Introduction to Python

Strings

Topics

1) Strings Concatenation
2) Indexing and Slicing
3) f-Strings
4) Escape Sequences
5) Multiline Strings

String

Strings in Python are created with single or double quotes.

```
In [17]: message = ‘what do you like?’
         response = ‘spam'
In [18]: len(response)
Out [18]: 4

In [19]: # Make uppercase. See also str.lower()
         response.upper()
Out [19]: 'SPAM’
```

String Concatenation

```
In [21]: # concatenation with +
         message + response
Out [21]: ‘what do you like?spam’

In [22]: # multiplication is multiple concatenation
         5 * response
Out [22]: ‘spamspamspamspamspam’
```

String Indexing

```
In [23]: message = "what do you like?"


In [24]: message[0]
Out [24]: 'w'

In [24]: # negative indices wraps around the end
         message[-1]   # last character
Out [24]: '?'
```

String Indexing and Slicing

```
In [23]: message = "what do you like?"


In [24]: # Access individual characters (zero-based indexing)
         message[0]
Out [24]: 'w'


In [25]:  message[0:4] # up to but not including index 4
Out [25]: 'what'


In [25]:  message[0:7:2] # step size of 2
Out [25]: 'wa o'
```

String Indexing and Slicing

```
In [26]: message = ”python"
In [26]: # default start index is 0
          message[:4]
Out [26]: ‘pyth’

In [27]: # default end index is length of string
          message[4:]
Out [27]: ‘on’

In [28]:  message[:]  # default 0 to end of string
Out [28]: ‘python’
```

String Indexing and Slicing

```
In [26]: message = ”python"
In [24]: # negative indices wraps around the end
         message[-1]   # last character
Out [24]: ‘n’

In [25]: # all except the last character
         message[:-1]
Out [25]: ‘pytho’

In [25]: # negative step size traverses backwards
         message[::-1]
Out [25]: ‘nohtyp’
```

f-Strings

f-Strings is the new way to format strings in Python. (v 3.6)

```
In [26]: name = “Mike”
         gpa = 3.2
         f_str = f“I am {name} with a {gpa} gpa.”
         print(f_str)

Out [26]: ‘I am Mike with a 3.2 gpa.’
```

f-Strings Precision

```
In [26]: import math
         x = math.pi
         print(f“{x}”)
         print(f“{x:.2f}”)
         print(f“{x:.3f}”)

3.141592653589793
3.14
3.142
```

str()

The function str() can be construct string objects from integer or float literals.

```
In [1]: y = str(2)     # y will be '2'
        z = str(3.0)   # z will be '3.0'
```

Special Characters

It is not valid syntax to have a single quote inside of a single quoted string.

```
In [1]: 'that's not legal'
  File "<ipython-input-7-2762381d46b7>", line 1
    'that's not legal'
          ^
SyntaxError: invalid syntax
```

Instead, we can use double quotes outside the string.

```
In [2]: print("It's legal to do this.", 'And he said, "This is ok."')
It's legal to do this. And he said, "this is ok."
```

Escape Sequence

Escape sequence is a special sequence of characters used to represent certain special characters in a string.

`\n`	new line character
`\"`	double quote
`\'`	single quote
`\\`	backslash character
`\t`	tab

Escape Sequence

What is the output?

```
In [11]: print(“How \tmany \‘lines\’\n are\n shown\n \“here\”?”)
```

```
How     many ‘lines’
are
shown
“here”?
```

Multiline String

To span multiple lines, put three single quotes or three double quotes around the string instead of one. The string can then span as many lines as you want:

```
In [1]: '''three
        lines
        of
        output'''


Out[1]: 'three\nlines \nof output'
```

Notice that the string Python creates contains a \n sequence everywhere our input started a new line. Each newline is a character in the string.

Multiline strings are often used in documentation strings as we will see later.

References

1) Vanderplas, Jake, A Whirlwind Tour of Python, O'reilly Media.

This book is completely free and can be downloaded online at O'reilly's site.

2) Paul Gries, Jennifer Campbell, Jason Montojo, Practical Programming, The Pragmatic Bookself. 2017.