String Processing

CS 1111
Introduction to Programming
Spring 2019

[The Coder’s Apprentice, §10]
# Collections

## Ordered, Dup allow
- List
- Range
- String
- Tuple

- \textit{collection}[\textit{index}]
- Access an element’s value via index
- Index is int, starting at 0

## Unordered, No Dup
- Dict

- \textit{collection}[\textit{key}]
- Access an element’s value via key
- Key is primitive type, unique
Strings

- Sequence of characters (letters, numbers, punctuation marks, spaces, ...)

- String literals = sequence of characters enclosed by quotations
  - print(“Hello World!”)

- Quotations inside quotations
  - “Python’s fun!”

- Must match quotations
  - Single with single, double with double
Length of Strings

• Length of a string = the number of characters in a string

  • len("Hello World!")  # length = 12
  • len("")  # length of empty string = 0
String Concatenation ("+")

- Attach string to another string

```python
firstName = "Thomas"
lastName = "Jefferson"
name = firstName + " " + lastName
print("Name is ", name)  # Name is Thomas Jefferson
```
String Repetition ("\(*\)"")

• Produce a string that is composed of repeated characters

\[
dashes = “-” \ast 10 \quad \# \text{dashes} = “----------”
\]

Note: results in string
String Conversion

- Convert numbers to strings

```
average_grades = 85
result = "Average Test1 grade is " + str(average_grades)
print(result)
```

What happens if `average_grades` is not casted

- Convert strings to numbers

```
prod_id = int("149")
price = float("85.45")
print(prod_id)
print(price)
```
Special Characters

• Escape character
  
  print(“Python is an \“interpreted\” language”)

• New line character
  
  print(“Jake
John
Jane”)
String Equality

string1 equals to string2 if and only if
• They are of the same length, and
• They contain the exact same sequence of characters, and case sensitive

```python
name1 = "Thomas Jefferson"
name2 = "Thomas Jefferson"
name3 = "Thomas jefferson"
name4 = "Thomas Jenkinson"

# only name1 = name2
```
Slicing

Refer to a range of values

\[ s[start : stop] \]
- Slice from start index (inclusive) to stop index (exclusive)

\[ s[start : ] \]
- Slice from start index (inclusive) to the end of the collection

\[ s[ : stop] \]
- Slice from the beginning of the collection to the stop index (exclusive)

\[ s[start : stop : step] \]
- Slice from start index (inclusive) to stop index (exclusive), skip step-1 then grab
in Operator

substring in string

- Returns True if substring exists in string
- False, otherwise

name = “Thomas Jefferson”
print(“Jeff” in name)     # True
String Testing Methods: \texttt{isdigit( )}, \texttt{isspace( )}

\texttt{isdigit( )}

- Returns True if the string \( s \) consists of only digits and contains at least one character; and False otherwise

\begin{verbatim}
name = "Thomas Jefferson"
print(name.isdigit())  # False, contains non-digits
\end{verbatim}

\texttt{isspace( )}

- Returns True if the string \( s \) consists of only white space characters (blank, newline, tab) and contains at least one character; and False otherwise

\begin{verbatim}
name = "Thomas Jefferson"
print(name.isspace())  # False, contains non-white space
\end{verbatim}
**String Testing Methods:**

**isalnum( )**, **isalpha( )**

**isalnum( )**
- Returns True if the string \( s \) consists of only letters or digits and contains at least one character; and False otherwise

```python
name = “Thomas Jefferson”
print(name.isalnum())  # False, contains space
```

**isalpha( )**
- Returns True if the string \( s \) consists of only letters and contains at least one character; and False otherwise

```python
name = “Thomas Jefferson”
print(name.isalpha())  # False, contains space
```
String Testing Methods: 
islower(), isupper()

islower()

• Returns True if the characters in the string are lowercase and the string contains at least one character; and False otherwise

name = “Thomas Jefferson”
print(name.islower())  # False, contains uppercase

isupper()

• Returns True if the characters in the string are uppercase and the string contains at least one character; and False otherwise

name = “Thomas Jefferson”
print(name.isupper())  # False, contains lowercase
String Modification Methods: lower( ), upper( )

lower( )
• Convert a string to lowercase

```python
name = "Thomas Jefferson"
print(name.lower())  # thomas jefferson
```

upper( )
• Convert a string to uppercase

```python
name = "Thomas Jefferson"
print(name.upper())  # THOMAS JEFFERSON
```
String Modification Methods: \texttt{strip()}, \texttt{rstrip()}, and \texttt{lstrip()}

\textbf{\texttt{strip}(\texttt{char})}

• Returns a copy of the string with all instances of \texttt{char} that appear at the beginning and the end of the string removed

\begin{verbatim}
name = “ Thomas Jefferson ”
print(name.strip())
\end{verbatim}

\textbf{\texttt{rstrip}(\texttt{char})}

• Returns a copy of the string with all instances of \texttt{char} that appear at the end of the string removed

\textbf{\texttt{lstrip}(\texttt{char})}

• Returns a copy of the string with all instances of \texttt{char} that appear at the beginning of the string removed

Default: remove whitespace characters
• spaces, newlines (\textbackslash n), and tabs (\textbackslash t)
Other String Methods: `join()`

`join(seq_list)`

- Returns a copy of the string, which is the concatenation of the string with intervening occurrences of `seq_list`.
- `seq_list` must be a list

```python
name = "Thomas Jefferson"
seq_list = ["$", "--", "@"]
print(name.join(seq_list))
# $Thomas Jefferson--Thomas Jefferson@
```
Other String Methods: \texttt{split( )}

\texttt{split(delimiter)}

- Returns a list of all the words in the string, using \textit{delimiter} as the separator
- Default separator: space

\begin{verbatim}
name = "Thomas Jefferson"
print(name.split())    # ['Thomas', 'Jefferson']
\end{verbatim}
Other String Methods: \textbf{count}() \\

\textbf{count}\( (\text{substring}) \)  
- Returns the number of non-overlapping occurrences of \textit{substring} in the string \textit{s} 

\begin{verbatim}
name = "Thomas Jefferson"
print(name.count("f"))  # 2
print(name.count("ff"))  # 1
\end{verbatim}
Search and Replace Methods: \texttt{startswith()} 

\texttt{startswith(substring)}

- Returns True if the string \textit{s} begins with \textit{substring} and False otherwise.

\begin{verbatim}
name = "Thomas Jefferson"
print(name.startswith("Th"))  # True
\end{verbatim}
Search and Replace Methods: `endswith()`

`endswith(substring)`

- Returns True if the string `s` ends with `substring` and False otherwise

```python
name = "Thomas Jefferson"
print(name.endswith("son"))  # True
```
Search and Replace Methods: \texttt{find()} and \texttt{rfind()}

\textbf{find(substring)}

- Returns the lowest index in the string \( s \) where \textit{substring} begins, or \(-1\) if substring is not found

\begin{verbatim}
name = "Thomas Jefferson"
print(name.find("f"))    # 9
print(name.find("cs1111"))    # -1
\end{verbatim}

\textbf{rfind(substring)}

- Returns the highest index in the string \( s \) where \textit{substring} begins, or \(-1\) if substring is not found

\begin{verbatim}
name = "Thomas Jefferson"
print(name.rfind("f"))    # 10
\end{verbatim}
Search and Replace Methods: 
\texttt{index()} 

\texttt{index(substring, \texttt{beg}=0 \texttt{end}=\texttt{len(string)})} 

- Returns the lowest index in the string \( s \) where \texttt{substring} begins, or raises an exception (\texttt{ValueError: substring not found}) if \texttt{substring} is not found 

\texttt{str1 = "Thomas Jefferson"} 
\texttt{str2 = "e"} 
\texttt{print(str1.index(str2))} \quad \# 8 
\texttt{print(str1.index(str2, 9))} \quad \# 11 
\texttt{print(str1.index(str2, 9, 12))} \quad \# 11 
\texttt{print(str1.index(str2, 9, 11))} \quad \# error 

\# from index 9, up to index 11 (not include index 11)
Search and Replace Methods: 
\texttt{\textbf{replace( )}}

\texttt{replace(old, new)}

- Returns a copy of the string with all instances of \textit{old} replaced by \textit{new}

\begin{verbatim}
name = "Thomas Jefferson"
print(name.replace("Thomas","George"))
# George Jefferson
\end{verbatim}
Summary

- Must know (based on exam2 topic list, as of 03/04/2019)
  - substring in string
  - string1 + string2
  - string.strip()
  - string.split()
  - string.split(delimiter)
  - string.find(substring)