APS106

UNIVERSITY OF TORONTO

Strings: Conversions, Indexing, Slicing, Immutability.

Week 5 | Lecture 2 (5.2)

While waiting, open the Jupyter Notebook for today's lecture

Upcoming

- NO lab due this Friday.
- Lab 3 Released Thursday 6:00 pm.
- Reflection 5 Released Friday 6:00 pm.
- Tutorial (in-person AND online) running all week.
- Practical sessions (in-person AND online) running ONLY Friday this week.

if nothing else, write #cleancode

This Week's Content

- **Lecture 5.1**
  - Objects & Strings: Operators and Methods
- **Lecture 5.2**
  - **Strings: Conversions, Indexing, Slicing, and Immutability**
- **Lecture 5.3**
  - Introduction to Object-Oriented Programming and the File Object

Working with Strings

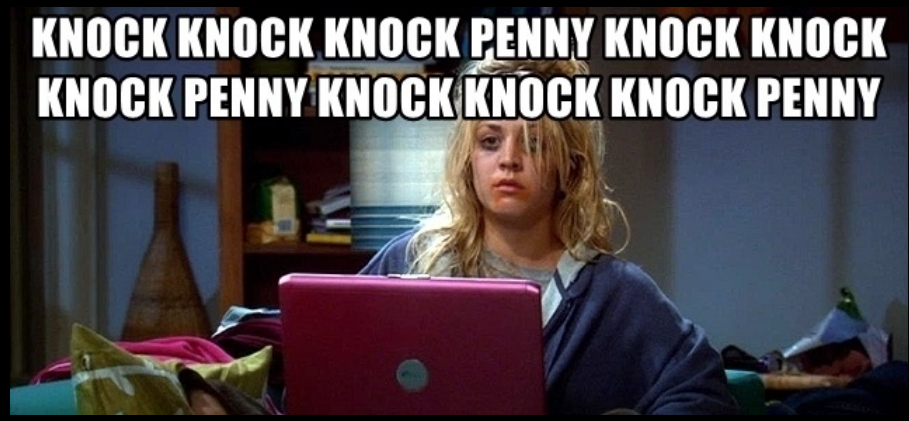
- The string (str) type was briefly introduced in previous weeks
- Let's take our string knowledge to the next level!
  - escape sequences ✓
  - str operations ✓
  - type conversion ✓
  - str indexing and slicing
  - str methods

Consider this…

- Ask the user how many times they would like to see the string "knock knock knock… Penny" repeated, and print it!
- Can you customize the name?

Consider this…

- Ask the user how many times they would like to see the string "knock knock knock… Penny" repeated, and print it!
- Can you customize the name?



Hints for getting started:

- Ask the user for a number of times (think: input function)
  - Remember input function returns a string…
- Repeated string (think: concatenation, * operator might be useful)
- Make the output readable (think: escape characters)

Working with Strings

- The string (str) type was briefly introduced in previous weeks
- Let's take our string knowledge to the next level!
  - escape sequences ✓
  - str operations ✓
  - type conversion ✓
  - str indexing and slicing
  - str methods

String Indexing

- An index is a position within the string
- A particular element of the string is accessed by the index of the element surrounded by square brackets [ ]
- Positive indices count from the left-hand side, beginning with the first character at index 0, the second index 1, and so on…
- Negative indices count from the right-hand side, beginning with the last character at index -1, the second last at index -2, and so on…

```
>>>"Yolo"[0]
'Y'
```

```
>>> x = "Yolo"
>>> x[2]
'l'
```

```
>>> x = "Yolo"
>>> x[-1]
'o'
```

String Indexing

- An index is a position within the string
- A particular element of the string is accessed by the index of the element surrounded by square brackets
- Positive indices count from the left-hand side, beginning with the first character at index 0, the second index 1…
- Negative indices count from the right-hand side, beginning with the last character at index -1, the second last at index -2, and so on…

```
>>> x = "I Love Cats"
```

0	1	2	3	4	5	6	7	8	9	10
I		**L**	**o**	**v**	**e**		**C**	**a**	**t**	**s**
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x[0]
'I'
```

```
>>> x[6]
' '
```

```
>>> x[-11]
'I'
```

String Slicing

- We can extract more than one character (or substring) using slicing
- Uses the syntax [start : finish], where:
  - start is the index where we start the slice
  - finish is the index of **one after** where we end the slice
- When either start or finish are not provided:
  - If start index is missing, it defaults to the beginning
  - If finish index is missing, it defaults to the end

```
>>> x = "Yolo"
>>> x[0:3]
'Yol'
```

```
>>> x = "Yolo"
>>> x[:3]
'Yol'
```

```
>>> x = "Yolo"
>>> x[-2:]
'lo'
```

String Slicing

- We can extract more than one character (or substring) using slicing
- Uses the syntax [start : finish], where:
  - start is the index where we start the slice
  - finish is the index of **one after** where we end the slice
- When either start or finish are not provided:
  - If start index is missing, it defaults to the beginning
  - If finish index is missing, it defaults to the end

```
>>> x = "I Love Cats"
```

0	1	2	3	4	5	6	7	8	9	10
I		**L**	**o**	**v**	**e**		**C**	**a**	**t**	**s**
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x[2:6]
'Love'
```

```
>>> x[-9:-5]
'Love'
```

```
>>> x[-4:]
'Cats'
```

```
>>> x[:]
'I Love Cats'
```

String Length

- To obtain the length of a string you can use the built-in function len
- The len function takes a string as an argument and returns an integer indicating the length of the string
  - Note: This will always be the final index + 1

```
>>> x = "I Love Cats"
```

0	1	2	3	4	5	6	7	8	9	10
I		**L**	**o**	**v**	**e**		**C**	**a**	**t**	**s**
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> len(x)
11
```

Extended Slicing

- We can slice (select) every ***n***th character by providing three arguments
- Uses the syntax [start : finish : step], where:
  - start is the index where we start the slice
  - finish is the index of **one after** where we end the slice
  - step is how much we count by between each character
- When step is not provided, it defaults to 1

```
>>> x = "Yolo"
>>> x[::2]
'Yl'
```

```
>>> x = "Yolo"
>>> x[::]
'Yolo'
```

Extended Slicing

- We can slice (select) every ***n***th character by providing three arguments
- Uses the syntax [start : finish : step], where:
  - start is the index where we start the slice
  - finish is the index of **one after** where we end the slice
  - step is how much we count by between each character
- When step is not provided, it defaults to 1

```
>>> x = "I Love Cats"
```

0	1	2	3	4	5	6	7	8	9	10
I		**L**	**o**	**v**	**e**		**C**	**a**	**t**	**s**
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x[0:6:2]
'ILv'
```

```
>>> x[::3]
'Io t'
```

```
>>> x[::-1]
'staC evoL I'
```

```
>>> x[::-2]
'sa vLI'
```

Let's Code!

- Let's take a look at how this works in Python!
  - String indexing
  - String slicing
  - String length
  - String slicing with a 'step'

Open your notebook

Click Link:
2. String Indexing and Slicing

Modifying Strings

- The indexing and slicing operations do not modify the string they act on
  - We cannot change a string!
- Strings are immutable, meaning they **CANNOT** be changed

```
>>> x = "I Love Cats"
```

0	1	2	3	4	5	6	7	8	9	10
I		**L**	**o**	**v**	**e**		**C**	**a**	**t**	**s**
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x[7] = 'B'
TypeError: 'str' object does not support item assignment
```

Modifying Strings

- To "modify" a string, we must create a new one
  - Let's change this to "I Love Dogs"

```
>>> x = "I Love Cats"
```

0	1	2	3	4	5	6	7	8	9	10
I		**L**	**o**	**v**	**e**		**C**	**a**	**t**	**s**
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x_new = x[:7] + 'Dogs'      #x_new points to the new string object
>>> x = x_new                   #x points to the new object
>>> print(x)
'I Love Dogs'
```

Let's Code!

- Let's take a look at how this works in Python!
  - Modifying strings
  - String immutability

Open your notebook

Click Link:
3. Modifying Strings

Mentimeter Checkpoint

- **Join at www.menti.com:**
  - **Code: 95 32 42 1**
- **Link**
  - https://www.menti.com/bl69ar1bwgee

Working with Strings

- The string (str) type was briefly introduced in previous weeks
- Let's take our string knowledge to the next level!
  - escape sequences ✓
  - str operations ✓
  - type conversion ✓
  - str indexing and slicing ✓
  - str methods

String Methods

- Strings are objects and just like other objects, the str type has associated methods that are only valid for strings
- To find out which methods are associated with objects, use the built-in function dir

```
>>> dir(str)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
'__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize',
'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find',
'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal',
'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace',
'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans',
'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit',
'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title',
'translate', 'upper', 'zfill']
```

String Method: upper

- upper is a string method that generates a new string that has all upper case characters

```
>>> white_rabbit = "I'm late! I'm late! For a very important date!"

>>> white_rabbit.upper()
"I'M LATE! I'M LATE! FOR A VERY IMPORTANT DATE!"

>>> white_rabbit
"I'm late! I'm late! For a very important date!"
```

String Method: lower

- lower is a string method that generates a new string that has all lower case characters

```
>>> white_rabbit = "I'm late! I'm late! For a very important date!"

>>> white_rabbit.lower()
"i'm late! i'm late! for a very important date!"

>>> white_rabbit
"I'm late! I'm late! For a very important date!"
```

String Method: find (and rfind)

- The method find returns first index where a substring is found
- The method rfind returns the last index where a substring is found
- Returns -1 if no such substring exists

```
>>> white_rabbit = "I'm late! I'm late! For a very important date!"

>>> white_rabbit.find('late')
4

>>> white_rabbit.rfind('late')
14
```

String Method: replace

- The method replace(old, new) returns a copy of the string in which the occurrences of old have been replaced with new.

```
>>> white_rabbit = "I'm late! I'm late! For a very important date!"


>>> white_rabbit.replace('late','early')
"I'm early! I'm early! For a very important date!"
```

Chaining Methods

- How would you replace the word “forward” with “backward”?

```
>>> s = 'Forward, forward we must go, for there is no other way to go!'
```

- Methods can be chained together
  - Perform first operation, which returns an object
  - Use the returned object for the next method

```
>>> s.lower().replace('forward','backward')
'backward, backward we must go, for there is no other way to go!'
>>> s.lower().replace('forward','backward').capitalize()
'Backward, backward we must go, for there is no other way to go!'
```

More String Methods

- There are many more string methods available which you will explore in lab assignments and tutorials
  - `str.islower()`
  - `str.count(sub)`
  - `str.ljust(width)`
  - `str.lstrip()`
  - `str.split()`
  - `str.strip()`

Do not need to memorize all of the string methods!

Should know:

`str.lower`
`str.upper`
`str.find`
`str.rfind`
`str.replace`
`str.capitalize`

(any others will be indicated in the review)

Let's Code!

- Let's take a look at how this works in Python!
  - String methods
    - capitalize
    - upper
    - lower
    - find
    - rfind
    - And more…

Open your notebook

Click Link:
4. String Methods

Mentimeter Checkpoint

- **Join at www.menti.com:**
  - **Code: 9493 9270**
- **Link**
  - https://www.menti.com/bldrkshx7mcz

APS106

UNIVERSITY OF TORONTO

Strings: Conversions, Indexing, Slicing, Immutability.

Week 5 | Lecture 2 (5.2)

if nothing else, write #cleancode