Strings

Upsorn Praphamontripong

CS 1111
Introduction to Programming
Spring 2018
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

```python
dashes = "-" * 10  # dashes = "----------"
```

Note: results in string
String Conversion

• Convert numbers to strings

average_grades = 85
result = “Average Test 1 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)
Strings versus Lists

How is a string similar to a list?
• Both can be indexed (because they are ordered)
• Both can be sliced (sub-list and sub-string), using [ : ]
• Both are complex type
• Lists can be changed (mutable), strings cannot be updated (immutable) after assignment

Each character inside a string has an index number

```python
name = “Thomas Jefferson”
first_letter = name[0]  # T
last_letter = name[len(name)-1]  # n
firstname = name[ : 6]  # Thomas
lastname = name[7: ]  # Jefferson
“Jeff” in name  # True
“upsorn” not in name  # True
```
Strings versus Lists (2)

How is a string differ from a list?

• Does not support the operations that change the contents of the list (such as append, remove, insert, reverse, and sort).
Special Characters

Escape character

```python
print("Python is an \"interpreted\" language")
```

New line character

```python
print("Jake\nJohn\nJane\n")
```
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

name1 = “Thomas Jefferson”
name2 = “Thomas Jefferson”
name3 = “Thomas jefferson”
name4 = “Thomas Jenkinson”

# only name1 = name2
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

```python
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

```python
name = "Thomas Jefferson"
print(name.isupper())  # False, contains lowercase
```
String Modification Methods: strip(), rstrip(), and lstrip()

strip(char)
- Returns a copy of the string with all instances of char that appear at the beginning and the end of the string removed.

```python
name = "  Thomas Jefferson  
print(name.strip()) # Thomas Jefferson (no spaces)
```

rstrip(char)
- Returns a copy of the string with all instances of char that appear at the end of the string removed.

lstrip(char)
- Returns a copy of the string with all instances of char that appear at the beginning of the string removed.

Default: remove whitespace characters
- spaces, newlines (\n), and tabs (\t)
Other String Methods: \texttt{join()} \\

\texttt{join(seq\_list)}

- Returns a copy of the string, which is the concatenation of the string with intervening occurrences of \texttt{seq\_list}.
- \texttt{seq\_list} must be a list

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

`split(char)`
- Returns a list of all the words in the string, using `char` as the separator
- Default separator: space

```python
name = "Thomas Jefferson"
print(name.split())  # ['Thomas', 'Jefferson']
```
Other String Methods: \texttt{count( )}

\texttt{count(substring)}

- Returns the number of non-overlapping occurrences of \textit{substring} in the string \texttt{s}

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

`startswith(substring)`

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

```python
name = "Thomas Jefferson"
print(name.startswith("Th"))  # True
```
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:

**find() and rfind()**

**find(substring)**
- Returns the lowest index in the string s where substring begins, or -1 if substring is not found

```python
name = "Thomas Jefferson"
print(name.find("f"))  # 9
print(name.find("cs1111"))  # -1
```

**rfind(substring)**
- Returns the highest index in the string s where substring begins, or -1 if substring is not found

```python
name = "Thomas Jefferson"
print(name.rfind("f"))  # 10
```
Search and Replace Methods:

index( )

index(substring, beg=0 end=len(string))

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

```python
str1 = "Thomas Jefferson"
str2 = "e"
print(str1.index(str2))       # 8
print(str1.index(str2, 9))    # 11
print(str1.index(str2, 9, 12)) # 11
print(str1.index(str2, 9, 11)) # error
    # from index 9, upto index 11 (not include index 11)
```
Search and Replace Methods: `replace()`

`replace(old, new)`

- Returns a copy of the string with all instances of `old` replaced by `new`

  ```python
  name = "Thomas Jefferson"
  print(name.replace("Thomas", "George"))
  # George Jefferson
  ```