


Manipulating Strings

Thomas Schwarz, SJ

How to change strings

- Strings are immutable
 - ```
my_string = "hello world"
```

```
my_string[3] = "p" 
```

    - fails
- Alternative:
  - Make strings into lists
  - Change the list
  - Convert the list into a string

# Strings to Lists

- Use the list command

```
>>> my_string = "Maharashtra"
>>> list(my_string)
['M', 'a', 'h', 'a', 'r', 'a', 's', 'h', 't', 'r', 'a']
>>> |
```

# Changing the string

- E.g. a simple password generator scheme
  - (Pretty insecure, just a tad safer than using dictionary words)
  - Take a common word
    - Change all 'a' to 'e', all 'e' to 'i', 'i' to 'o', 'o' to 'u', and 'u' to 'a'

# Changing the String

- A solution based on indices

```
def change_list(my_list):
 for i in range(len(my_list)):
 if my_list[i] == 'a':
 my_list[i] = 'e'
 elif my_list[i] == 'e':
 my_list[i] = 'i'
 elif my_list[i] == 'i':
 my_list[i] = 'o'
 elif my_list[i] == 'o':
 my_list[i] = 'u'
 elif my_list[i] == 'u':
 my_list[i] = 'a'
```

# Changing the String

- A solution based on indices

```
def change_list(my_list):
 for i in range(len(my_list)):
 if my_list[i] == 'a':
 my_list[i] = 'e'
 if my_list[i] == 'e':
 my_list[i] = 'i'
 if my_list[i] == 'i':
 my_list[i] = 'o'
 if my_list[i] == 'o':
 my_list[i] = 'u'
 if my_list[i] == 'u':
 my_list[i] = 'a'
```

- Can you spot the error in this version?

# Changing the String

- A solution based on our list-changing paradigm

```
def change_list_1(my_list):
 results = []
 for element in results:
 if element == 'a':
 results.append('e')
 elif element == 'e':
 results.append('i')
 elif element == 'i':
 results.append('o')
 elif element == 'o':
 results.append('u')
 elif element == 'u':
 results.append('a')
 else:
 results.append(element)
 return results
```

# Changing the String

- This turns out to be 100 times faster



# From list to string

- 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 “Afaiirefes”

# 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")
'AfaLefejafandrifiafa'
>>> |
```