1 Python for Data Analysis and Numerical Analysis 101

1.1 Homework

1.1.1 Instructor: Evelyn J. Boettcher, DiDacTex, LLC

1.1.2 Week 1: Lecture 2

1.2 1 Lists

- Create an empty list. Append 4 strings to the list. Then pop one item off the end of the list.

1.3 2 Dictionaries

- Create a dictionary using with a zip and two lists
- Add to this dictionary using the key “HW2” with value “Done”
- Define a dictionary using both string literals and variables containing strings.

1.4 3 Strings

- Use a literal to create a string containing:
  - a single quote,
  - a double quote,
  - both a single and double quote.

1.5 4 Strings Multi-line

- Write a string literal that spans multiple lines.

1.6 5 `join`

- Use the string `join` operation to create a string that contains a colon as a separator.

1.7 6 String Format.

- Use string formatting to produce a string containing your last and first names, separated by a comma.

1.8 Solutions

Solution ##1 :

```
In [25]: a = []
In [26]: a.append('aaa')
In [27]: a.append('bbb')
In [28]: a.append('ccc')
In [29]: a.append('ddd')
In [30]: print a
['aaa', 'bbb', 'ccc', 'ddd']
In [31]: a.pop()
Out[31]: 'ddd'
```

Solution ##2 :

```
first = 'Dave'
last = 'Bowie'
name_dict = {first: last, 'Elvis': 'Presley'}
print( name_dict)
#      {'Dave': 'Bowie', 'Elvis': 'Presley'}
```

Solution ##3 :

```
quotes = "Some 'quoted' text."
more_quotes = 'Some "quoted" text.'
even_more_quotes = 'Some "quoted" \'extra\' text.'
```

Solution ## 4: - Write a string literal that spans multiple lines. Solution:

```
my_str= "This string\
spans several lines\
because it is a little long.\
"
```

Solution ##5 - Use the string `join` operation to create a string that contains a colon as a separator.

Solution:

```
content = []
content.append('finch')
content.append('sparrow')
content.append('thrush')
content.append('jay')
new_str = ':'.join(content)
print( new_str)
#      finch:sparrow:thrush:jay
```

Solution 6

- Use string formatting to produce a string containing your last and first names, separated by a comma.

Solution:

```
first = 'Dave'
last = 'Bowie'
full = '%s, %s' % (last, first, )
print( full)
#       Bowie, Dave
```