Lecture 7: Sequences

Eleanor Birrell February 10, 2026

Review: Programming in Python

- Values
  - `47`
  - `"hello, world!\n"`
- Types
  - `str`
  - `bool`
- Variables
  - `dist_in_miles = 3.1`
  - `a_string = "hello"`
- Operations
  - `1 * 2 * 3`
  - `a_string + " world"`
- Functions

  -
    ```
    def example(x):
        y = 2*x
        return y
    ```
  - `z = example(25)`
  - `print("hello, world!")`
- Control Flow
  - `if`
  - `if-else`
  - `for x in range(10)`
  - `while (x < 5)`

Strings are Sequences

```
string = "Sam I am"
```

('S', 'a', 'm', ' ', 'I', ' ', 'a', 'm')
  0 1 2 3 4 5 6 7

- length function `len`
  - `x = len(string)`
- indexing
  - `char = string[2]`
  - `char2 = string[-2]`

Example

- Define a function str_even that takes one parameter s (a string) and returns a string comprised of only the even characters of s

Exercise 1

- Define a function `findchar` that takes two parameters, a string `s` and a character `c` and returns the index of the first instance of that character. If that character does not appear in the string, it returns -1

- findchar("hello", "h") == 0
- findchar("hello", "l") == 2
- findchar("hello", "a") == -1

Two ways to process each char in a string

- 1. iterate based on index

```
for i in range(len(string)):
      print(string[i])
```

- 2. iterate over items

```
for char in string:
      print(char)
```

Example

- Define a function str_even that takes one parameter s (a string) and returns a string comprised of only the even characters of s

Exercise 2

- Without using string indexing, define a function `countchar` that takes two parameters, a string `s` and an char `c` and returns the number of times that character appears in the string.

- `countchar("hello", "h") == 1`
- `countchar("hello", "l") == 2`
- `countchar("hello", "a") == 0`

slicing (1)

- For extracting part of a sequence

```
s[:]
s[start:]
s[:end]
s[start:end]
```

```
>>> s = "Hello world!\n\n"
>>> s[6]
     'w'
>>> s[2:7]
     'llo w'
>>> s[5:]
     ' world!\n\n'
>>> s[:5]
     'Hello'
```

slicing (2)

- For extracting part of a sequence

```
s[:]
s[start:]
s[:end]
s[start:end]

s[start::step]
s[:end:step]
s[start:end:step]
```

```
>>> s = "Hello world!\n\n"
>>> s[2::2]
     'lowrd\n'
>>> s[1:10:3]
     'eoo'
>>> s[:5:2]
     'Hlo'
>>> s[-3:-10:-1]
     '!dlrow '
```

Exercise 3

- Evaluate the following expressions.

```
test = "This is a string"
```

- `test[10]`
- `test[-1]`
- `test[0:2]`
- `test[2:6]`
- `test[:5]`
- `test[::2]`

Example

- Define a function str_even that takes one parameter s (a string) and returns a string comprised of only the even characters of s

Tuples

- a tuple is an ordered set of elements:

```
(3, 6, 2, 1)
```

- examples to create a tuple:

```
tup = (3, 6, 2, 1)
tup1 = ("a", "b", "c")
tup2 = tuple("abc") #cast from str
```

- a tuple is a sequence, so can index into, loop over, check for membership, slice, etc

```
>>> tup[1]
>>> 6
```

- operators: + and *

Exercise 4

- Define a function `average` that takes one parameter (a tuple `vals` containing a sequence of numbers) and returns the average of the numbers in the tuple.

Strings and Tuples are immutable

- strings and tuples are immutable (you can make new sequences, but you can’t change an existing one in place)

```
tup = (3, 6, 2, 1)
tup[0] = 4
```

- TypeError: ’tuple’ object does not support item assignment