square)
console.log(f)
```

Returns - >> function square(x) {return x*x;}
25

Remove the (5)

Python

```
def square(x):
    return x * x

f = square

print(square)
print(f)
```

Assignment

* assigned a function to a variable *

f is equal to our square function

Returns -- > <function square at 0x10ddc87b8>
<function square at 0x10ddc87b8>

What it means to be a first-class function

* now we can treat f as a function *

JavaScript

```
function square(x) {
  return x * x;
}

var f = square

console.log(square)
console.log(f)
```

Returns -- > function square(x) {return x*x:}
function square(x) {return x*x:}

Python

```
def square(x):
    return x * x

f = square

print(square)
print(f(5))
```

Treating f as a function

Returns -- > <function square at 0x10ddc8840>
25

JavaScript

```
function square(x) {
  return x * x;
}

var f = square

console.log(square)
console.log(f(5))
```

Returns -- > function square(x) {return x*x:}
25

We can pass functions as arguments

★ Passing a function or an argument to another function. ★

Custom Build Map Function

Python

```
def square(x):
    return x * x


def my_map(func, arg_list):
    result = []

    for i in arg_list:
        result.append(func(i))

    return result

squares = my_map(square,[1,2,3,4,5])

print(squares)

def cube(x):
    return x * x * x
```

Results -- > [1, 4, 9, 16, 25]

arrray

run each item through the function

now we call this function!

Note: Where not adding the "()" to it it would try to execute the function

passing in square function

JavaScript

```
function square(x) {
   return x * x;
}

function my_map(func, arg_list){
  result = [];
  for (var i = 1; i <= arg_list.length; i++){
    result.push(func(i))
  }
  return result;

}
var squares = my_map(square, [1,2,3,4,5])

console.log(squares)

function cube(x) {
  return x * x * x;
}
```

Results -- > [1, 4, 9, 16, 25]

Pass the Cube Function in

Python:

```
def square(x):
    return x * x

def cube(x):
    return x * x * x


def my_map(func, arg_list):
    result = []

    for i in arg_list:
        result.append(func(i))

    return result

squares = my_map(cube,[1,2,3,4,5])

print(squares)
```

Returns -- > [1, 8, 27, 64, 125]

its useful to pass around functions

passed in this new function as an argument

JavaScript:

```
function square(x) {
   return x * x;
}

function my_map(func, arg_list){
  result = [];
  for (var i = 1; i <= arg_list.length; i++){
    result.push(func(i))
  }
  return result;

}
var squares = my_map(cube, [1,2,3,4,5])

console.log(squares)

function cube(x) {
  return x * x * x;
}
```

Returns -- > [1, 8, 27, 64, 125]

★ Return a function from another function. ★

- One of the aspects of what it mean to be a first-Class function.

Logger Example

Python

```
def logger(msg):

    def log_message():
        print('Log:', msg)

    return log_message

log_hi = logger('Hi!')
log_hi()
```

Set Variable → `log_hi = logger('Hi!')`

passed in a message

our function

our log-hi variable is now equal to log-message, that what got returned

Results --> Log: Hi !

★ Now we can run this log-hi variable just like its a function b/c its a function

★ We can treat log-hi variable just like log-message() function (it doesnt take in any arguments), and its going to execute log-message and print out our message ★

JavaScript

```
function logger(msg){
  function log_message(){
    console.log('Log: '+ msg)
  }
  return log_message
}

log_hi = logger('Hi !')

log_hi()
```

Results --> Log: Hi !

Why returning another function is useful:

Wrap Text Function

Python

```
def html_tag(tag):

    def wrap_text(msg):
        print('<{0}>{1}</{0}>'.format(tag, msg))

    return wrap_text

print_hi = html_tag('h1')
print_hi('Test Headline!')
print_hi('Another Headline!')

print_p = html_tag('p')
print_p('Test Paragraph!')
```

passing in

We can now use print_hi like a function

pass in a message to our print_hi variable

pass in another message

remembered the tag that we passed in before?

Results:

```
<hi>Test Headline!</hi>
<hi>Another Headline!</hi>
<p>Test Paragraph!</p>
```

JavaScript

```
function html_tag(tag) {
  function wrap_text(msg){
    console.log('<' + tag + '>' + msg + '</' + tag + '>')
  }
  return wrap_text

}

print_h1 = html_tag('h1')

print_h1('Test Headline!')

print_h1('Another Headline!')

print_p = html_tag('p')
print_p('Test Paragraph!')
```

Results:

```
<hi>Test Headline!</hi>
<hi>Another Headline!</hi>
<p>Test Paragraph!</p>
```