

BI694

Bioinformatics & Phylogenetics

Winter Semester 2017

WEEK 6

Homework review
Basics of Python Programming

Review of Assignment 1

```
#!/bin/bash

# retrieve data from github repository

wget https://raw.githubusercontent.com/bastodian/2017_BioinformaticsPhylogenetics/master/Week_3/Assignment_1/Hydrozoa.fst

# take last 40 sequences and copy them into another fasta file

tail -n 80 Hydrozoa.fst > Hydrozoa_Last40-Seqs.fst

# make file read only

chmod 444 Hydrozoa_Last40-Seqs.fst

# align the last 40 sequences using MUSCLE and CLUSTAL

muscle -in Hydrozoa_Last40-Seqs.fst -out Hydrozoa_Last40-Seqs.MUSCLE.fst

clustalo -i Hydrozoa_Last40-Seqs.fst -o Hydrozoa_Last40-Seqs.CLUSTAL.fst

# open the alignments in Seaview and put Seaview into the background using &

seaview Hydrozoa_Last40-Seqs.MUSCLE.fst &

seaview Hydrozoa_Last40-Seqs.CLUSTAL.fst &
```

Let's check our BLAST runs...

Connect to the server again using ssh

Connect to your screen session:

- screen -ls
- screen -r BLAST

Are we done yet? If not why not?

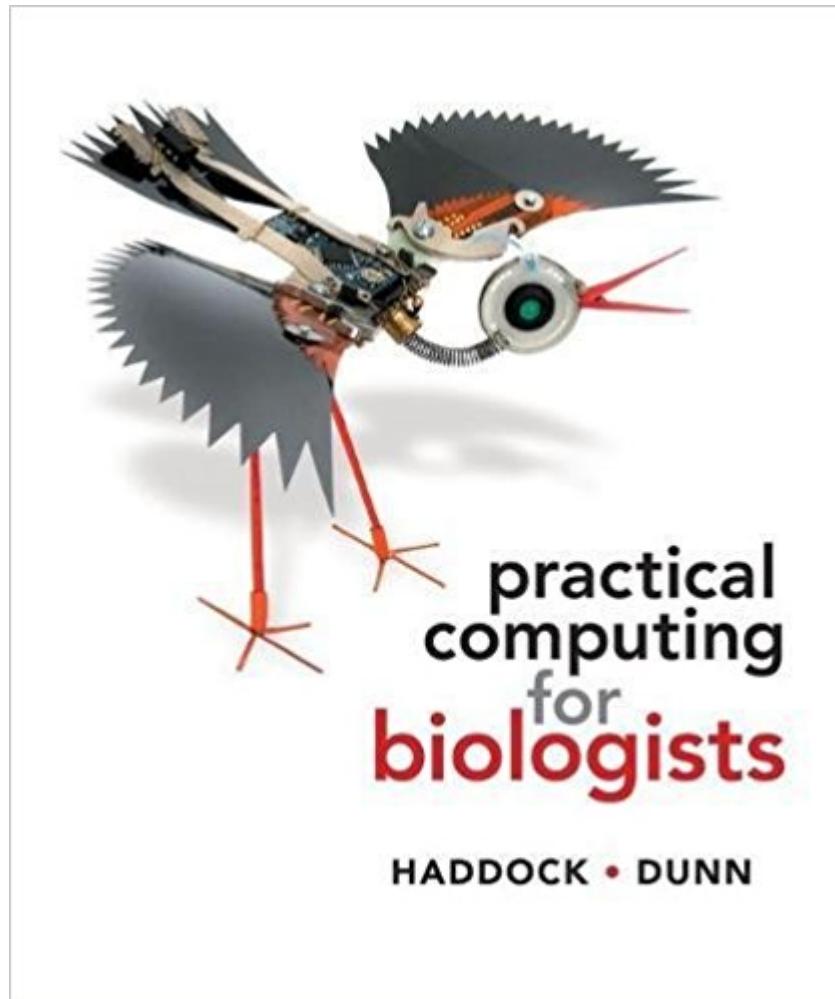
Parsing BLAST output

- Parsing tabular BLAST output using grep and awk
 - Keep only selected rows
 - Extract select columns
- We can do this more elegantly with Python
 - Python is an interpreted language like Bash
 - Python is easy to read and intuitive (in my opinion)
 - Python has lots of add-on features specifically targeted to bioinformatics

Install Some Additional Software

```
$ sudo apt-get install ipython      # interactive python shell  
$ sudo apt-get install geany        # Integrated Development Environment (IDE)
```

Components of Programming



Chapters 7 through 13 cover Programming in Python.

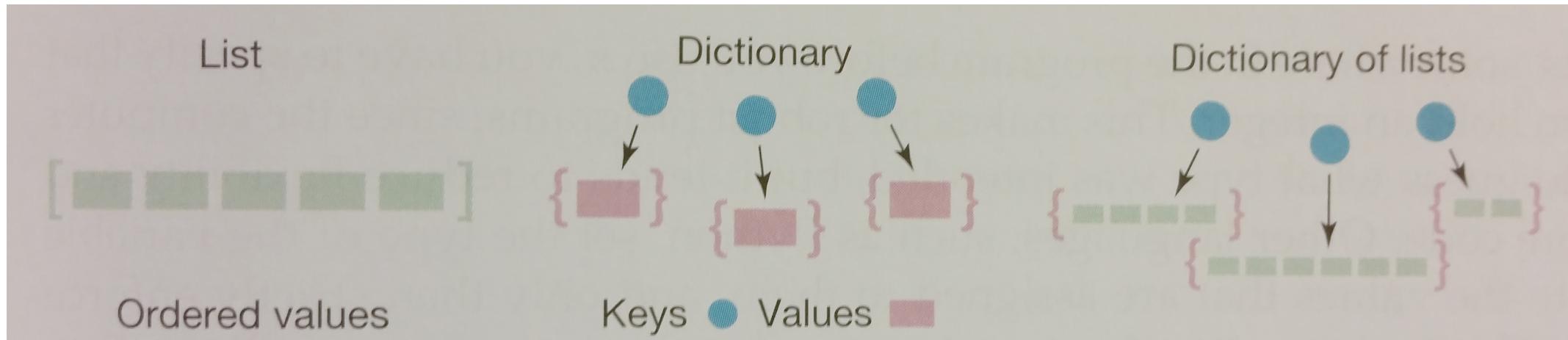
Components of Programming

- Arguments
 - Values that are sent to the program at the time it is run
- Code
 - Noun: a program or line of a program, also called source code
 - Verb: the act of writing a program
- Execute/Run
 - To begin and carry out the operation of a program
- Parse
 - To extract particular data elements from a larger block of text

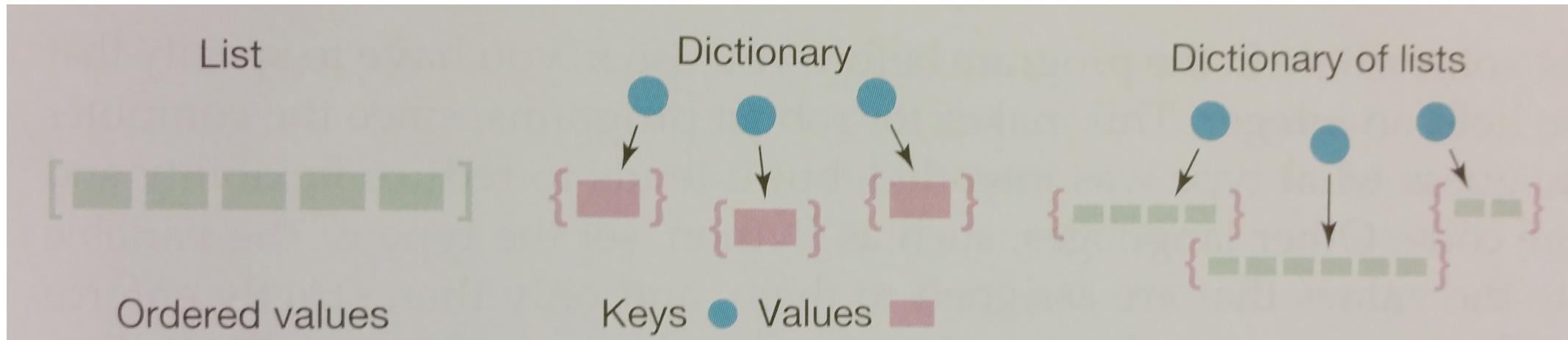
Components of Programming – Variable types

Type	Example	Python Function
Integer	98	int()
Float	98.6	float()
Boolean	False	bool()
String	'Acropora digitifera'	str()

Components of Programming – Lists, Dictionaries, Tuples, Sets

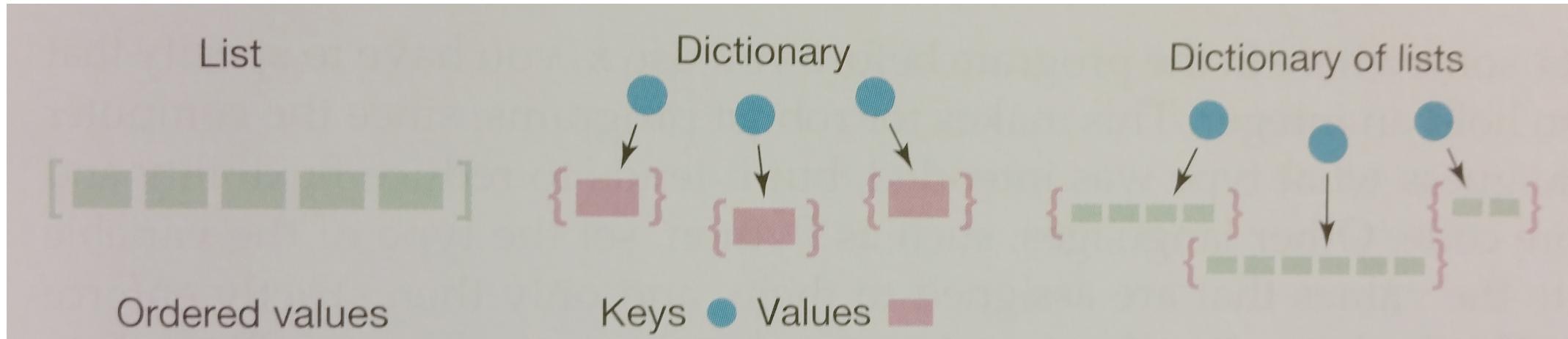


Components of Programming – Lists, Dictionaries, Tuples, Sets

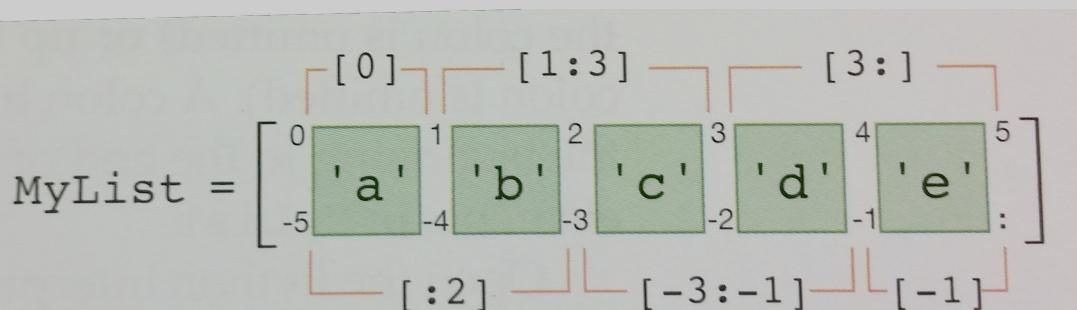


```
$ListOfIntegers = [1, 2, 4, 5]      # ordered collection of values, indexed starting at 0  
$ListOfStrings = ['a', 'b', 'c', 'd', 'e']
```

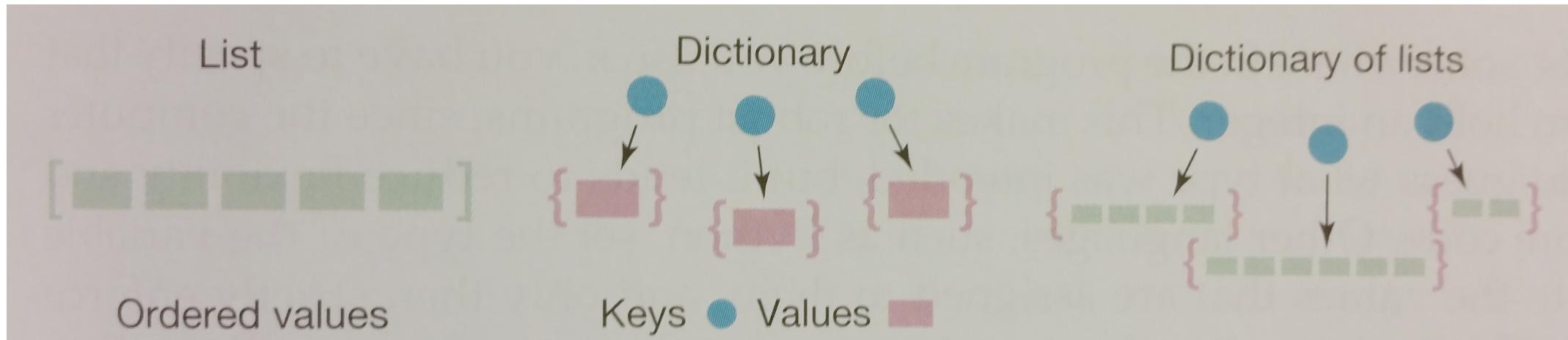
Components of Programming – Lists, Dictionaries, Tuples, Sets



```
$ListOfIntegers = [1, 2, 4, 5]      # ordered collection of values, indexed starting at 0  
$ListOfStrings = ['a', 'b', 'c', 'd', 'e']
```

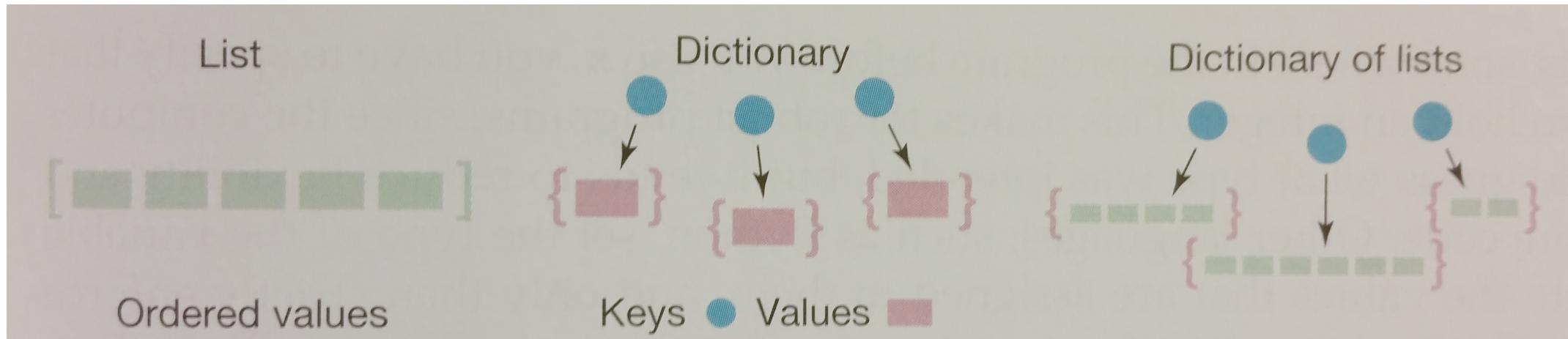


Components of Programming – Lists, Dictionaries, Tuples, Sets



```
$ListOfIntegers = [1, 2, 4, 5]          # ordered collection of values, indexed starting at 0
$ListOfStrings = ['a', 'b', 'c', 'd', 'e']
$DictionaryOfIntegers = {'A': 1, 'B': 2}  # also known as a hashtable; indexed using keys
$DictionaryOfLists = {'A': [1, 2, 3], 'B': [3, 4, 5]}
```

Components of Programming – Lists, Dictionaries, Tuples, Sets



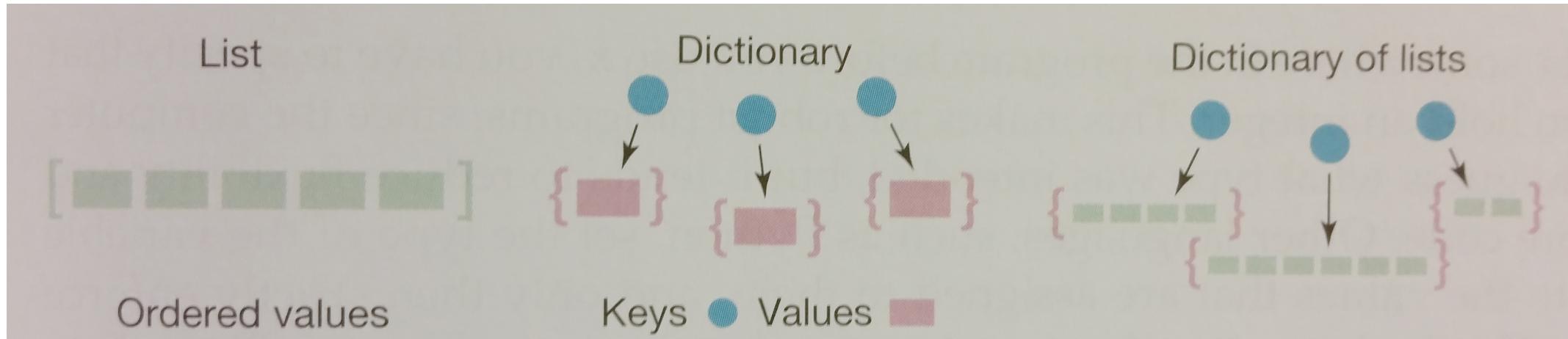
```
$ListOfIntegers = [1, 2, 4, 5]          # ordered collection of values, indexed starting at 0
$ListOfStrings = ['a', 'b', 'c', 'd', 'e']
$DictionaryOfIntegers = {'A': 1, 'B': 2}  # also known as a hashtable; indexed using keys
$DictionaryOfLists = {'A': [1, 2, 3], 'B': [3, 4, 5]}
```

Components of Programming – Lists, Dictionaries, Tuples, Sets

Create a dictionary and add data to it. Here, the keys are strings of specimens names and the values in the dictionary are floating point numbers (latitude and longitude).

```
$ TreeStat = {}      # creates an empty dictionary  
$ TreeStat['Kodiak'] = [57.8, -152.5]  
$ TreeStat['Juneau'] = [58.3, -134.5]
```

Components of Programming – Lists, Dictionaries, Tuples, Sets



```
$ListOfIntegers = [1, 2, 4, 5]          # ordered collection of values, indexed starting at 0
$ListOfStrings = ['a', 'b', 'c', 'd', 'e']
$DictionaryOfIntegers = {'A': 1, 'B': 2}  # also known as a hashtable; indexed using keys
$DictionaryOfLists = {'A': [1, 2, 3], 'B': [3, 4, 5]}
$TupleOfIntegers = (1, 2, 3)            # ordered, immutable list
$SetOfIntegers = {1, 2, 3}              # unordered collection of unique elements
```

Components of Programming – Objects and Functions (Methods)

What can we do with sets?

For example, we can find the intersection between two lists...

```
$ dir(set)      # lists the functions associated with set
```

Components of Programming – Objects and Functions (Methods)

What can we do with sets?

For example, we can find the intersection between two lists...

```
$ dir(set)      # lists the functions associated with set
```

Consider **set** as an object. Set has many properties associated with it, which can be listed using **dir**.

These attributes can be accessed using the **dot notation**.

```
$ set(A).intersection(set(B))
```

Components of Programming – Objects and Functions (Methods)

What can we do with sets?

For example, we can find the intersection between two lists...

```
$ dir(set) # lists the functions associated with set
```

Consider **set** as an object. Set has many properties associated with it, which can be listed using **dir**.

These attributes can be accessed using the **dot notation**.

```
$ set(A).intersection(set(B))
```

```
$ SetOfIntegers_A = {1, 2, 3, 7, 10}
```

```
$ SetOfIntegers_B = {1, 2, 3, 4, 5, 11}
```

```
$ SetOfIntegers_A.intersection(SetOfIntegers_B)      # what does .union produce?
```

Components of Programming – Objects and Functions (Methods)

An example of functions associated with string objects:

```
$ DNASEq = 'ATGCAC'  
$ dir(DNASEq)  
$ DNASEq.isdigit()  
$ DNASEq.lower()  
$ DNASEq.startswith('ATG')  
$ DNASEq.endswith('cac')    # case sensitive!!
```

Components of Programming – Converting Between Variables

Variables can be converted into each other. Example: **str()**, **int()**, and **float()**

```
$ print '7' + str(3*2)  
$ print 7 + str(3*2)      # does this work?  
$ print 7 + 3*2  
$ print float(7) + 3*2
```

Components of Programming – Operators

Mathematical

Addition	+
Subtraction	-
Multiplication	*
Division	/
Power	**
Modulo	% # remainder after division
Truncated Division	//

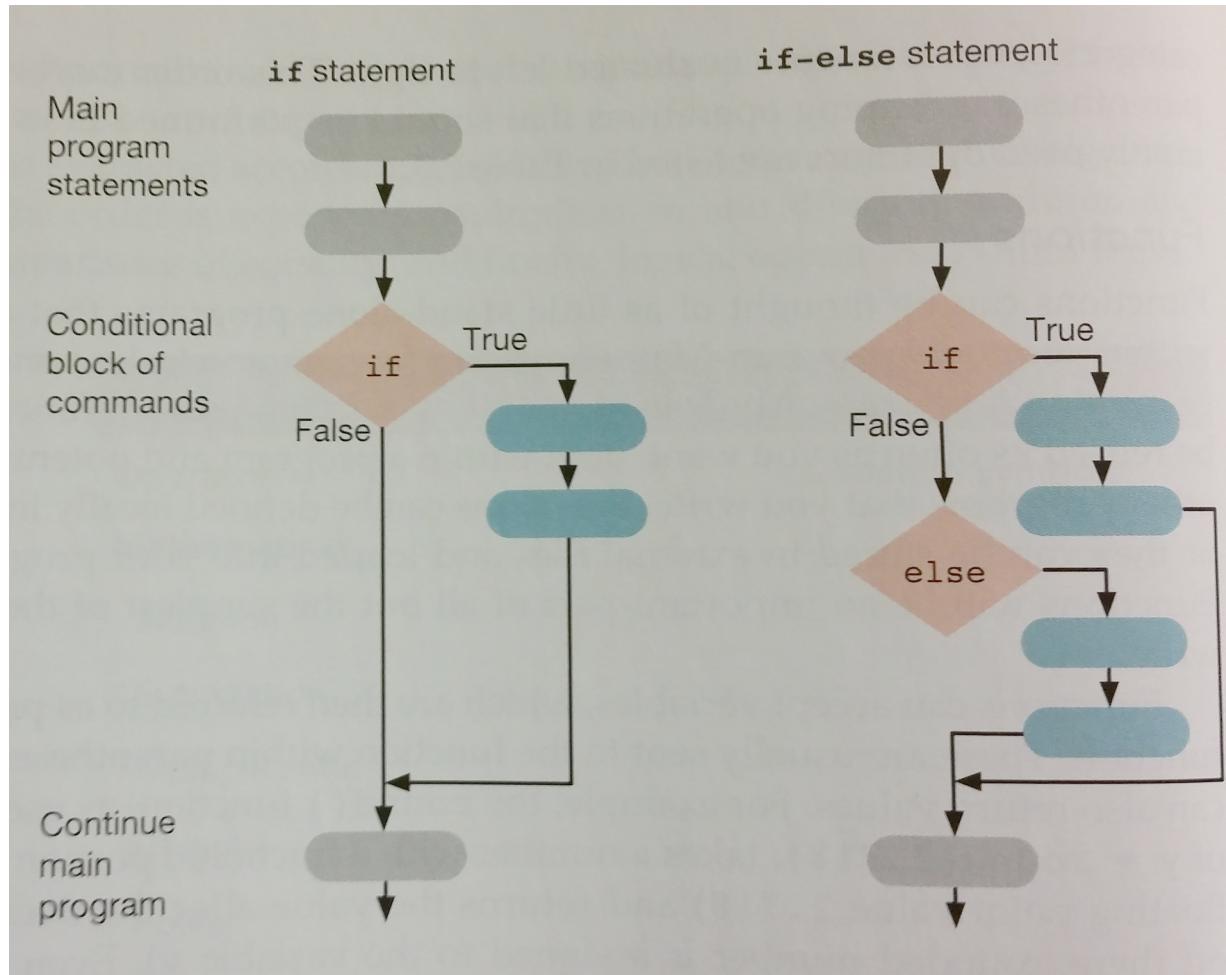
Comparative

Equal to	==
Not equal to	!=, <>, ~=
Greater than	>
Less than	<
Greater or equal	>=
Less or equal	<=

Logical

And	and, &, &&
Or	or, ,
Not	not, !, ~

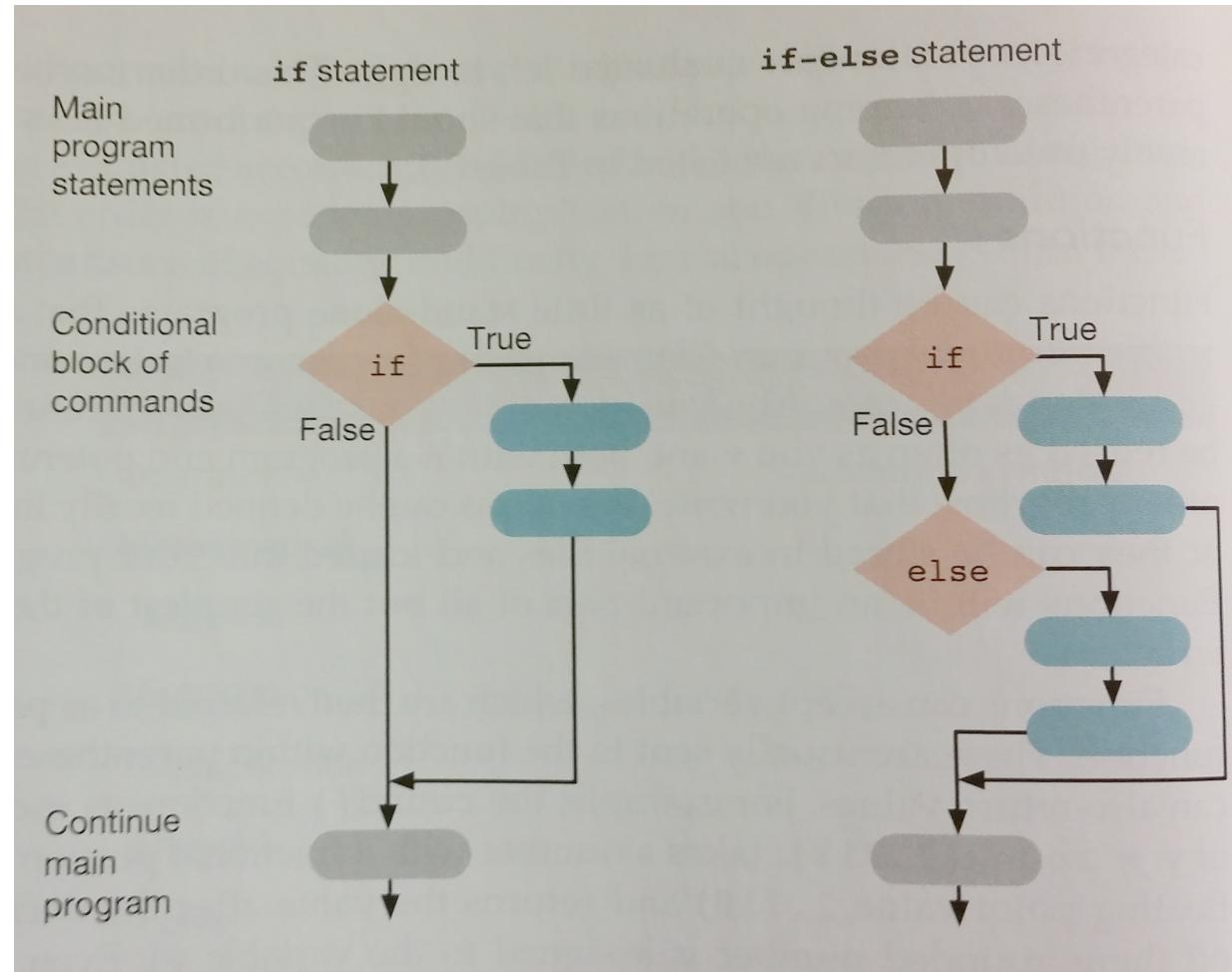
Components of Programming – Conditionals



A = 5

```
if A < 0:  
    print "Negative Number"  
else:  
    print "Zero or positive number"
```

Components of Programming – Conditionals

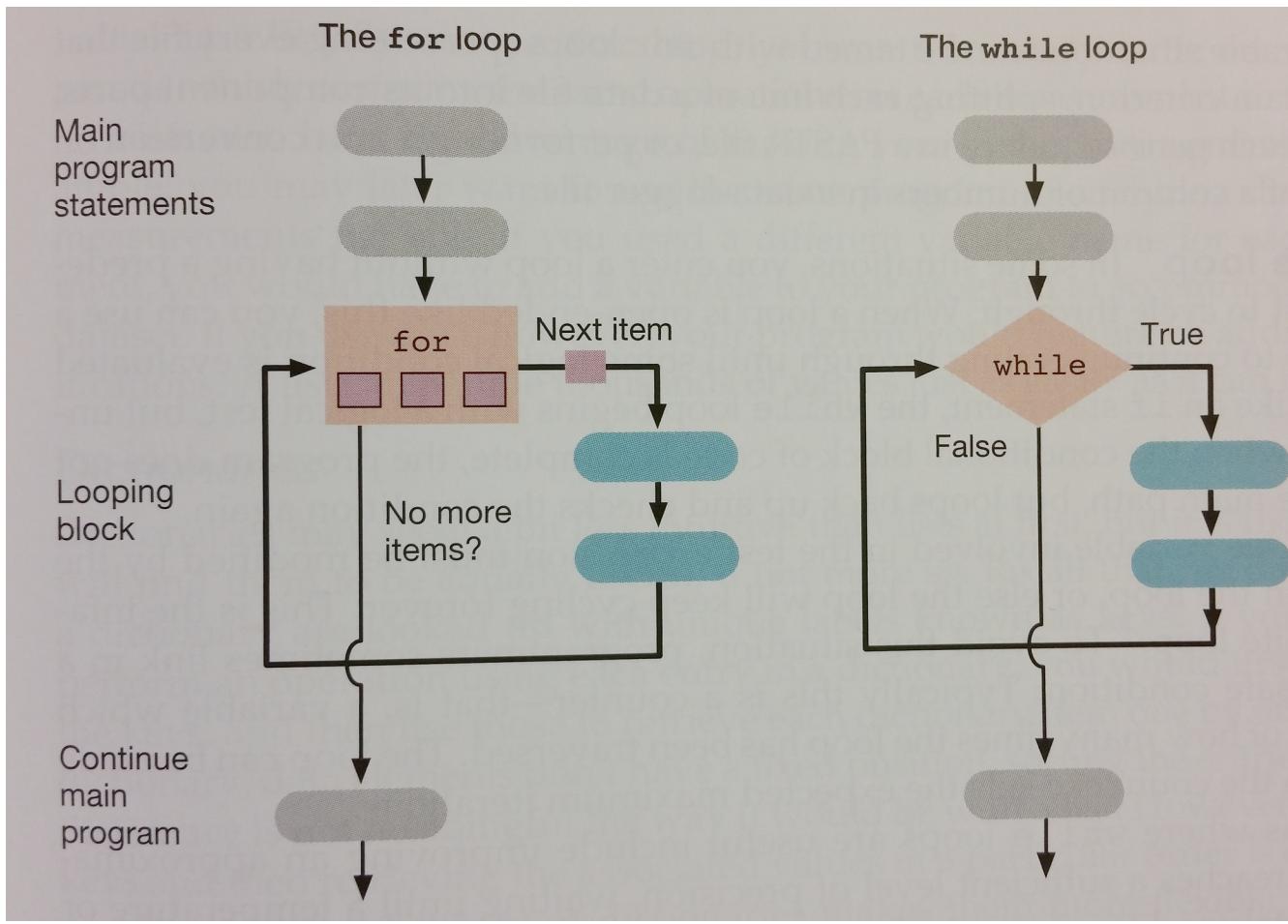


A = 5

```
if A < 0:  
    print "Negative Number"  
else:  
    print "Zero or positive number"
```

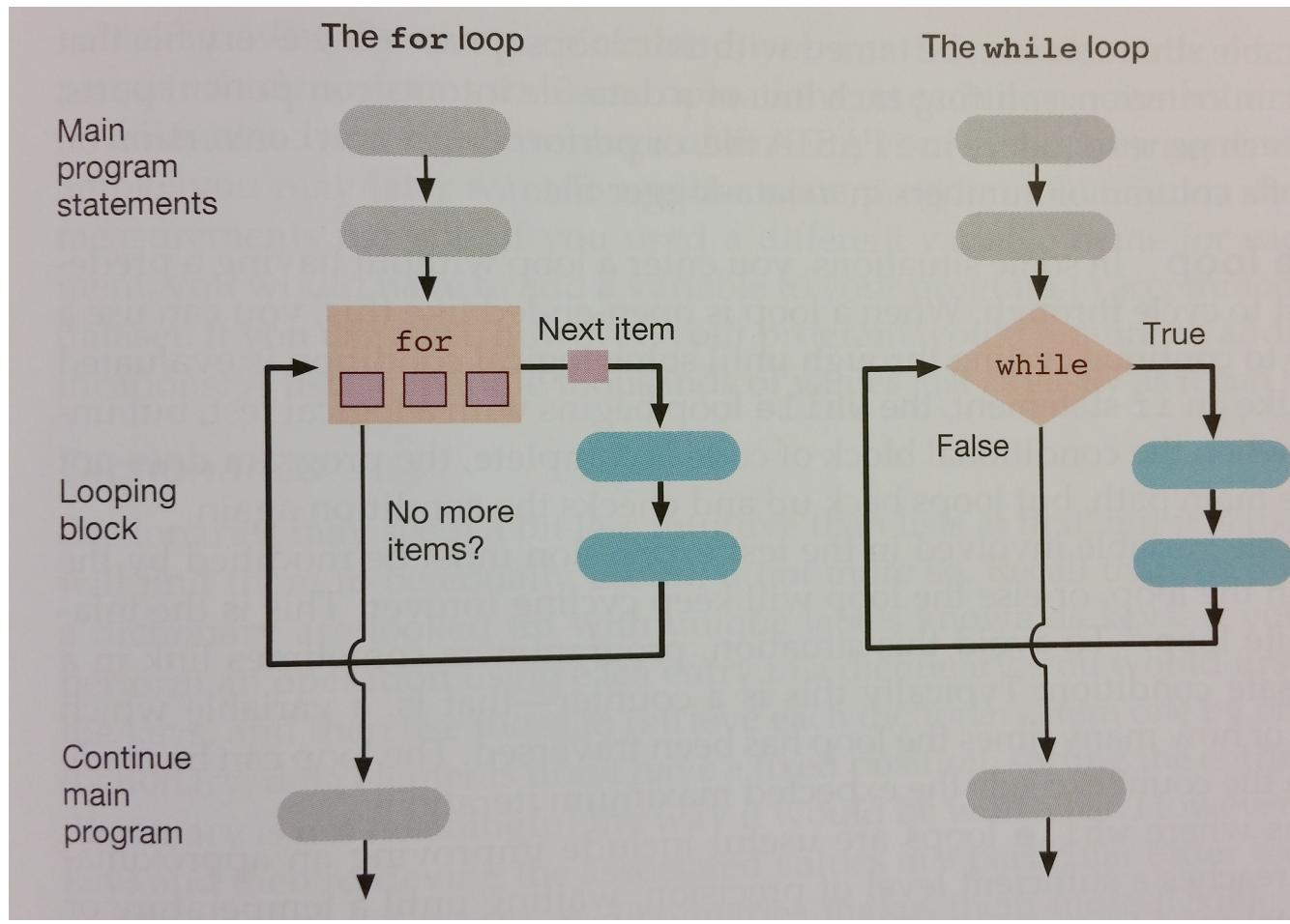
```
if A < 0:  
    print "Negative Number"  
elif A > 0:  
    print "Positive number"  
else:  
    print "Zero"
```

Components of Programming – Loops



```
for Num in range(10):  
    print Num
```

Components of Programming – Loops



```
for Num in range(10):  
    print Num
```

Num = -1

```
while Num <= 9:  
    num += 1  
    print Num
```

Components of Programming – Readin/Writing Files

```
BLAST = open('Acropora_genes.blastn', 'r')
for Line in BLAST:
    print Line
BLAST.close()
```

Components of Programming – Readin/Writing Files

```
BLAST = open('Acropora_genes.blastn', 'r')
```

```
for Line in BLAST:
```

```
    print Line
```

```
BLAST.close()
```

```
with open('Acopora_genes.blastn', 'r') as BLAST:
```

```
    with open('Temp.txt', 'w') as TEMP:
```

```
        for Line in BLAST:
```

```
            TEMP.write(Line)
```

Components of Programming – Parsing Data From Files

