Variables and Memory

- Variables are boxes in memory that hold values.
- Each box has a name.
- Python maintains a dictionary (keys that map to values) that maps names to the boxes in memory.
- The variable name *points* to the value in memory.
- Names do not have type, but the values they reference do.

Variables and Memory

- Assigning one variable name to another variable name establishes an alias. They both point to the same box in memory.

```
a  =  6
b  =  2
c  =  a
a  =  b
b  =  c
```

Variables and Memory

- Assigning one variable name to another variable name establishes an alias. They both point to the same box in memory.
- Subsequent assignments to variables break aliases and associates names with new boxes in memory.

```
a = 7
b = a
b = b + 3
```

Arrays

- Arrays are handled the same way.

```
a = [8,-6,0,3]
b = a
b = [6,-2,1,0]
```

Arrays

- Arrays are handled the same way.
- Important to note that changing the value of a cell in an array is not the same as change the value of the array itself. The referenced cell is modified, but the alias is not broken.

```
a = [4,15,-5,6]
b = a
b[1] = -2
```

Functions and Parameters

- Parameters to a function are variables in that function.
- When a function is called with values for parameters, each parameter is assigned the value implicitly.
- The same rules for aliases and subsequent assignments apply.

```
def set_value_at(arr,index,val):
  if index < 0 or index >=
len(arr):
    return
  b = [None] * len(arr)
  for i in range(len(arr)):
    b[i] = arr[i]
  b[index] = val
  arr = b

a = [4,15,-5,6]
set_value_at(a,1,-2)
```

```
def set_value_at(arr,index,val):
  if index < 0 or index >=
len(arr):
    return
  arr[index] = val


a = [4,15,-5,6]
set_value_at(a,1,-2)
```

Sorting an Array (Insertion Sort)

- Insertion sort is one of the easier sorting algorithms to understand.
- Starting with the second value in the array, we guarantee that the values from the beginning of the array to the current position are sorted.
- For each value, start at its original position *k* and make a copy of the value (*cur*).

Sorting an Array (Insertion Sort)

- As long as we are inside of the array and the value to *k*'s left is greater than *cur*, shift the value to *k*'s left right one spot and move left to the next possible position.
- When we are at position 0 or the value to *k*'s left is not greater than *cur*, we have found the correct location for *cur*. Place *cur* in that position, increment *k* and continue with the next candidate.

```
def insertion_sort(arr):
  for k in range(1, len(arr)):
    cur = arr[k]
    j = k
    while j > 0 and arr[j-1] >
cur:
      arr[j] = arr[j-1]
      j = j - 1
    arr[j] = cur

a = [17,8,-4,2,8,9]
insertion_sort(a)
```