UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science

Lecture 14:

Mutability

1

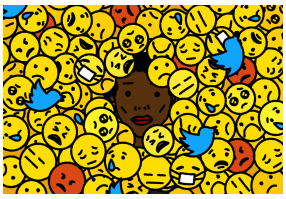
Announcements

- Maps checkpoint due Weds 10/21, 11:59pm
  - I was wrong last week: Maps *is* a partner project. You may work with 1 other partner.
  - You may not share code with anyone in CS88, or previously in CS88 or 61A.
- Reminder: You have 8 total slip days to use on all projects, labs and homework
  - You may still use no more than 3 per single deadline.
- Midterm Grading: Please review your exam.
  - We should have compiled all the missing answers, but if they are still missing SUBMIT A REGRADE REQUEST. We have all the logs, but a few responses were not transferred correctly.

2

Computing In the News

- Is Everybody Doing ... OK? Let's Ask Social Media (NYT)
- Researchers are looking at online behavior to gauge public mental health. The results aren't pretty.
- The Computational Story Lab is part of a small but growing field of researchers who try to parse our national mental health through the prism of our online life
- The researchers call it the Hedonometer. It is the invention of Chris Danforth and his partner Peter Dodds, both trained mathematicians and computer scientists and the co-directors of the lab. The Hedonometer has been up and running for more than a decade now, measuring word choices across millions of tweets, every day, the world over, to come up with a moving measure of well-being.



3

UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science

Mutability: Lists

4

Learning Objectives

- Distinguish between when a function changes some data, vs returns a new object
- Understand modifying objects in place
- Python provides "is" and "==" for checking if items are the same, in different ways

5

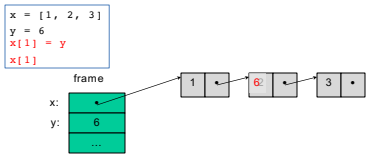
Objects

- An Abstract Data Type consist of data and behavior bundled together to abstract a view on the data
- An object is a concrete instance of an abstract data type.
- **Objects can have state**
  - mutable vs immutable
  - Mutable: We can change the object after it has been created
  - Immutabe: We cannot change the object.
- **Next topic: Object-oriented programming**
- In Python, every value is an object
  - All objects have attributes
  - Manipulation happens through method calls

6

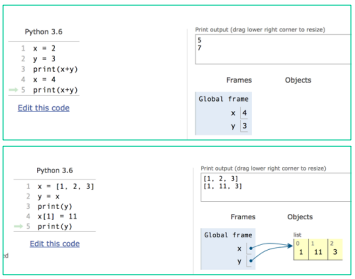
From value to storage …

- A variable assigned a compound value (object) is a reference to that object.
- Mutable object can be changed but the variable(s) still refer to it



7

Mutation makes sharing visible



8

Mutating Input Data

- Functions can mutate objects passed in as an argument
- Declaring a new variable with the same name as an argument only exists within the scope of our function
- BUT, we can still modify the object passed in, even though it was created in some other frame or environment.
- Python Tutor

9

Copies, 'is' and '=='

```
>>> alist = [1, 2, 3, 4]
>>> alist == [1, 2, 3, 4]  # Equal values?
True
>>> alist is [1, 2, 3, 4]  # same object?
False
>>> blist = alist          # assignment refers
>>> alist is blist         # to same object
True
>>> blist = list(alist)    # type constructors copy
>>> blist is alist
False
>>> blist = alist[ : ]     # so does slicing
>>> blist is alist
False
>>> blist
[1, 2, 3, 4]
>>>
```


10

UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science

Dictionaries

11

Learning Objectives

- Dictionaries store key-value pairs
  - "a" maps to some value "b"
- You can change the values of the dictionary
- You can change the keys, including adding and removing
- Each key in each dictionary maps to one value.

12

Mutability

- Immutable – the value of the object cannot be changed
  - integers, floats, booleans
  - strings, tuples
- Mutable – the value of the object can change
  - Lists
  - Dictionaries

```
>>> alist = [1,2,3,4]
>>> alist
[1, 2, 3, 4]
>>> alist[2]
3
>>> alist[2] = 'elephant'
>>> alist
[1, 2, 'elephant', 4]
```

```
>>> adict = {'a':1, 'b':2}
>>> adict
{'b': 2, 'a': 1}
>>> adict['b']
2
>>> adict['b'] = 42
>>> adict['c'] = 'elephant'
>>> adict
{'b': 42, 'c': 'elephant', 'a': 1}
```


13

Dictionaries – by example

- **Constructors:**
  - `dict( hi=32, lo=17)`
  - `dict([('hi',212),('lo',32),(17,3)])`
  - `{'x':1, 'y':2, 3:4}`
  - `{wd:len(wd) for wd in "The quick brown fox".split()}`
- **Selectors:**
  - **`water['lo']`**
  - **`<dict>.keys(), .items(), .values()`**
  - **`<dict>.get(key [, default] )`**
- **Operations:**
  - **`in, not in, len, min, max`**
  - **`'lo' in water`**
- **Mutators**
  - `water[ 'lo' ] = 33`

14

UC Berkeley EECS
Lecturer
Michael Ball

Computational Structures in Data Science

Mutable Functions

15

Learning Objectives

- Distinguish between when a function changes some data, vs returns a new object
- Understand modifying objects in place
- global allows a function to modify a global variable.
- nonlocal allows a function to modify a variable in an outer scope, such as a parent frame
  - But does not look in the global frame even if it is the parent

16

Creating mutating 'functions'

- Pure functions have *referential transparency*
  - `c = greet() + name()` is "referentially transparent" if we can replace that expression with the value, maybe that's `c = "Hello, CS 88"`
  - Result value depends only on the inputs
    - Same inputs, same result value
- **Functions that use global variables are not pure**
- They can be "mutating", and rely on some state

```
>>> counter = 0
>>> def count_fun():
...     global counter
...     counter += 1
...     return counter
...
>>> count_fun()
1
>>> count_fun()
2
```


17

Creating mutating 'functions'

```
>>> counter = 0
>>> def count_fun():
...     global counter
...     counter += 1
...     return counter
...
>>> count_fun()
1
>>> count_fun()
2
```

How do I make a second counter?

```
>>> def make_counter():
...     counter = 0
...     def counts():
...         nonlocal counter
...         counter +=1
...         return counter
...     return counts
...
>>> count_fun = make_counter()
>>> count_fun()
1
>>> count_fun()
2
>>> another_one = make_counter()
>>> another_one()
1
>>> count_fun()
3
```


18