Quiz Review

April 9th 2018

teresa@cs50.net

About Quiz

- Released at noon on Sat 4/21
- Due by noon on Mon 4/23
- Lectures 0 through 11 (and Problem Sets 0 through 8)
- Emphasis on Lecture 7 (and Problem Set 6) onward
- Review sections on Mon 4/9, Sun 4/15, Mon 4/16, Tue 4/17, and Wed 4/18
- Office hours from Tue 4/10 through Fri 4/20

Python: Variables

- Python variables do not have explicit data types
- Python variables do not need to be declared in advance
- Python variables **do** have underlying data types:
  - Number
  - Stirng
  - List
  - Tuple dictionary

C

```
int a = 3;
char b = "c";
int arr[] = {1, 2, 3, 4};
char myString[] = "This is a string";
```

Python

```
a = 3
b = "c"
arr = [1, 2, 3, 4]
myString = "This is a string"
```

Python: Conditionals

- Conditions don't have to be inside parentheses, except for grouping.
- Instead of using curly braces, we use a colon and indentation.
- The conditional is terminated by returning to the previous indentation level.
- &&, || are replaced with and, or. ! is replaced with not.
- else if is shortened to elif

Python: Conditionals

C

```
if (a != 50)
{
    b = 1;
}
else if (b > 0)
{
    b = 0;
}
else
{
    b = -1;
}
```

Python

```
if a != 50:
    b = 1
elif b > 0:
    b = 0
else:
    b = -1
```

Python: Loops

- `for` and `while` are the two primary iterating constructs in Python.
- `for`, in particular, has extreme flexibility relative to its C cousin.
- `do-while` does not exist in Python and has to be hacked with a `while True:` and a `break`
- The code subject to a loop is introduced by `:` instead of `{`.
- All code subject to the loop **must** be indented in order things to work as intended.
- The loop is terminated by returning to the previous indentation level.

Python: Loops

C

```c
for (int i = 0; i < 50; i++)
{
    // Do something
}
```

python

```python
for i in range(50):
    # Do something
```

Or:

```python
for i in range(0, 50):
    # Do something
```

Or:

```python
for i in range(0, 50, 1):
    # Do something
```

Python: Loops

C

```c
int i = 0;
while(i < 100)
{
    printf("%i\n", ++i);
}

for(int j = 0; j < 100; j += 2)
{
    printf("%i\n", j);
}
```

Python

```python
i = 0
while i < 100:
    print(i)
    i += 1

for j in range(0, 101, 2):
    print(j)
```

Python: Loops

C

```
for (int i = 1337; i > 50; i -= 3)
{
    // Do something
}
```

python

```
for i in range(1337, 50, -3):
    # Do something
```

C

```
int vals[] = {4, 5, 6, 1, 2, 4, 2, 44, 5};
int sum = 0;
for (int i = 0; i < 9; i++)
{
    sum += vals[i];
}
```

python

```
vals = [4, 5, 6, 1, 2, 4, 2, 44, 5]
sum = 0
for num in vals:
    sum += num
```

Python: Lists

- Lists are the Python version of Arrays.
- They can be dynamically grown or shrunk.
- They can also contain variables of multiple types.

Python: Lists

- To initialize an array you use the square brackets:

  vals = [] - vals is an empty list

  vals = [1, 2, 3] - vals contains the ints 1, 2 and 3.

- To initialize an array containing the number 1 through 100:

  `vals = [for i in range(1, 101)]`

- To initialize an array of 1000 zeros:

  `vals = [0 for _ in range(0, 1000)]`

- A list can also be initialized using the list() function: vals = list(), will initialize vals as an empty list.

Python: Lists

Appending, inserting and concatenating

you can add a value to the end of the list using the append method.

```
a = [1, 2, 3]
a.append(5)
# a is now equal to [1, 2, 3, 5]
```

You can insert a value at a specific place in the list using the insert method. list.insert(i, x), will insert x before the i-th element in the list.

```
a = ["a", "b", "c", "d"]
a.insert(2, "derp")
# a is now ["a", "b", "derp", "c", "d"]
```

To stick one list to the end of another one, you can use the + operator:

```
a = [3, 4, 5]
b = [0, 1, 2]
c = a + b
# c is now [3, 4, 5, 0, 1, 2]
```

```
a.append(b)
# a is now [3, 4, 5, [0, 1, 2]]
```

Python: Printing

No more “\n”!

The print method automatically adds a newline at the end.

If you don’t want a new line:

```
print("a string", end="")
```

C

```
int a = 7;
char c = 'x';

printf("%d, %c\n", a, c);
```

python

```
a = 7
c = "x"

# Method 1
print(a, end = "")
print(", ", end = "")
print(c)

# Method 2
# Here we use the + operator to concatenate strings and the str function to convert other variables to strings.
print(str(a) + ", " + str(c))

# Method 3
print("{}, {}".format(a, c))

print(f “{a}, {c}”)
```

Python: Dictionaries

```
weather = {
    "England": "Rainy",
    "California": "Warm",
    "Florida": "Humid",
    "Estonia": "Cold"
}

# This will print out "humid"
print(weather["Florida"])

# This will change england's weather to windy
weather["England"] = "windy"
```

You can iterate over the **keys** of a dictionary using a simple for loop:

```
for place in weather:
    print(place)
```

This will print out all of the places from the dictionary from the previous example. If you also want the values stored in the dictionaries, you can use the following:

```
for place, status in weather.items():
    print("The weather in {} is {}".format(place, status))
```

Python: Functions

Function are declared using the def keyword.

C

```
int square(int a) {
    return a*a;
}
```

python

```
def square(a):
    return a**2
```

Note: In Python you can use the ** operator to exponentiate values.

In Python functions don't have prototypes, but do need to be defined before they are called

Python doesn't have a main function. You can simply start writing code in a .py file and it will get run. However if you still want a main function, then you can do the following:

```
def main():
    # do stuff

# this part is important to make sure main gets executed:
if __name__ == "__main__":
    main()
```

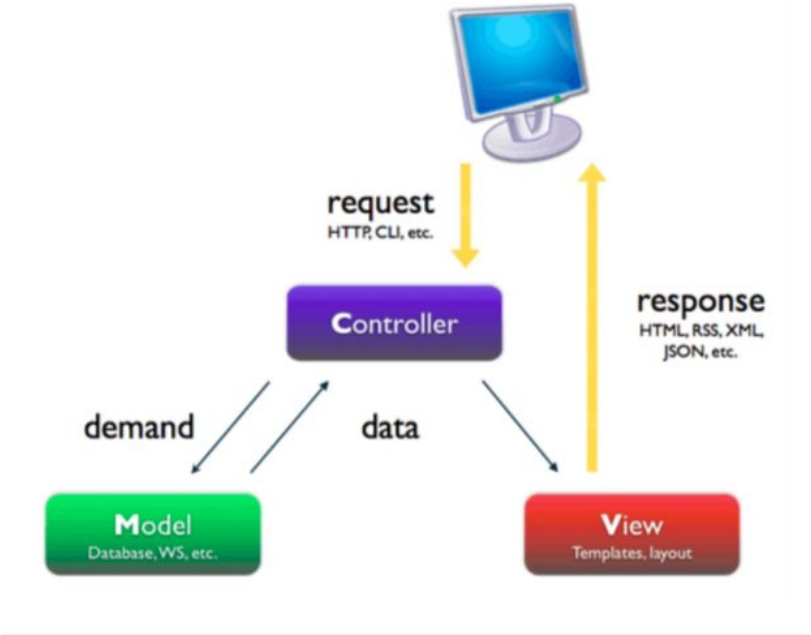
Python: Classes and Objects

- Objects are similar to structures in C in that they have fields. Additionally they have methods which are functions that are inherently part of that object, and may only be called directly by those objects.
- You define the methods and properties of an object inside of a class.
- Classes are created using the class keyword. Class names conventionally start with a capital letter.
- At a minimum, a class must contain a method called __init__, which sets the initial values of properties in the object.
- All methods of classes must include the self parameter as their first parameter, which is a reference to the object that is invoking the method.

Let’s look at an Example

`ex1_student`

Model-View-Controller



	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

SQL

- Structured Query Language
- A programming language which is used for managing and manipulating data in relational databases.
- In CS50, you will use a SQLite database.
- SQLlite databases are stored in a `.db` file.
- In order to insert, delete or modify the data in the database you will use SQL queries.

4 basic queries:

`UPDATE`

`INSERT INTO`

`SELECT`

`DELETE`

SQL: Creating a Table

- Create Fields
- Choose data type from options such as `INTEGER`, `REAL` (a floating-point number), `TEXT` (a string), `BLOB` (binary data), `NUMERIC` (numbers that can be either integers or floats), `BOOLEAN`, `DATETIME` (to store dates and times in a standard way).
- `Primary Key` indicates whether that field is the key that uniquely identifies all the rows in that table. But it's possible that two people share the same name and dorm, so we won't check that.
- `Autoincrement` allows us to have an integer field that increments itself every time a new row is added (like for an ID number), so we'll leave that unchecked too.
- `Not NULL` means that the field cannot be empty, or null. Since we want both fields to filled for every row, we'll check this for both.
- Specify some `Default Value` if no value is provided, but we won't use that either.

SQL: UPDATE

`UPDATE`: Update data in a database table

```
# update table, changing values in particular columns

UPDATE table SET col1 = :val1, col2 = :val2, ... )
```

```
# update table, changing col1 to val1 where "name" equals "identifier"

UPDATE table SET col1 = val1 WHERE identifier = "name"
```

SQL: INSERT INTO

`INSERT INTO`: Insert certain values into a table

```
 # insert into table the row of values
INSERT INTO table VALUES values
```

```
# insert into table under columns col1 & col2, val1 & val2
INSERT INTO table (col1, col2) VALUES (val1, val2)
```

SQL: SELECT

`SELECT`: Select values to view

```
# select a column from table to compare/ view

SELECT col FROM table WHERE col = "identifier"
```

```
# select all columns from a table

SELECT * FROM table
```

SQL: DELETE

`DELETE`: Delete from table

```
# delete a row from table

DELETE FROM table WHERE col = "identifier"
```

SQL

To execute a SQL query in Flask you can use the provided `db.execute()` method.

```
# query database for username

rows = db.execute("SELECT * FROM users WHERE username =
:username", username=request.form.get("username"))
```

To manipulate a database directly (this will come in handy when setting your database up) you will use phpliteadmin, which is a web-based interface.

To run phpliteadmin run the following command in the IDE, substituting `mydatabase.db` for the name of your database file:

```
$ phpliteadmin finance.db
Running phpLiteAdmin at https://ide50-teresa-lee.cs50.io:8081/?pin=qq7KjylNOzo9fQnh
Exit with ctrl-c...
```

Let’s look at an Example

`ex2_student`

JavaScript

A dynamic programming language used by web browsers on the ***client side*** that allows users to communicate ***asynchronously*** with the browser.

Client side means that the code is executed on the user's computer as opposed to on the server. This is much faster since you don't have to communicate with another device.

Asynchronously means that one piece of code doesn't necessarily need to wait until another finishes.

A few characteristic features of Javascript are:

- Just like with Python, you don't need to compile Javascript.
- It's loosely and dynamically typed. In order to declare a variable of any type you can just use the `let` keyword.
- Its ability to use the `DOM` (document object model) to dynamically change `HTML` is one reason why it is so widely used throughout the world wide web.

JavaScript: Basic Sytax

```
// a simple variable
let age = 20;

// an array
let array = [1, 2, 3, 4, 5];

// string
let str = "This is CS50!";

// an object
let teacher = {name: "David", course:
50};
```

JavaScript has ability to behave like an “object-oriented” language

An object is analogous to C structure or Python’s dictionary

Note that {} and ; are back!

JavaScript: Conditions and loops

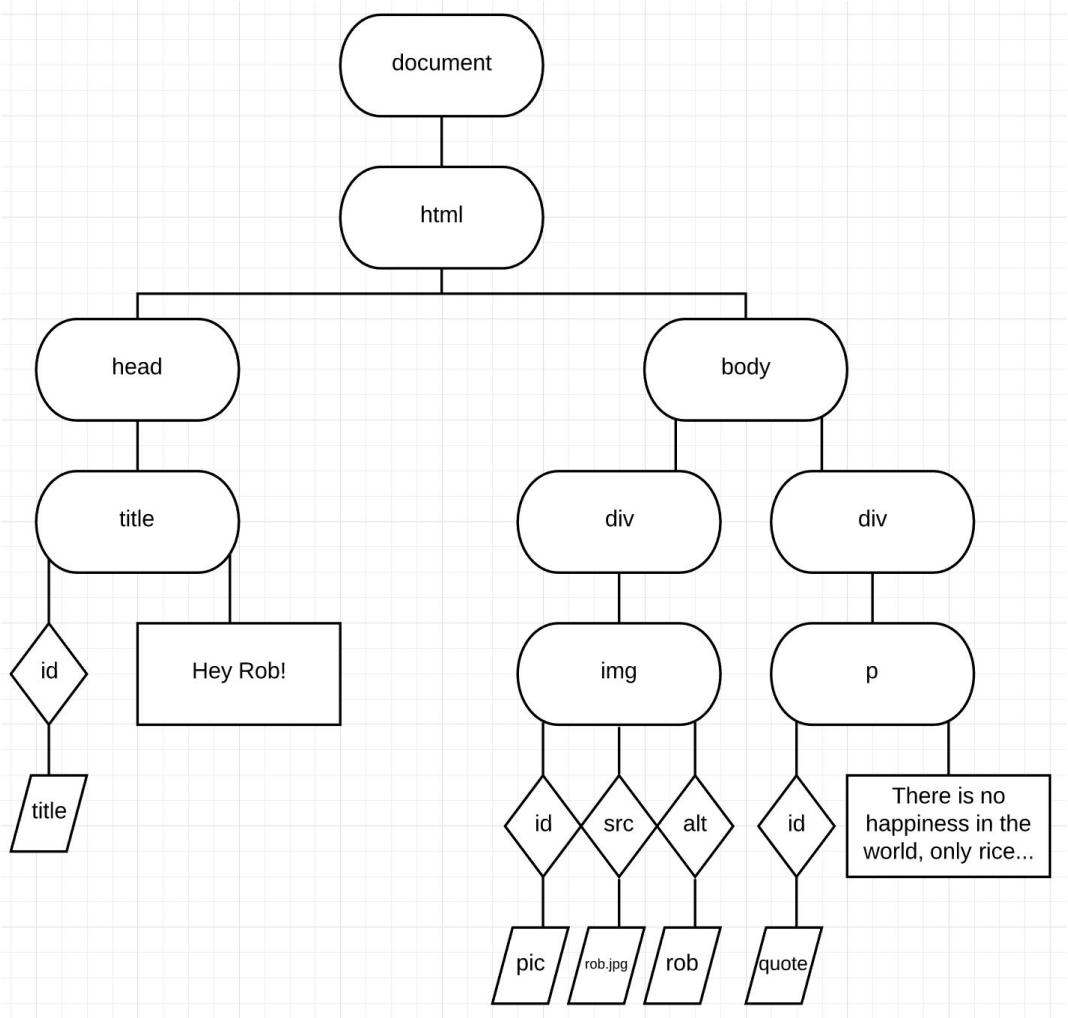
```
// if condition
if (true)
{
    // do something
}
```

```
// while loop
while (true)
{
    // do something
}
```

```
// for loop
for (initialization; condition;
update)
{
    // do something
}
```

Turn this into DOM

```
<!DOCTYPE html>
    <head>
        <title id="title">Hey Rob!</title>
    </head>
    <body>
        <div>
            <img id="pic" src="rob.jpg" alt="rob"/>
        </div>
        <div>
            <p id="quote">There is no happiness in the world, only rice...</p>
        </div>
    </body>
</html>
```


document
html
head
body
title
div
div
id
Hey Rob!
img
p
title
id
src
alt
id
There is no happiness in the world, only rice...
pic
rob.jpg
rob
quote

Change the DOM dynamically with JavaScript

```
<script>
    var title = document.getElementById("title");
    title.innerHTML = "David";

    var pic = document.getElementById("pic");
    pic.src = "david.jpg";

    var alt = document.getElementById("pic");
    alt.alt = "David";

    document.getElementById("quote").innerHTML = "allllright";
</script>
```

DOM: Properties

PROPERTY	DESCRIPTION
`innerHTML`	Holds the HTML inside a set of HTML tags.
`nodeName`	The name of an HTML element or element's attribute.
`id`	The "`id`" attribute of an HTML element
`parentNode`	A reference to the node one level up in the DOM.
`childNodes`	An array of references to the nodes one level down in the DOM.
`attributes`	An array of attributes of an HTML element.
`style`	An object encapsulating the CSS/HTML styling of an element.

DOM: Methods

METHOD	DESCRIPTION
`getElementById(id)`	Gets the element with a given ID below this point in the DOM.
`getElementsByTagName(tag)`	Gets all elements with the given tag below this point in the DOM.
`appendChild(node)`	Add the given node to the DOM below this point.
`removeChild(node)`	Remove the specified child node from the DOM.

JavaScript: Events

An event in HTML and JavaScript is a response to user interaction with the web page

JavaScript has support for event handlers, which are callback functions that respond to HTML events

Many HTML elements have support for events as an attribute

```
<html>
	<head>
		<title>Event Handlers</title>
	</head>
	<body>
		<button onclick=”alertName(event)”>Button1</button>
		<button onclick=”alertName(event)”>Button2</button>
	</body>
</html>

function alertName(event)
{
	let trigger = event.srcElement;
	alert(‘You clicked on ‘ + trigger.innerHTML);
}
```

JavaScript: Functions

Declare with `function` keyword

Anonymous functions don't need names. Particularly for those bound to HTML elements

Dom0

```
<html>
   <head>
      <script>
         function greet()
         {
            alert('hello, ' + document.getElementById('name').value + '!');
         }
      </script>
      <title>dom0</title>
   </head>
   <body>
      <form id="demo" onsubmit="greet(); return false;">
         <input id="name" placeholder="Name" type="text"/>
         <input type="submit"/>
      </form>
   </body>
</html>
```

Let’s Look at an Example

`ex3_students`

Last words

1. Review each lecture's **notes**.
2. Review each lecture's **source code**.
3. Review each lecture's **slides**.
4. Attend or watch the **review session**.
5. Review each lecture's **video**.
6. Take **last year's quiz**.
7. Review problem sets.

Good Luck!