

# **Web Application Development using Python**

**Introduction to Programming using Python**

**Prepared by George Khoury**



# Outline

- What are strings?
- Updating strings
- String operations / methods
- String formatting



# Strings

## W2/S2/strings

- A string is a sequence of characters.
- A character is simply a symbol. For example, the English language has 26 characters.
- This conversion of character to a number is called encoding, and the reverse process is decoding. ASCII and Unicode are some of the popular encodings used.

```
my_string = 'Hello, world!'  
print(my_string)
```

# Accessing characters in a string

## W2/S2/strings

- We can access individual characters using **indexing** and a range of characters using **slicing**.
- **The index**
  - Integer value starts from 0.
  - Trying to access a character out of index range will raise an **IndexError**.

```
my_string = 'Hello, world!'  
print(my_string)
```

# Accessing characters in a string

## W2/S2/strings

```
# Accessing string characters in Python
my_string = 'Hello, world!'
print('my_string = ', my_string)
```

```
# first character
print('my_string[0] = ', my_string[0])
```

```
# last character
print('my_string[-1] = ', my_string[-1])
```

```
# slice from 2nd to the 5th character
print('my_string[1:5] = ', my_string[1:5])
```

```
# slice from 6th to the 2nd character
print('my_string[5:-2] = ', my_string[5:-2])
```

# Updating a string

## W2/S2/strings

- Strings are **immutable**.
  - This means that elements of a string cannot be changed once they have been assigned.
  - We can simply reassign different strings to the same name.
  - We can delete the entire string using the **del** keyword.

```
>>> my_string = 'Hello, world!'
>>> my_string[5] = 'a'
TypeError: 'str' object does not support item assignment

>>> my_string = 'Python'
>>> my_string
'Python'
```

# Updating a string

## W2/S2/strings

```
# Accessing string characters in Python
my_string = 'Hello, world!'
print('my_string = ', my_string)
```

```
# first character
print('my_string[0] = ', my_string[0])
```

```
# last character
print('my_string[-1] = ', my_string[-1])
```

```
# slice from 2nd to the 5th character
print('my_string[1:5] = ', my_string[1:5])
```

```
# slice from 6th to the 2nd character
print('my_string[5:-2] = ', my_string[5:-2])
```

# String operations

W2/S2/strings/string\_basics.py

- There are many operations that can be performed with strings which makes it one of the most used data types in Python.
  - String length
  - String concatenation
  - String repetition
  - Iterating through a string
  - Membership test
  - Other built-in functions

The diagram illustrates the execution of the following Python code:

```
print('%d %s cost $%.2f' % (6, 'bananas', 1.74))
```

Annotations in the diagram include:

- A bracket labeled "format string" underlines the string `'%d %s cost $%.2f'`.
- A bracket labeled "values" underlines the tuple `(6, 'bananas', 1.74)`.
- An arrow labeled "modulo operator" points to the `%` symbol between the string and the tuple.
- A long arrow points from the entire code line down to the resulting output: `6 bananas cost $1.74`.
- Four curved arrows at the top show the mapping of values from the tuple to their respective placeholders in the format string: `%d` to `6`, `%s` to `'bananas'`, and `%.2f` to `1.74`.



# String methods

W2/S2/strings/string\_methods.py

- There are numerous methods available with the string object.
- The **format()** method is one of them.
- Some of the commonly used methods are
  - lower(), upper(), capitalize()
  - join(), split()
  - find(), replace()
- Additional methods can be found here.
  - <https://docs.python.org/3/library/stdtypes.html#string-methods>

The diagram illustrates the components of a Python string formatting statement. The code `print('%d %s cost $%.2f' % (6, 'bananas', 1.74))` is shown. Brackets and arrows identify the parts: the string `'%d %s cost $%.2f'` is labeled 'format string'; the tuple `(6, 'bananas', 1.74)` is labeled 'values'; the `%` symbol is labeled 'modulo operator'. A long arrow points from the entire expression to the resulting output: `6 bananas cost $1.74`.

```
print('%d %s cost $%.2f' % (6, 'bananas', 1.74))
```

format string      modulo operator      values

6 bananas cost \$1.74

# String formatting

## W2/S2/strings

- You can read the following resources to learn more about formatting.
- <https://docs.python.org/3/library/string.html#formatstrings>

# Learning Resources

- <https://docs.python.org/3/tutorial/introduction.html#lists>
- <https://docs.python.org/3/tutorial/datastructures.html#more-on-lists>
- <https://docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences>
- <https://docs.python.org/3/tutorial/datastructures.html#sets>
- <https://docs.python.org/3/tutorial/datastructures.html#dictionaries>