# Strings and methods

CS195 - Lecture 3
Instructor:  Dr. V

# Literal strings in python

- single quote
  ```
  'hello world'
  ```
- double quote
  ```
  "hello world"
  ```
- triple quote
  ```
  '''hello world'''
  """hello world"""
  ```

# When to use double-quote vs single-quote?

- General rule – be consistent
  - either stick with single or double-quotes throughout your code, but…

- There are times when one is more appropriate than the other
  - How would you tell python to print the following two literal strings:

```
So I asked him what's going on.
And he said, "nothing is going on."
```

# When to use triple-quotes?

- Triple-quotes are great for
  - including both single and double -quotes inside
  - including multi-line text

```
s = '''So I asked him what's going on.
And he said, "nothing is going on."

That's it. That's my whole story.'''
```

- What do you think you'd use triple-double-quotes for?

# When to use triple-quotes?

- You can use """..."""  to include ''' inside
- You can use '''...''' to include """ inside

```
print( """
    s = '''So I asked him what's going on.
    And he said, "nothing is going on."

    That's it. That's my whole story.""""""  )
```

# Escape characters

- \'  Single Quote (use inside single-quoted string)
- \"  Double Quote (use inside double-quoted string)

- \n  New Line
- \r  Carriage Return

- \t  Tab

- \b  Backspace

- \\  Backslash

```
# what will this print?
print( '\\hello\\world\\\n' )
```

# Concatenating and repeating strings

```
s1 = 'hello'
s2 = 'world'

print( s1 + s2 )

print( (s1 + s2) * 2 )

print( s1 + s2 * 2 )
```

# Changing a string

- strings in python are immutable (cannot be changed)
  - that means you cannot change a given character in a string

- however, this will work:

```
s = 'hello'
s += 'world'
s *= 2
```

# String methods

- `startswith`
- `endswith`

- `isalnum`
- `isalpha`

- `lower`
- `upper`

- `strip`
- `replace`

- `find`
- `split`

# String methods

```
s = 'Hello world!'

# What do you think each of these lines prints?
print( s.startswith('hello') )
print( s.endswith('!') )
print( s.isalnum() )

print( s.lower() )
print( s.upper() )

print( s.strip('!') )
print( s.strip('dlH!e') )
print( '  Hello world!\n'.strip() )
print( '  Hello world!\n'.lstrip() )
print( '  Hello world!\n'.rstrip() )
```

# String methods

```
s = 'Hello world!'

# What do you think each of these lines prints?
print( s.find('Hello') )
print( s.find('hello') )
print( s.find('world') )

print( s.replace('!','?') )
print( s.replace('l','!') )

print( s.split() )
print( s.split('o') )
```

# Methods

- Methods are functions that belong to specific objects
- Note the **syntax** difference between a global function and a method:
  - `s = 'hello world'`
  - `print( s )  # print is a global function`
  - `s.upper()  # upper is a string method`
- If your object, *o*, has methods,
  - list methods via dir(*o*) or autocomplete
  - use a method like this *o.method*(...)

# f-strings

- f-strings are special in that they allow python expressions inside them
- add the character f in front of a literal string to turn it into an f-string
- add python expressions within curly brackets inside the string

```
# what do you think this would print?
name = 'ANDY'
age = 20
print( f"Hello {name.lower()}. You're {age} years older than me!")
print( f"If I was 10, you'd be {age-10} years older than me.")
```

# f-strings

- to indicate a literal brace, use double-braces
- be careful with single vs double quotes, even inside braces

```
# what do you think this will print?
print( f" {{ 'hello'.upper() }} { 'hello'.upper() } " )
```

# f-strings

- You can use f-strings to format numbers
  - specify number of decimal places
  - specify number of padding zeros or spaces

```
# what do you think each of these will print?
print( f"less decimal places { 3.14159 :.2f}" )
print( f"more decimal places { 3.14159 :.10f}" )
print( f"pad with zeros { 33 :05}" )
print( f"pad with spaces { 33 :5}" )
print( f"pad and specify decimal places { 3.14159 :05.2f}" )
print( f"pad and specify decimal places { 3.14159 :5.2f}" )
```

# f-strings

- You can use f-strings to left-justify, right-justify, or to center text

```
# what do you think the statements below will print?
s1, s2, s3 = 'abc', 'defghi', 'jklmnopqrstu'
print(f'|{s1:<20}|{s1:>20}|{s1:^20}|')
print(f'|{s2:<20}|{s2:>20}|{s2:^20}|')
print(f'|{s3:<20}|{s3:>20}|{s3:^20}|')
```

# Assignment 3

- change your chatbot.py code to use the following concepts
  - f-strings
    - use f-strings to control decimal spaces in your floats, and to embed arithmetic operations inline
  - string methods
    - upper
      - capitalize user's name
    - strip
      - remove any spaces in user-entered strings
  - use either newline character \n or triple-quoted strings to print multi-line output
- make sure to comment your code and use correct naming conventions for your variables