

Objectives

- Wrap up string formatting
- A new data type: Lists

March 4, 2019

Sprenkle - CSCI111

1

Review

- What is the special name for sequences, like newlines, tabs, ...?
 - How do we represent them in strings?
- How can we get fine-grained control over how our data is displayed?
 - What is that called?
 - How do we write them?
 - How would you display a float to 3 decimal places?

March 4, 2019

Sprenkle - CSCI111

2

Example: Printing Out Tables

- A table of temperature conversions

Temp F	Temp C	Temp K
-459.7	-273.1	0.0
0.0	-17.8	255.2
32.0	0.0	273.1

- If we want to print data in rows, what is the template for what a row looks like?
 - How do we make the column labels line up?

March 4, 2019

Sprenkle - CSCI111

temp_table.py

3

Sequences of Data


- Sequences so far ...
 - `str`: sequence of characters
 - `range`: generator (sequence of numbers)
- We commonly group a sequence of data together and refer to them by one name
 - Days of the week: Sunday, Monday, Tuesday, ...
 - Months of the year: Jan, Feb, Mar, ...
 - Shopping list
- Can represent this data as a **list** in Python
 - Similar to **arrays** in other languages

March 4, 2019


Sprenkle - CSCI111

4

Lists: A Sequence of Data Elements

element  daysInWeek

"Sun"	"Mon"	"Tue"	"Wed"	"Thu"	"Fri"	"Sat"
0	1	2	3	4	5	6

Position/index in the list  len(daysInWeek) is 7

- Elements in lists can be *any* data type

What does this look similar to, in structure?

March 4, 2019

Sprenkle - CSCI111

5

Example Lists in Python

- Empty List: `[]`
- List of `str`s:
 - `daysInWeek=["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]`
- List of `float`s
 - `highTemps=[60.4, 70.2, 63.8, 55.7, 54.2]`
- Lists can contain >1 type
 - `wheelOfFortune=[250, 1000, "Bankrupt", "Free Play"]`

Syntax for list: `[]`
How different from accessing a character in a string?

March 4, 2019

Sprenkle - CSCI111

6

Benefits of Lists

- Group related items together
 - Instead of creating separate variables
 - `sunday = "Sun"`
 - `monday = "Mon"`
- Convenient for dealing with large amounts of data
 - Example: could keep all the temperature data in a list if needed to reuse later
- Functions and methods for handling, manipulating lists

March 4, 2019

Sprenkle - CSCI111

7

List Operations

Similar to operations for strings

Concatenation	<code><seq> + <seq></code>
Repetition	<code><seq> * <int-expr></code>
Indexing	<code><seq>[<int-expr>]</code>
Length	<code>len(<seq>)</code>
Slicing	<code><seq>[:]</code>
Iteration	<code>for <var> in <seq>:</code>
Membership	<code><expr> in <seq></code>

March 4, 2019

Sprenkle - CSCI111

8

Lists: A Sequence of Data Elements

"Sun"	"Mon"	"Tue"	"Wed"	"Thu"	"Fri"	"Sat"
0	1	2	3	4	5	6

len(daysInWeek) is 7

- `<listname>[<int_expr>]`

- Similar to accessing characters in a string
- `daysInWeek[-1]` is "Sat"
- `daysInWeek[0]` is "Sun"

March 4, 2019

Sprenkle - CSCI111

9

Iterating through a List

- Read as

- For every element in the list ...

An item in the list

list object

```
for item in list:  
    print(item)
```

Iterates through
items in list

- Output equivalent to

```
for x in range(len(list)):  
    print(list[x])
```

Iterates through
positions in list

March 4, 2019

Sprenkle - CSCI111

daysOfWeek.py

10

Example Code

```
friends = ["Alice", "Bjorn", "Casey", "Duane", \
           "Elsa", "Farrah"]

for name in friends:
    print("I know " + name + ".")
    print(name, "is a friend of mine.")

print("Those are the people I know.")
```

friends.py

March 4, 2019

Sprenkle - CSCI111

11

Practice

- Get the *list* of weekend days from the days of the week list
 - daysInWeek=["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]

March 4, 2019

Sprenkle - CSCI111

12

Practice

- Get the *list* of weekend days from the days of the week list

➤ `daysInWeek=["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]`

➤ `weekend = daysInWeek[:1] + daysInWeek[-1:]` ← Gives back a *list*

or

➤ `weekend = [daysInWeek[0]] + [daysInWeek[-1]]` ← Gives back an element of list, which is a *str* ¹³

March 4, 2019

Sprenkle - CSCI111

Membership

- **Check if a list contains an element**

- Example usage

➤ `enrolledstudents` is a list of students who are enrolled in the class

➤ Want to check if a student who attends the class is enrolled in the class

```
if student not in enrolledstudents:  
    print(student, "is not enrolled")
```

March 4, 2019

Sprenkle - CSCI111

14

Making Lists of Integers Quickly

- If you want to make a list of integers that are evenly spaced, you can use the **range** generator
- Example: to make a list of the even numbers from 0 to 99:

➤ `evenNumList = list(range(0, 99, 2))`

Converts the generated numbers into a list

March 4, 2019

Sprenkle - CSCI111

15

str Method Flashback

- `string.split([sep])`
 - Returns a **list** of the words in the string `string`, using `sep` as the delimiter string
 - If `sep` is not specified or is `None`, any *whitespace* (space, new line, tab, etc.) is a separator
 - Example:

```
phrase = "Hello, Computational Thinkers!"  
x = phrase.split()
```

What is x? Its data type? What does x contain?

March 4, 2019

Sprenkle - CSCI111

16

str Method Flashback

- `string.join(iterable)`

- Return a string which is the concatenation of the *strings* in the **iterable**/sequence. The separator between elements is **string**.

- Example:

```
x = ["1", "2", "3"]  
phrase = " ".join(x)
```

What is `x`'s data type?
What is `phrase`'s data type?
What does `phrase` contain?

March 4, 2019

Sprenkle - CSCI111

17

Looking Ahead

- Pre lab for Lab 7 due tomorrow before lab
 - Think about the Caesar Cipher implementation
- Lab 7 pairs
- Broader Issue: Cryptography

March 4, 2019

Sprenkle - CSCI111

18