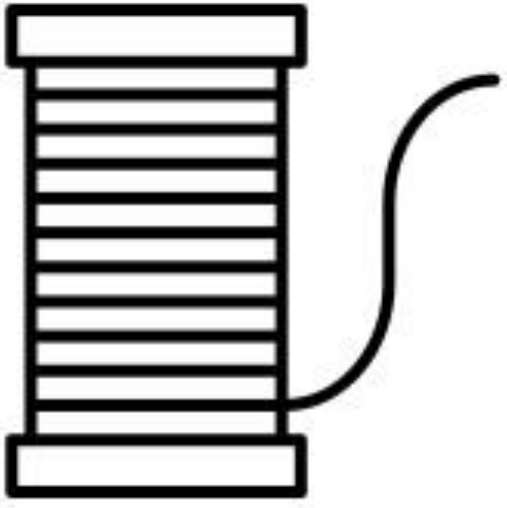
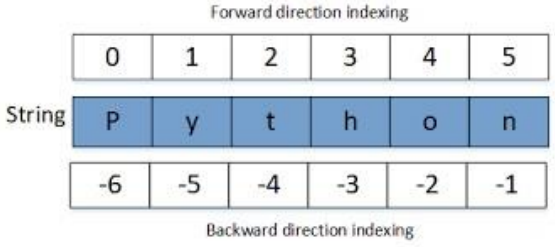
Chapter 8
String Operations

Outline

- String Indexing
- String Slicing
- String Methods
  - Testing
  - Modifying
  - Searching / Replacing
  - Splitting / tokenizing
- Working with CSV Files



String Indexing



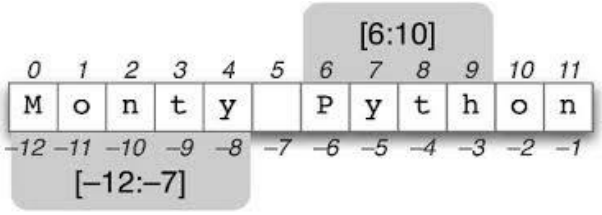
- Strings are sequences of characters
  - Behave as lists (tuples) where each element is a character: letter, digit or a special character (space, comma, …)
  - Iterate over a string character by character

```
>>> for ch in full_name:
        print(ch)
```

0	1	2	3	4	5	6	7	8	9	10
S	m	i	t	h	,		J	o	h	n
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- **Lect8_Strings.py**
- Indexing
  - Every string character can be directly access through an index
  - Just like lists (tuples) indices go from 0 to length of string – 1
    - **`len`** function determines the length of a string
    - Like tuples strings are immutable
  - **`IndexError`** exception raised if you attempt to access string character beyond the end of a string

String Concatenating & Slicing



- Concatenation
  - One of the most common string operation using the + operator; often used to dynamically concatenate characters
- Slicing
  - Opposite of concatenation; break strings into pieces
  - Just like with lists need to provide the **`start`** and **`end`** of the slice
    - Determine the index of the space character
    - Slice from the end of the string to one before space index
  - Many varieties of slices
    - Omitting **`start`** will start slicing from 0
    - Omitting **`end`** will slice to the end of string
    - Omitting both will make a copy of a string

0	1	2	3	4	5	6	7	8	9	10
S	m	i	t	h	,		J	o	h	n
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

String Methods – Modifying & Testing

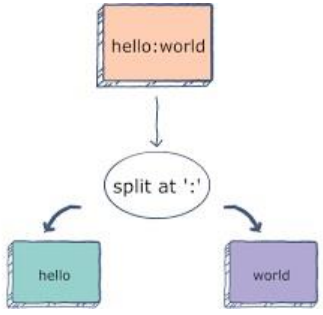
- Changing all letters to lowercase / uppercase
  - **`lower()`**, **`upper()`**
  - Used to make case-insensitive comparisons
- Stripping whitespace and escape characters
  - **`lstrip()`**, **`rstrip()`**
- Testing if all letters are lowercase / uppercase
  - **`islower()`**, **`isupper()`**
- Testing for letters, digits, spaces, etc..
  - **`isalnum()`**, **`isalpha()`**, **`isdigit()`**, **`isspace()`**
  - Useful for testing validity of passwords

String Methods – Searching

- Searching for substrings
  - **`find()`** and **`index()`**
  - Both have **`substr`** to look for, **`start`** / **`end`** optional indices
  - If substring is not found **`find()`** returns **`-1`**, **`index()`** error
  - If substring found, returns index of first encounter
    - Comma would be found at index 5
    - Space would be found at index 6
- Standard **`replace()`** method
- Search for substrings at the beginning / end of string
  - **`startswith()`**, **`endswith()`**

0	1	2	3	4	5	6	7	8	9	10
S	m	i	t	h	,		J	o	h	n
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

String Methods – Splitting



- More complex strings consists of
  - Tokens: words and numbers mostly
  - Delimiters: special characters like commas, spaces, hyphens, colons, semicolons, …
- Tokenizing – a process of breaking strings into tokens
  - Splitting full name into first and last
    - **`Last, first`** – delimiter is **`','`**
  - Breaking date into month, day and year
    - **`mm/dd/yyyy`** – delimiters are forward slashes **`/`**
  - Splitting time into hour, minute and second
    - **`hh:mm:ss`** – delimiters are colons **`:`**
  - Breaking address into city, state and zip
  - Splitting a row from CSV files into attribute columns

Working with CSV Files

- CSV files for distributing tabular data
  - Simple CSV files
    - Clean data, simple rows and columns
  - Real CSV files:
    - Messy data, commas as part of data enclosed in double-quotes, special and escape characters, missing values, multiple formats, several tables, some including headings, some without, different dialects, …
    - Much more difficult to simply split the lines based on comma delimiters
- **`csv`** package
  - Not perfect, handles much of the standard "messiness"
  - **`csv.reader`**:
    - Reads data from a CSV file into an object that can be iterated over, row by row
    - Each read row is list, each list element is a value of a particular column
  - **`csv.writer`**:
    - Writes data into a CSV file, typically row by row
    - Each row to write is a list, elements are separated by commas when written
  - **Lect8_Loan_CSV.py**

Summary

- Defined strings as sequences of characters
  - Strings are immutable like tuples
- String characters indexed like lists/tuples
- Basic operations
  - Concatenation: creates new strings by appending strings together
  - Slicing: extracts substrings from strings
- String methods
  - Modifying (creates new strings) and testing
  - Searching, replacing and splitting
- Described reading/writing from/to CSV files with standard **`csv`** package