Introduction to Bioinformatics

Control Structures, Reading Files in Python, Regular Expressions

Janyl Jumadinova
February 9-11, 2016
Python Data Type

These are the most basic (and most frequently used) data types:

- Integer: 0, 7, 13, 21, -1
- Floating point: 3.14, 2.718, 1.618
- String: “1”, “2.718”, “True”, “None”, “My name is ”
- Boolean: True, False
- Null: None
Python Data Structures

- **List**: an ordered, zero-indexed collection of objects. For example: [1, “A”, 3.0]

- **Set**: an unordered collection of elements, guarantees each element is unique. For example: {1, 5, 3.0}

- **Dictionary**: an unordered collection of key/value pairs. Each key is unique. For example: {1:“One”, “A”:5, 3.0:“Three”}
Python Ordered Types

- **Strings** ‘this’ or “this”... - **immutable**
  - S=‘act’; s.upper(); s.lower()
  - S.replace(old, new,[count])
  - S.count(sub,[start[,end]])
  - Len(s)
  - Find
  - Index
  - Split
  - join

- **Tuples**
  - Pt=(2,4,1)
  - Pt[0]
  - immutable

- **Lists** - are **mutable**
  - L1=[1,2,3]; L1[0]
  - L2 = ['aat', 'tft', 5]
  - Set: A={1,2,3}; [2*x for x in A]
  - List(string)
  - L2=copy.copy(l1)
  - L.append(33)
  - L.insert(pos,elem)
  - L.Extend(l2) same as l1+l2
  - L1.pop(2)
  - L.remove(elem) or del list[2]
  - L.count(x)
  - L.index(x)
  - L.reverse()
  - L.sort()
Python Unordered Types

- **Dictionaries** - stores indexed data
  - 1) \{key:value, key:value\}
  - code=\{‘A’:‘Ala’, ‘C’:‘Cys’\}
  - code[‘C’]
  - 2) code = dict(a=‘Ala’,c=‘Cys’)
  - Dict(‘d’)=‘dist’
  - for elem in dict:
    - Print elem
  - Code.keys(); code.values()
  - Code.items()
  - Code.get(‘A’,’Not in dict’)
  - Del code[‘A’]

- **Sets** - like a list BUT with unique elements & no order
  - S1=set ([1,2,’pink’,5])
  - S1.add(‘red’)
  - S1.intersection(s2)
  - S1.union(s2)
  - S1-s2
  - S1.symmetric_difference(s2)
  - Min, max, len, in
  - List(set1)
  - Frozenset(list) -- immutable
Python basic language structure

- Newline terminates a command - no semicolon required.
- Indentation alone designates nested code blocks - no curly braces required.
- Functions, nested loops and conditionally evaluated code are all indicated using indentation.
- `#` denotes the start of a single line comment.
Control Structures

1. Sequential Structure
2. Selection Structures
   - if: single selection
   - if/elif/else: double or multiple selection
Control Structures

1. Sequential Structure
2. Selection Structures
3. Repetition Structure
   
   while
   
   for

Loop (Iteration)
Reading Files

- `fileInput = open('file-name.fasta')`
Reading Files

- `fileInput = open('file-name.fasta')`
- Can process line by line:
  ```
  for line in fileInput:
    ...
  ```
Reading Files

- `fileInput = open('file-name.fasta')`
- Can process line by line:
  
  ```python
  for line in fileInput:
      ...
  ```

- `fileInput.close()`
Writing to Files

- \( \text{fileOutput} = \text{open('results-name.fasta', 'w')} \)
Writing to Files

- `fileOutput = open('results-name.fasta', 'w')`
- Can write to the file: `fileOutput.write`
Writing to Files

- `fileOutput = open('results-name.fasta', 'w')`
- Can write to the file:
  ```python
  fileOutput.write:
  ```
- `fileOutput.close()`
Regular Expression

also referred to as “regex” or “regexp”, provides a concise and flexible means for matching strings of text, such as particular characters, words, or patterns of characters. A regular expression is written in a formal language that can be interpreted by a regular expression processor.
Regular Expression

- Very powerful, compact and cryptic
- Regular expressions are a language
- A language of “marker characters” - programming with characters
Regular Expression

\^ Matches the **beginning** of a line
\$ Matches the **end** of the line
. Matches **any** character
\s Matches **whitespace**
\S Matches any **non-whitespace** character
* Repeats a character zero or more times
*? Repeats a character zero or more times (non-greedy)
+ Repeats a character one or more times
+? Repeats a character one or more times (non-greedy)
[aeiou] Matches a single character in the listed set
[^XYZ] Matches a single character **not in** the listed set
[a-zA-Z0-9] The set of characters can include a **range**
( Indicates where string **extraction** is to start
) Indicates where string **extraction** is to end
We fine-tune what is matched by adding special characters to the string

- The hat character matches the start of the line
- The dot character matches any character
- If you add the asterisk character, the character is “any number of times”
- Plus means “one or more times”