Slicing Strings

M	o	n	t	y		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11

We can also take a look at any continuous section of a string using a colon operator

- s = ‘Monty Python’
  print(s[0:4])
  Mont

The second is one beyond the end of the slice - “up to but not including”

- print(s[6:7])
  P

If the second number is beyond the end of the string, it stops at the end

- print(s[6:20])
  Python

If we leave off the first number or the last number of the slice, it is assumed to be the beginning or end of the string respectively

- print(s[:2])
  Mo
  - print(s[8:])
    thon
    - print(s[:])
      Monty Python

String Concatenation

When the + operator is applied to strings, it means concatenation

- A = ‘Hello’
  B = A + ‘There’
  print(b)
  HelloThere
  - C = A + ‘ ‘ + ‘There
    Hello There

Using in as a logical operator

The in keyword can also be used to check to see if one string is “in” another string

- fruit = ‘banana’
  ‘n’ in fruit
  True
    - ‘m’ in fruit
      False
        - ‘nan’ in fruit
          True

The in expression is a logical expression that returns True or False and can be used in an if statement

- if ‘a’ in fruit:
      print(‘Found it!’)
  Found it!

String Comparison

-
  ```
  If word == ‘banana’:
      print(‘All right, bananas.’)
  ```

-
  ```
  If word < ‘banana’:
      print(‘Your word,’ + word + ‘ , comes before banana.’)
  elif word > ‘banana’:
      print(‘Your word,’ + word + ‘ , comes after banana.’)
  else:
      print(‘All right, bananas.’)
  ```

The letter case and special characters can throw off string comparison, so be careful.

String Library

Python has a number of string functions which are in the string library.

These functions are already built-in to every string - we invoke them by appending the function to the string variable.

- greet = ‘Hello Bob’
  zap = greet.lower()
  print(zap)
  hello bob
    - print(greet)
      Hello Bob

These functions do not modify the original string, instead they return a new string that has been altered.

- print(‘Hi There’.lower())
  hi there
    - stuff = ‘Hello world’
      type(stuff)
      <class ‘str’>
      dir(stuff)
      ‘capitalize’, ‘casefold’, ‘center’, etc., etc., etc.

Below is the link to the documentation on Python 3’s string methods.

- https://docs.python.org/3/library/stdtypes.html#string-methods

str.capitalize()	str.replace(old, new[, count])
str.center(width[, filichar])	str.lower()
str.endswith(suffix[, start[, end]])	
str.rstrip([chars])	
str.find(sub[, start[, end]])	str.strip([chars])
str.lstrip([chars])	str.upper()

Searching a String

b	a	n	a	n	a
0	1	2	3	4	5

We use the find() function to search for a substring within another string.

find() finds the first occurrence of the substring.

-
  ```
  fruit = ‘banana’
  pos = fruit.find(‘na’)
  print(pos)
  2
  ```

If the substring is not found find() returns -1.

-
  ```
  aa = fruit.find(‘z’)
  print(aa)
  -1
  ```

Remember that string position starts at zero.

Making everything UPPERCASE

You can make a copy of a string in lower() or upper() case.

- greet = ‘Hello Bob’
  nnn = greet.upper()
  print(nnn)
  HELLO BOB
    - www = greeting.lower()
      print(www)
      hello bob

Often when we’re searching for a string using find() we first convert the string to lowercase so we can search a string regardless of case.

Search and Replace

The replace() function is like a “search & replace” operation in a word processor.

-
  ```
  greet = ‘Hello Bob’
  nstr = greet.replace(‘Bob’, ‘Jane’)
  print(nstr)
  Hello Jane
  ```

It replaces all occurrences of the searched string with the replacement string.

-
  ```
  mstr = greet.replace(‘o’, ‘X’)
  print(mstr)
  HellX BXb
  ```

Stripping Whitespace

Sometimes we want to take a string and remove the whitespace at the beginning and/or the end.

- greet = ‘   Hello Bob   ‘

The lstrip() and rstrip() remove the whitespace at the left or right.

- print(greet.lstrip())
  ‘Hello Bob   ‘
    - print(greet.rstrip())
      ‘   Hello Bob’

strip() removes both beginning and ending whitespace.

- print(greet.strip())
  ‘Hello Bob’

Prefixes

```
line = ‘Please have a nice day’

        line.startswith(‘Please’)
        True
                line.startswith(‘p’)
                False
```

Parsing and Extracting

```
From stephen.marquard@utc.ac.za  Sat Jan    5 09:14:16 2008

data = ‘From stephen.marquard@utc.ac.za Sat Jan    5 09:14:16 2008’
atpos = data.find(‘@’)
print(atpos)
21

sppos = data.find(‘ ‘, atpos)
print(sppos)
31

host = data[atpos+1 : sppos]
print(host)
uts.ac.za
```

Two Kinds of Strings

Python 3.5.1

```
type(X)
<class ‘str’>
Y = u’0|abc’
type(Y)
<class ‘str’>
```

Python 2.7.10

```
 X = ‘0|abc’
type(X)
<class ‘str’>
Y = u’0|abc’
type(Y)
<class ‘unicode’>
```

All strings in Python 3 are Unicode

The u before the string is a unicode indicator