THOMAS MORE

PYTHON

YT0744

MANIPULATING STRINGS

Johan Van Bauwel
E-mail: johan.vanbauwel@thomasmore.be
Office: A114

EMSYS EMBEDDED SYSTEMS TECHNOLOGY @THOMAS MORE

MEMBER OF ASSOCIATIE KU LEUVEN

MANIPULATING STRINGS

- You already know string concatenation using the + operator, but you can do much more than that.
- You can:
  - extract partial strings from string values
  - add or remove spacing
  - convert letters to lowercase or uppercase
  - check that strings are formatted correctly
  - write Python code to access the clipboard for copying and pasting text

WORKING WITH STRINGS

- String in Python begin and end with a single quote. But then how can you use a quote inside a string? Typing 'That is Alice's cat.' won't work, because Python thinks the string ends after Alice and the rest (s cat.') is invalid Python code. Fortunately, there are ways to solve this: double quotes and escape characters
- **Double Quotes**

  Strings can also begin and end with double quotes. Python knows the string started with a double quote so it will see a single quote as part of the string

  "That is Alice's cat." is valid python code

ESCAPE CHARACTERS

- An **escape character** lets you use characters that are otherwise impossible to put into a string. An escape character consists of a backslash (\) followed by the character you want to add to the string.

  'That is Alice\'s cat.' is valid python code

Escape character	Prints as
\'	Single quote
\"	Double quote
\t	Tab
\n	Newline (line break)
\\	backslash

- You can place an r before the beginning quotation mark of a string to make it a raw string. **A raw string completely ignores all escape characters** and prints any backslash that appears in the string.
- i.e. print(r'That is Alice\'s cat.') -> That is Alice\'s cat.

MULTILINE STRING AND COMMENTS

- A **multiline string** in Python begins and ends with either **three single quotes or three double quotes**. Any quotes, tabs, or newlines in between the “triple quotes” are considered part of the string.

  i.e. *print('''Dear Alice,*

  Eve's cat has been arrested for catnapping, cat burglary, and extortion.''')

  This code will print as:

  Dear Alice,

  Eve's cat has been arrested for catnapping, cat burglary, and extortion.

- A multiline string is often used for comments that span multiple lines.

  i.e. *'''‘This is a*
  multiline comment'''‘

  This is valid python code that will be ignored when executed.

INDEXING AND SLICING STRINGS

- Strings use indexes and slices the same way lists do. The space and exclamation point are included in the character count, so 'Hello world!' is 12 characters long, from H at index 0 to ! at index 11.

  i.e. *>>> spam = 'Hello world!'*

  >>> spam[0]

  'H'

  >>> spam[4]

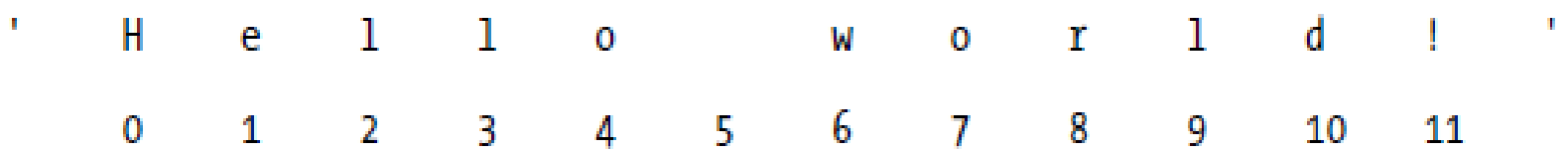
  'o'

  >>> spam[-1]

  '!'

  >>> spam[0:5]

  'Hello'



THE ‘IN’ AND ‘NOT IN’ OPERATORS

- The in and not in operators can be used with strings just like with list values. An expression with two strings joined using in or not in will evaluate to a Boolean True or False. These expressions test whether the first string (the exact string, case sensitive) can be found in the second string.

  i.e. *>>> 'Hello' in 'Hello World'*

  True

  >>> 'Hello' in 'Hello'

  True

  >>> 'HELLO' in 'Hello World'

  False

  >>> '' in 'spam'

  True

  >>> 'cats' not in 'cats and dogs'

  False

THE UPPER(), LOWER(), ISUPPER(), AND ISLOWER() STRING METHODS

- The upper() and lower() string methods return a new string where all the letters in the original string have been converted to uppercase or lowercase, respectively.

```
>>> spam = 'Hello world!‘
>>> spam = spam.upper()
>>> spam
'HELLO WORLD!‘
>>> spam = spam.lower()
>>> spam
'hello world!‘
```

- The isupper() and islower() methods will return a Boolean True value if the string has at least one letter and all the letters are uppercase or lowercase, respectively.

```
>>> 'HELLO'.isupper()
True
>>> 'abc12345'.islower()
True
>>> '12345'.islower()
False
>>> '12345'.isupper()
False
```

THE ISX STRING METHODS

- Along with islower() and isupper(), there are several string methods that have names beginning with the word is. These methods return a Boolean value that describes the nature of the string.
  - ***isalpha()*** *returns True if the string consists* ***only of letters and is not blank****.*
  - ***isalnum()*** *returns True if the string consists* ***only of letters and numbers and is not blank****.*
  - ***isdecimal()*** *returns True if the string consists* ***only of numeric characters and is not blank****.*
  - ***isspace()*** *returns True if the string consists* ***only of spaces, tabs, and newlines and is not blank****.*
  - ***istitle()*** *returns True if the string consists* ***only of words that begin with an uppercase letter followed by only lowercase letters****.*

STARTSWITH() AND ENDSWITH() STRING METHODS

- The **startswith()** and **endswith()** methods return True if the string value they are called on begins or ends (respectively) with the string passed to the method; otherwise, they return False.

```
>>> 'Hello world!'.startswith('Hello')
True
>>> 'Hello world!'.endswith('world!')
True
>>> 'abc123'.startswith('abcdef')
False
>>> 'abc123'.endswith('12')
False
>>> 'Hello world!'.startswith('Hello world!')
True
>>> 'Hello world!'.endswith('Hello world!')
True
```

JOIN() STRING METHODS

- The join() method is useful when you have a list of strings that need to be joined together into a single string value. The join() method is called on a string, gets passed a list of strings, and returns a string. The returned string is the concatenation of each string in the passed-in list.

  >>> ', '.join(['cats', 'rats', 'bats'])

  'cats, rats, bats'

  >>> ' '.join(['My', 'name', 'is', 'Simon'])

  'My name is Simon'

  >>> 'ABC'.join(['My', 'name', 'is', 'Simon'])

  'MyABCnameABCisABCSimon'

- Notice that the string join() calls on is inserted between each string of the list argument. For example, when join(['cats', 'rats', 'bats']) is called on the ', ' string, the returned string is 'cats, rats, bats'.

SPLIT() STRING METHODS

- The split() method does the opposite of the join() method: It's called on a string value and returns a list of strings.

```
>>> 'My name is Simon'.split()
['My', 'name', 'is', 'Simon']
```

- By default, the string 'My name is Simon' is split wherever whitespace characters such as the space, tab, or newline characters are found. These whitespace characters are not included in the strings in the returned list. You can pass a delimiter string to the split() method to specify a different string to split upon.

```
>>> 'MyABCnameABCisABCSimon'.split('ABC')
['My', 'name', 'is', 'Simon']
>>> 'My name is Simon'.split('m')
['My na', 'e is Si', 'on']
```

JUSTIFYING TEXT

- The **rjust()**, **ljust()** and **center()** string methods return a padded version of the string they are called on.

  >>> 'Hello'.rjust(20, '')*

  *'**********Hello' ← 15 '*' + 5 characters from 'Hello' = 20 characters*

  >>> 'Hello'.ljust(20, '-')

  'Hello----------'

  >>> 'Hello'.center(20, '=')

  '=======Hello========'

- *The second argument is optional.* ***By default these methods will uses spaces****.*

  >>> 'Hello World'.rjust(20)

  ' Hello World'

REMOVING WHITESPACE WITH STRIP(), RSTRIP(), AND LSTRIP()

- Sometimes you may want to strip off characters from the left side, right side, or both sides of a string.

```
>>> spam = ‘      Hello World      '
>>> spam.strip()
'Hello World‘
```

- *Optionally, a string argument will specify which characters on the ends should be stripped.*

```
>>> spam = 'SpamSpamBaconSpamEggsSpamSpam'
>>> spam.strip('ampS')
'BaconSpamEggs‘
```

- *Passing strip() the argument 'ampS' will tell it to strip occurences of a, m, p, and capital S from the ends of the string stored in spam. The order of the characters in the string passed to strip() does not matter: strip('ampS') will do the same thing as strip('mapS') or strip('Spam').*

COPYING AND PASTING STRINGS WITH THE PYPERCLIP MODULE

- The pip tool
  - *The pip tool is used to install modules.*
  - *The executable file for the pip tool is called pip on Windows and pip3 on OS X and Linux. On Windows, you can find pip at C:\Python34\Scripts\pip.exe. On OS X, it is in /Library/Frameworks/Python.framework/Versions/3.4/bin/pip3. On Linux, it is in /usr/bin/pip3.*
  - *Pip comes automatically installed with Python 3.4 on Windows and OS X.*
  - *You must install it separately on Linux. To install pip3 on Ubuntu or Debian Linux, open a new Terminal window and enter **sudo apt-get install python3-pip**. To install pip3 on Fedora Linux, enter **sudo yum install python3-pip** into a Terminal window.*
  - *The pip tool is meant to be run from the command line: You pass it the command install followed by the name of the module you want to install.*
  - *i.e. Windows: pip install pyperclip*
    Linux / Osx: sudo pip3 install pyperclip

COPYING AND PASTING STRINGS WITH THE PYPERCLIP MODULE CONT.

- After installing you have to import the module.

  >>> import pyperclip
- **Copying to the clipboard**

  >>> pyperclip.copy('Hello world!')
- **Pasting from the Clipboard**

  >>> pyperclip.paste()
  'Hello world!‘

EXAMPLE QUESTIONS 1/2

- What are escape characters?
- What do the \n and \t escape characters represent?
- How can you put a \ backslash character in a string?
- The string value "Howl's Moving Castle" is a valid string. Why isn't it a problem that the single quote character in the word Howl's isn't escaped?
- What do the following expressions evaluate to?
  - 'Hello world!'[1]
  - 'Hello world!'[0:5]
  - 'Hello world!'[:5]
  - 'Hello world!'[3:]

EXAMPLE QUESTIONS 2/2

- What do the following expressions evaluate to?
  - 'Hello'.upper()
  - 'Hello'.upper().isupper()
  - 'Hello'.upper().lower()
- What do the following expressions evaluate to?
  - 'Remember, remember, the fifth of November.'.split()
  - '-'.join('There can be only one.'.split())
- What string methods can you use to right-justify, left-justify, and center a string?
- How can you trim whitespace characters from the beginning or end of a string?

Q&A

- Questions?