Lecture 13: Lists

CS 51P

October 23, 2019

Programs operate on values

- compute new values using expressions
- store values in variables
- pass values to functions (as arguments)
- pass values to caller (as return value)

Can we operate on multiple values at the same time?

- Can we define a variable that stores the colors of the rainbow?
- Can we define a function that returns the squares of all the numbers in a specified range?
- Can we define a function that returns all the words in a string that begin with uppercase letters?

Data Structures

- a **data structure** is a type that stores a collection of values
- Python provides some built-in data structure types

Sequences

- sequences are ordered sets of values
  - ranges are sequences of integers
  - strings are sequences of characters
  - files are sequences of strings

- we can perform operations on sequences
  - indexing (e.g., "hello"[0])
  - slicing (e.g., "hello"[1:5])
  - looping (with for loop) (e.g., for i in range(1,10):)
  - check membership (e.g., char in "abcd")

Can we have a sequence of arbitrary values?

Lists

- a list is an ordered set of elements:

```
a_list = [3, 6, 2, 1]
```

- many ways to create a list including:

```
a_list = [3, 6, 2, 1]
b_list = []
c_list = "a b c d".split()
d_list = open("temp.txt","r").readlines()
```

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

- operators: + and *

Lists as sequences

```
string = "Hello world !! "
print(string[1:3])
print(string[-1])
print(string[:2])

str_list = string.split()
print(str_list)
print(str_list[1:3])
print(str_list[-1])
print(str_list[:2])
```

Example: Lists as sequences

- Can we define a variable that stores the colors of the rainbow?
  - colors = ["red", "orange", "yellow", "green", "blue", "purple"]

Differences about Lists

- the elements of a list can have any value and any type

```
a_list = [3.5, 6, [1, 2], "abc"]
```

- lists are mutable
  - add elements

```
a_list.append("c")
a_list.extend(["c","b"])
```

  - modify elements

```
a_list[3] = 3.33333
a_list[:2] = ["a", "b"]
```

  - remove elements

```
a_list.pop()
del(a_list[0:1])
```

List Operations

adding to a list

- a_list.extend(*list*)
- a_list.append(*elem*)
- a_list.insert(*index*, *elem*)

other

- min(a_list), max(a_list), len(a_list)
- *elem* in a_list
  - returns bool
- a_list.index(*elem*)
  - returns int or error

removing from a list

- del(a_list[*slice*])
- a_list.remove(*elem*)
  - error if *elem* not in a_list
- a_list.pop()
  - returns (and removes) a_list[-1]
- a_list.pop(*index*)
  - returns (and removes) a_list[index]

modifying a list

- direct assignment
  - a_list[0] = 2

Exercise

```
a_list = [3.5, 6, [1, 2], "abc"]
a_list[3] = list(range(0,5,2))
a_list[:2] = ["a", "b"]
a_list.extend([5,3,1])

print(len(a_list))
for elem in a_list:
     print(str(elem) + ":" + str(type(elem)))

del(a_list[3:5])
a_list.remove("a")
print(a_list)
```

Example

- Can we define a function that returns the squares of all the numbers in a specified range?

Exercise

- Define a function digits that takes one parameter num (an positive int) and returns a list of the digits of num

Example

- Define a function word_list that takes a filename as an argument and returns a list of all the words in that file.

Exercise

- Define a function count_words that takes a filename as input and returns the total number of unique words in that file