Introduction to Python

Lists and Dictionaries

29th September 2016

Creating a list

- Lists are denoted by square brackets. Here is an empty list.

```
x = []
```

Creating a list

- Lists are denoted by square brackets. Here is an empty list.

```
x = []
```

- We can also initialise the list with some elements

```
x = [1,"hi",72]
```

Creating a list

- Lists are denoted by square brackets. Here is an empty list.

```
x = []
```

- We can also initialise the list with some elements

```
x = [1,"hi",72]
```

To access these we use the square bracket accessors:

```
print x[0] # prints 1
print x[1] # prints "hi"
```

Mutating lists

- Square brackets also let us change entries in a list.

```
x = [1,"hi",72]
x[0] = "foo"
print x # prints ["foo", "hi",72]
```

Mutating lists

- Square brackets also let us change entries in a list.

```
x = [1,"hi",72]
x[0] = "foo"
print x # prints ["foo", "hi",72]
```

- If we want to add something to a list we can use "append" or list concatenation with "+"

```
x.append(0.3) # x = x + [0.3] also works
print x[-1] # prints 0.3
```

Iterating through lists

- You have already seen looping through lists!

```
for i in range(1,10): #range returns a list
    print i
```

Iterating through lists

- You have already seen looping through lists!

```
for i in range(1,10): #range returns a list
    print i
```

- The same syntax works for our own lists.

```
x = [1, "hi", 3]
for i in x:
  print i
```

Iterating through lists

You can also loop by index - only do it if you need the index.

```
x = [1,2,3,4]
for i in range(len(x)): #range returns a list
    print i, x[i]
```

Strings as lists

- When it comes to square brackets accessing you can treat strings like lists.

```
x = "hello there"
print x[1] #prints e
for char in x: #iterates by character
        print char
```

Strings as lists

- When it comes to square brackets accessing you can treat strings like lists.

```
x = "hello there"
print x[1] #prints e
for char in x: #iterates by character
        print char
```

- But do not try to change individual characters in a string.

```
x[1] = 'i' #Error
```

Example Time!

Slicing

- In built into Python are slices - ways of taking sublists of lists and treating them as lists!
  Remember, closed left hand - open right hand intervals are the norm.

```
x = [1,"hi",72,0.3]
print x[1:3] # prints ["hi", 72]
```

Slicing

- In built into Python are slices - ways of taking sublists of lists and treating them as lists!
  Remember, closed left hand - open right hand intervals are the norm.

```
x = [1,"hi",72,0.3]
print x[1:3] # prints ["hi", 72]
```

```
print x[:2] # prints [1, "hi"] = x[0:2]
```

Slicing

- In built into Python are slices - ways of taking sublists of lists and treating them as lists!
  Remember, closed left hand - open right hand intervals are the norm.

```
x = [1,"hi",72,0.3]
print x[1:3] # prints ["hi", 72]
```

```
print x[:2] # prints [1, "hi"] = x[0:2]
```

```
print x[2:] #prints [72, 0.3] = x[2:4]
```

Slicing

- In built into Python are slices - ways of taking sublists of lists and treating them as lists!
  Remember, closed left hand - open right hand intervals are the norm.

```
x = [1,"hi",72,0.3]
print x[1:3] # prints ["hi", 72]
```

```
print x[:2] # prints [1, "hi"]  = x[0:2]
```

```
print x[2:] #prints [72, 0.3] = x[2:4]
```

```
print x[:] # Any guesses?
```

Slicing with steps

- Using a second colon, we can specify the step size to move through a list.

```
x = [1,"hi",72,0.3]
print x[:3:2] # prints [1, 72]
```

Slicing with steps

- Using a second colon, we can specify the step size to move through a list.

```
x = [1,"hi",72,0.3]
print x[:3:2] # prints [1, 72]
```

```
print x[:2:3] # prints [1]
```

Slicing with steps

- Using a second colon, we can specify the step size to move through a list.

```
x = [1,"hi",72,0.3]
print x[:3:2] # prints [1, 72]
```

```
print x[:2:3] # prints [1]
```

```
print x[::-1] #prints [0.3,72, "hi", 1]
```

Example Time!

Creating a dictionary

- Dictionaries are a way of storing a mapping from some data to other data.
- Dictionaries are denoted by squiggly brackets.

```
x ={}
```

Creating a dictionary

- Dictionaries are a way of storing a mapping from some data to other data.
- Dictionaries are denoted by squiggly brackets.

```
x ={}
```

- We can also initialise the dictionary with some elements

```
x = {1 : "one", 2: "two", 3 : "three"  }
```

Creating a dictionary

- Dictionaries are a way of storing a mapping from some data to other data.
- Dictionaries are denoted by squiggly brackets.

```
x ={}
```

- We can also initialise the dictionary with some elements

```
x = {1 : "one", 2: "two", 3 : "three"  }
```

To access these we use the square bracket accessors:

```
print x[1] # prints "one"
print x[3] # prints "three"
```

Example dictionary

- But we don't need to use integers as our index set.

```
x = {"four" : 4, "five": 5, "six" : 6 }
print x["four"] # prints 4
```

Example dictionary

- But we don't need to use integers as our index set.

```
x = {"four" : 4, "five": 5, "six" : 6 }
print x["four"] # prints 4
```

- We can also add new mappings to our dictionary

```
x["seven"] = 7
x["six"] = 11
```

Example Time!