CPSC 231 Tutorial #8

michael-hung.ca/teaching

Reminders

TOMORROW

Assignment #2 Paired Component

NEXT TUESDAY

Quiz #4 Review

Quiz #5

OCTOBER 25

Alberta Collegiate Programming Contest (ACPC) Signup Deadline

*Sign up for **Division 2** since you haven't taken CPSC 319/331*

Quiz 3 available for pickup.

Last time...

- Variable Scope (Local/Global)
- Function Parameters, Arguments, and Returns
- Gambler’s Ruin Simulation
- Debugging in VS Code

Python Lists

- Use square brackets to declare and get element indices:

```
my_array = [element_0, element_1, …, element_n]
print(my_array[0])
```

- Lists are mutable: you can change the elements without changing the object reference

```
my_array[1] = 12
```

- The elements of a list can be different data types **(not recommended though)**

```
my_array = [1, ‘two’, 3.0]
```

Python Lists

- You can put a list inside a list

```
my_array = [[1, 2], [3, 4]]
```

- Lists are iterable!

```
my_array = [1, 2, 3]
for i in my_array:
    # do something with i
```

- Be careful with **bounds**
- Check the length of an array with **len(**my_array**)**

List Functions

We can **append** or **remove** elements from lists

```
a = [1, 2, 3]
```

- Append

```
a.append(4)      # a = [1, 2, 3, 4]
```

- Remove

```
a.remove(a[0])   # a = [2, 3, 4]
a.remove(4)      # a = [2, 3]
```

List Functions

We can also **sort** or **reverse** lists

```
a = [2, 3, 1]
```

- Sorting

```
a.sort()      # a = [1, 2, 3]
```

- Reversing

```
a.reverse()   # a = [3, 2, 1]
```

Aliases

```
a = [1, 2, 3]
print(a)          # Outputs [1, 2, 3]

b = a
print(b)          # Also outputs [1, 2, 3]

b[1] = 5

print(b)          # Outputs [1, 5, 3]
print(a)          # Also outputs [1, 5, 3]!
```

Aliases

- b is **not** a new object, it is an “alias” of a
- Both b and a reference the same object
- Changing b will change a, and vice versa
- In general, this is true for all mutable, non-primitive data types in Python
- This is known as “Call By Object Reference”

Aliases

```
def a_function(an_array):
    if (len(an_array) >= 1):  # If there's at least one element
        an_array[0] = 1       # Change first element to 1

a = [4, 5, 6]

function1(a)
print(a)                      # Outputs [1, 5, 6]
```

Copies

To avoid aliasing, make **copies**.

```
a = [1, 2, 3]
b = a.copy()

b[1] = 5

print(b)      # Outputs [1, 5, 3]
print(a)      # Outputs [1, 2, 3]; it's unchanged!
```

Copies

Let's say we don't want to change the original array:

```
def a_function(an_array):
    temp = an_array.copy()
    temp[0] = 1
    return temp

a = [4, 5, 6]

b = a_function(a)
print(a)              # Outputs [4, 5, 6]
print(b)              # Outputs [1, 5, 6]
```

Designing Functions

Play a Game of Pig