String Processing

Outline

Strings are Sequences

Strings are Objects

Converting Between Types

Quiz

Assignment 4: Text Processing

Strings are Sequences

- We've seen a bunch of strings already:
  - `"Literal strings"`
  - `f"Format {p} strings"`
  - `"Strings" + made + " from " + str(various_things)`
- Strings, like tuples, are sequences of values
- You may also see the term *iterable*

len

There are lots of things we can do with any sequence.

We can ask how long a sequence is with the `len` function.

What will this evaluate to?

```
len("hello") < len((1, 2, 3))
```

len

len produces an int so you can use it anywhere an integer is appropriate:

```
x = 0
while x < len("hello"):
    x = x + 1
print(f"The string is {x} characters long")
```

for. . . in. . .

We saw another type of sequence last week: `range`.

```
for x in range(0, 5):
    print(x)
```

for… in…

We can use `for var in seq` for any sequence!

```
for c in "hello":
    print(c)
```

What do you think this will do?

Finding a Character

With this technique we can build up some interesting ideas.

```
for c in "hello":
    if c == "o":
        print("Found an o!")
```

Exercise: Write a function `find_position(s, c)` that finds the position of the first occurrence of character `c` in string `s`. E.g., `find_position("hello", "e")` should be equal to 1 (remember, we count from 0).

Exercise: Finding the Last Character

Write a function `last_char(s)` that returns the final character of a string `s`. There are a few ways to do this!

Sequence Operations: +

- You've already seen +
  - It works on any kind of sequence!
- `"hi " + "there"`
- `(1, 2) + (3, 4)`
- and so on!

Sequence Operations: *

- * is kind of funny
  - The right hand side has to be a number
- `string * number`
- `tuple * number`

Sequence Operations: `in`

- `in` appears in another way too!
  - `"e" in "hello"`
  - `2 in (4,3,2,1)`
- We can check if something is inside of a seq
  - But it means sort of different things for different seqs
  - `item in tuple`
  - `str1 in str2`

Indexing: []

- Another thing we can do with sequences: Index them!
  - We saw this before:

    ```
    tup = (1, 2, 3)
    second_elt = tup[1]
    third_elt = tup[2]
    ```
- We can do it with strings too!

Write a function `nth_chars(str1, str2, n)` that returns the `n~th character of each of ~str1` and `str2`. Don't use a loop, and do return a tuple.

Slicing: Also []

We can use `[start:end]` to take a part of a string as well:

```
x = "some cool string"
y = x[5:9]
z = x[9:]
print(y)
print(z)
print(x[:4])
```

Strings are Objects

- So far we've seen a bunch of Python value types:
  - `int`, `float`, `str`, tuple, etc
- All values in Python are *objects*
  - There are different `types` of objects
  - E.g. `int`, `str`, `tuple`, whatever
- An object has *state* and *methods* to operate on that state

String Methods

What do you think these will do?

- `"hello".startswith("he")`
- `"hello".endswith("lo")`

-
  ```
  x = "hello"
  print(x.find("l"))
  ```

The Return of Dot Syntax

This is that dot again!

We saw it before in this situation:

```
import math
print(math.sqrt(16))
```

`math` here isn't an object, but we use the same syntax with objects: dot gives us access to the fields of an object.

More String Methods

- `lower`, `upper`, `islower`, `isupper`
- `replace`
- `count`
- `lstrip`, `rstrip`

Python String methods

`int()`, `str()`, etc

- We've seen a few functions to convert between types
  - `int(x)`, `float(x)`, `str(x)`

Find and slice

We can combine `find` and `slice` to do some cool tricks:

```
x = "My number is: 47"
pos = x.find(":") + 2
new_number = int(x[pos:])*2
print(f"My new number is {new_number}")
```

Exercise: Before and After

Write a function that splits a string into two parts: The part up to a vertical bar character `|`, and the string following the vertical bar.

Quiz

Assignment 4

Forbidden Functions

We saw a lot of cool string functions today!

You are allowed to use almost none of them. See the assignment for details!