

A first look at string processing

Python

Strings

- Basic data type in Python
 - Strings are immutable, meaning they cannot be shared
 - Why?
 - It's complicated, but string literals are very frequent. If strings cannot be changed, then multiple occurrences of the same string in a program can be placed in a single memory location.
 - More importantly, strings can serve keys in key-value pairs.
 - Don't worry, we are going to see what this means.

String Literals

- String literals are defined by using quotation marks

- Example:

```
>>> astring = "Hello World"
>>> bstring = 'Hello World'
>>> astring == bstring
True
```

- To create strings that span newlines, use the triple quotation mark

```
>>> cstring = """This is a very
complicated string with a few
line breaks."""
>>> cstring
'This is a very\ncomplicated string with a few\nline breaks.'
```

String Methods

- There are a number of methods for strings. Most of them are self-explaining
 - `s.lower()`, `s.upper()` : returns the lowercase or uppercase version of the string
 - `s.strip()` : returns a string with whitespace removed from the start and end
 - `s.isalpha()`/`s.isdigit()`/`s.isspace()` : tests if all the string chars are in the various character classes
 - `s.startswith('other')`, `s.endswith('other')` : tests if the string starts or ends with the given other string
 - `s.find('other')` : searches for the given other string (not a regular expression) within `s`, and returns the first index where it begins or -1 if not found
 - `s.replace('old', 'new')` : returns a string where all occurrences of 'old' have been replaced by 'new'

Strings and Characters

- Python does not have a special type for characters
 - Characters are just strings of length 1.

Accessing Elements of Strings

- We use the bracket notation to gain access to the characters in a string
 - `a_string[3]` is character number 3, i.e. the fourth character in the string

String Processing

- Since strings are immutable, we process strings by turning them into lists, then processing the list, then making the list into a string.
- String to list: Just use the list-command

```
>>> a_string = "Milwaukee"  
>>> list(a_string)  
['M', 'i', 'l', 'w', 'a', 'u', 'k', 'e', 'e']
```

String Processing

- Turn lists into strings with the join-method
 - The join-method has weird syntax
 - `a_string = "".join(a_list)`
 - The method is called on the empty string ""
 - The sole parameter is a list of characters or strings
 - You can use another string on which to call join
 - This string then becomes the glue

```
gluestr.join([str1, str2, str3, str4, str5])
```

str1	gluestr	str2	gluestr	str3	gluestr	str4	gluestr	str5
------	---------	------	---------	------	---------	------	---------	------

String Processing

- Examples

```
>>> a_list = ['M', 'a', 'h', 'a', 'r', 'a', 's', 'h', 't', 'r', 'a']
>>> "".join(a_list)
'Maharashtra'
>>> " ".join(a_list)
'M a h a r a s h t r a'
>>> "_".join(a_list)
'M_a_h_a_r_a_s_h_t_r_a'
>>> "oho".join(a_list)
'Mohoahohohoahorohoahosohohotohorohoa'
```

String Processing

- Procedure:
 - Take a string and convert to a list
 - Change the list or create a new list
 - Use join to recreate a new string
- Alternative Procedure:
 - Build a string one by one, using concatenation (+ -operator)
 - Creates lots of temporary strings cluttering up memory
 - Which is bad if you are dealing with large strings.

String Processing

- Example: Given a string, change all vowels to increasing digits.
- This is used as a (not very secure) password generator
 - Examples:
 - `Wisconsin` \rightarrow `W1sc2ns3n`
 - `AhmedabadGujaratIndia` \rightarrow
`1hm2d3b4dG5j6r7t8nd90`

String Processing

- Implementation:
 - Define an empty list for the result
 - We return the result by changing from list to string

```
def pwd1(string):  
    result = []  
  
    return "".join(result)
```

String Processing

- Need to keep a counter for the digits

```
def pwd1(string):  
    result = []  
    number = 1
```

String Processing

- Now go through the string with a for statement
- Create the list that will be returned converted into a string

```
def pwd1(string):  
    result = []  
    number = 1  
    for character in string:  
  
        #append to result here  
  
    return "".join(result)
```

String Processing

- We either append the letter from the string or we append the current integer, of course cast into a string

```
def pwd1(string):  
    result = []  
    number = 1  
    for character in string:  
        if character not in "aeiouAEIOU":  
            result.append(character)  
        else:  
            result.append(str(number))  
            number = (number+1)%10  
    return "".join(result)
```

String Processing

- Argot
 - A variation of a language that is not understandable to others
 - E.g. Lufardo – an argot from Buenos Aires that uses words from Italian dialects
 - Invented originally to prevent guards from understanding the inmates
 - Some words are just based on changing words
 - vesre - al revés (backwards)
 - chochamu - vesre for muchacho (chap)
 - lorca - vesre for calor (heat)

String Processing

- Argot
 - Pig Latin
 - Children's language that uses a scheme to change English words
 - Understandable to practitioners, but not to those untrained

String Processing

- Argot:
 - Efe-speech
 - A simple argot from Northern Argentina no longer in use
 - Take a word: “muchacho”
 - Replace each vowel with a vowel-f-vowel combination
 - “Muchacho” becomes Mufuchafachofo
 - “Aires” becomes “Afaifrefes”

String Processing

- Implementing efe-speech
 - Walk through the string, modifying the result list

```
def efe(string):  
    result = []  
    for character in string:  
        result.append(SOMETHING)  
    return "".join(result)
```

String Processing

- We need to be careful about capital letters
 - We can use the string method lower
 - Which you find with a [www-search](#)

```
def efe(string):
    result = [ ]
    for character in string:
        elif character in "AEIOU":
            result.append(character+'f'+character.lower())
    return "".join(result)
```

String Processing

```
def efe(string):
    result = [ ]
    for character in string:
        if character in "aeiou":
            result.append(character+'f'+character)
        elif character in "AEIOU":
            result.append(character+'f'+character.lower())
        else:
            result.append(character)
    return "".join(result)
```

String Processing

```
>>> efe("Alejandria")  
'Afafejafandrifa'  
>>> |
```