KILI: Seminar 2

Datatypes

14-15 September 2018

1 Numbers

- For numbers, there are three basic types: integer (3), floating point (7.2), and complex (2+4j).

-
  ```
  type()
  ```

  This function can operate on strings or numbers and returns the general type of the object it applies to.
- convert from string:

  -
    ```
    int('3')
    ```

    ```
    float('7.2')
    ```

  -
    ```
    complex('2+4j')
    ```

2 Booleans

- There is also a basic datatype for *booleans*, expressions that are true or false.
- This class includes only two basic members: True and False. It also includes the basic logical operators: and, or, and not.

-
  ```
  not (False or True)
  ```

  ```
  (not False) or True
  ```
- What do the following numeric comparisons mean when they are applied to strings?
  - ==
  - !=
  - >
  - >=
  - <

    – <=

(1) A HINT: 'flower' < 'zebra'   precedes alphabetically

- isinstance() – we can use to test if some element or variable is of any particular type.

  ```
  isinstance(3,int)
  ```

3 Strings

- 'hat' == "hat"
- "'This is a string that continues on more than one line"'
- len(x) – to determine the length of x
- x.count('a') – to count letters 'a' in x
- x.upper() – upper case
- x.lower() – lower case
- Pragmatics: sentiment, focus

  x = 'The bartender... absolutely horrible... we waited 10 min before we even got her attention... and then we had to wait 45 – FORTY FIVE! – minutes for our entrees... stalk the waitress to get the cheque... she didn't make eye contact or even break her stride to wait for a response ...'

  What will happen if we apply lower() to the string x?

  What information will we lose when lower() is applied to x?

  List one linguistic task where the application of lower() to x is useful and one task where it is harmful.

- x = '{} Mike'
- y = x.format('Hello')
- x = '{} Mike. {}?'
- x.format('Hello','How are you')
- x = 'one = {1}; two = {0}'
- x.format('dos','uno')
- 'one = uno; two = dos'
- Semitic morphology

  Semitic morphology involves intercalating vowels and consonants to express morphological categories. For example, the Arabic root k,t,b occurs in at least the following forms: *katab-a* 'he wrote', *kaatab-a* 'he corresponded', *kutib- a* 'it was written', *kitab* 'book', *kuttaab* 'writers', *uktub* 'write!', etc.

How might you use *format*() to describe this system? Give a sample representation for the root k,t,b and how *format*() could be used to express different categories.

- x[1] – a syntax for extracting part of a string. The syntax is to put one or more integers in square brackets after the string. Using a single integer in the square brackets extracts a single character. The characters in a string are counted from the left, starting at 0.
- x[2:4] – one can also refer to a sequence of characters by using two integers separated by a colon. The first index is the starting point of the sequence; the second index is just after the end of the sequence.

4 Lists

- Lists are an extremely important datatype: a single structure that holds a sequence of elements of whatever type you want. You can create a list by simply listing elements in square brackets: x = ['stops','fricatives','glides'].
- len(x) – returns the length of x
- x[1] – to index into a list
- x.append('vowels') – to add to a list
- x.pop(1) – to remove an element at a specified index position in a list. The method returns that element, altering the list at the same time.
- x.insert(1,'nasals') – takes two arguments, the index and the element to insert. The element is inserted just before the index you specify.
- x = list(range(4)) – this takes a single integer argument and returns a range object that represents the sequence from 0 up to that argument. This can be directly converted to a list with the list() function.
- x.sort() – to sort a list
- x.reverse() – to reverse a list
- 8 in x – to test for membership in a list

5 Tuples

- Another datatype that you will see quite often is a *tuple*, a fixed sequence of elements, similar to lists in many ways. The key difference is that tuples are fixed in length and, once created, cannot be changed.
- You create a tuple with parentheses. An empty tuple is just parentheses: ().
- y = (7,5,6)
- len(y) – the length of y
- y[2] – the second element of y
- 7 in y – to test for membership in y

- type(y)
- v = list(y) – to turn y into a list

6 Dictionaries

- One of the most useful datatypes for linguists are dictionaries. Dictionaries are effectively sets of pairs, where the first element in the pair can be used to look up the second. The set of first elements must thus be unique. A dictionary is marked with curly brackets where each pair of elements is separated with a colon.
- d = {'cat':'koshka','chair':'stul'}
- d['cat'] – to return the value for the key 'cat'
- len(d) – the length of d
- 'cat' in d – to test for membership in d
- d['cat'] = d['cat'] + 'kot' – to alter the value of the key 'cat'
- del(d['cat']) – to delete the item
- list(d.keys()) – to extract keys from a dictionary as a list
- list(d.values()) – to extract values from a dictionary as a list
- list(d.items()) – to extract items from a dictionary as a list

7 Sets

- A *set* is a collection which is unordered and unindexed. In Python sets are written with curly brackets.
- r = {"apple", "banana"}
- add () – adds an element to the set
- clear() – removes all the elements from the set
- copy() – returns a copy of the set
- difference() – returns a set containing the difference between two or more sets
- difference update() – removes the items in this set that are also included in another, specified set
- discard() – remove the specified item
- intersection() – returns a set that is the intersection of to other sets
- isdisjoint() – returns whether two sets have an intersection or not
- issubset() – returns whether another set contains this set or not

- issuperset() – returns whether this set contains another set or not
- pop() – removes the specified element
- remove()– removes the specified element
- symmetric difference() – returns a set with the symmetric differences of two sets
- symmetric difference update() – inserts the symmetric differences from this set and another
- union() – return a set containing the union of sets
- update() – update the set with the union of this set and others