Week 2 - Lecture 3

Istvan Albert
BMMB Bioinformatics
Homework deadline change

• Homework will be due on next week’s **Tuesday** rather than **Thursday**.

• Allows us to focus the entire week on the upcoming homework, rather than mixing the two

• Note: There are office hours on Monday before the Tuesday’s lecture.
List Containers

Lists are one of the simplest yet most important data containers. Think of them as a row of values.

Designated using the square brackets

- 
  - [ ] represents an empty list

- [ 1, 2, 3 ] is a list containing just numbers

- [ 1, “Hello”, [] ] is a list containing different types
List is an object type

```python
a = [1, 2, 3, 4]
print a
#print type(a)
#print dir(a)

# other builtin functions that operate on lists
print min(a), max(a), sum(a)
```

```
[1, 2, 3, 4]
1 4 10
```
Indexing and slicing → [start : end]
Index starts at 0!

data = ['a', 'b', 'c', 'd', 'e', 'f', 'g']

# first, last elements
print data[0]
print data[-1]

# first two elements
print data[0:2]

# last two elements
print data[-2:None]
print data[-2:]
Comparing the indexing systems

### 0 based indexing (half closed, BED format)

- **Third element**: `data[2]`
- **First ten**: `data[0:10]`
- **Second ten**: `data[10:20]`
- **Third ten**: `data[20:30]`

Size of the slice = 10 \( \rightarrow \) end - start

- Empty slice: `data[0:0]` – not sure

Five element long segment starting at 1000 `data[1000:10000 + 5]`

### 1 based indexing (inclusive, GFF format)

- **Third element**: `data[3]`
- **First ten**: `data[1:10]`
- **Second ten**: `data[11:20]`
- **Third ten**: `data[21:30]`

Size of the slice = 10 \( \rightarrow \) end - start + 1

- Empty slice: `data[?]` – not sure

Five element long segment starting at 1000 `data[1000:10000 + 4]`
Advanced slicing ➞ [ start: end: step ]

data = ['a', 'b', 'c', 'd', 'e', 'f', 'g']

# going by two for the first four
print data[0:4:2]

# going by three over the entire data
print data[None:None:3]
print data[::3]

# this actually reverses the data
print data[:::-1]

# negative strides
# can be counter intuitive

python -t C:\cygwin\home\ialbert\sources\ialbert-web\ppt\week2\test-index.py 2> &1 ` returned 0

[ 'a', 'c' ]
[ 'a', 'd', 'g' ]
[ 'a', 'd', 'g' ]
[ 'g', 'f', 'e', 'd', 'c', 'b', 'a' ]
Look at the list methods with `dir()`

```python
data = [ 3.0, 1.0, 5.0, 2.0, 4.0 ]
data.sort()
print data
```

Output:

```
[1.0, 2.0, 3.0, 4.0, 5.0]
```
Mapping Functions

Map = “Mass Transformation”

\[
\text{map}( \cos, [1, 2, 3] ) = [ \cos(1), \cos(2), \cos(3) ]
\]

A way to execute a function on each element of a list, and return the mapped values as a new list
Mapping Example

```python
# original data
data = [ '1', '4', '2', '5', '3']
print(data)

# remapped data
values = map(float, data)
print(values)

# derived quantities
print(sum(values))
print(sum(values)/len(values))
```

Output:
```
['1', '4', '2', '5', '3']
[1.0, 4.0, 2.0, 5.0, 3.0]
15.0
3.0
```
Start with the string\texttt{a=“10 8 10 9 2 3 1”} find the \texttt{sum} of the last five \texttt{numbers} in this string (use comments to document the code).

Have the following documented:

- find the method that splits strings (hint: \texttt{dir(a)})
- split the string into numbers, print the type of the result of the split
- print the type of the third element
- transform to float numbers, print the type of the result
- print the type of the third element
- slice and sum, print the sum
Homework 1 (cont)

Do the same as in homework 2 but now sort the data before slicing and summing:

- you could be sorting right after you split the numbers from a string
- you could sort right after you map the data to float

Does it affect the results?
Week 2 - Lecture 4

Istvan Albert
BMMB Bioinformatics
1. \( a = 1 + 2 \) returns a value that sets \( a=3 \)

2. But `values = data.sort()` is incorrect why?

   `data.sort()` mutates the original dataset

   (advantageous for large datasets)

There is also a function called `sorted()`:

   `data = sorted(data)`

   (better programming practice, the disadvantage is that for a short period both sorted and unsorted data coexist)
Data naming tips

```python
words = [ '3.0', '1.0', '5.0', '2.0', '4.0' ]
words = words[1:-1]
print words

values = map(float, words)
values = sorted(values, reverse=True)
values = values[1:]
print values
```

```
[ '1.0', '5.0', '2.0' ]
[2.0, 1.0]
```
Introducing functions

• So far we only named values `greeting=“Hello”`

• But we can also assign a series instructions to a name – these are called functions

• Makes it easy to call the same instructions repeatedly

• You can also pass them to `map` and other tools
def name ( param1, param2 ):
    instruction 1
    instruction 2
    instruction 3
    return value

The indentation determines what instructions are part of the function
A simple function

def double(x):
    return 2 * x

print type(double)

print double(2)

print double(10)
Add a “docstring” (help) to your functions.

A docstring is a string right below the function definition. It just hangs there, not assigned to anything.
Use them for transformations

```python
# Function to double a number
def double(x):
    return 2 * x

# List of numbers
once = [1, 2, 3]
twice = map(double, once)

# Print the results
print(once)
print(twice)
```

```
[1, 2, 3]
[3, 5, 7]
```
What happens when we map functions

def myfloat(x):
    print 'Inside myfloat, parameter =', x
    return float(x)

data = [1, 2, 3, 4]

values = map(myfloat, data)

print values

`python -t C:\cygwin\home\ialbert\sources\ialbert-web\ppt\week2\test-function2.py 2>&1` returned 0

Inside myfloat, parameter = 1
Inside myfloat, parameter = 2
Inside myfloat, parameter = 3
Inside myfloat, parameter = 4
[1.0, 2.0, 3.0, 4.0]
The built-in function `range()`

```
print range(5)
print range(-5, 5)
print range(-5, 5, 2)
```

```
[0, 1, 2, 3, 4]
[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
[-5, -3, -1, 1, 3]
```
Filtering data

Filter $\rightarrow$ data removal

\[
\text{filter( function, data )} = \left[ \text{values where function(value) is True} \right]
\]

Or you can think of it as removing those elements from the list where the function evaluates to false.
Define a simple function

```python
def positive(x):
    return x >= 0

print positive(300.0)
print positive(-20.0)
```

Command Output:
```
True
False
```
Applying a filter. Keep positive values

```python
def positive(x):
    return x >= 0

values = [-1, 2, 3, -7, 4]

print(list(filter(positive, values)))
```

Output: `[2, 3, 4]`
Map and filter allow you to solve a lot of problems

Identify the steps needed to process the data:

– To remove data use filter

– To transform data use map

The process will likely include several mapping and filtering steps.
Homework: next Tuesday

• Do the examples shown in class (these do not need to be turned in)

• Using map transform a list of integers ranging from -10 to 10 with the function: $x^3 - 2x^2 + 100$ (use ** for exponentiation operator)

• Using filter create a function that keeps only the elements that have an absolute value smaller than 500

• What happens if by accident you use the filtering function with the map(). What does the data contain?