SDS Base Camp

Names and Containers

Samantha Breslin
September 15, 2021

KØBENHAVNS UNIVERSITET

Schedule

- Naming & Commenting
- Containers
  - Lists (recap)
    - List-like things with strings
  - Dictionaries
  - Tuples
  - Sets

Naming and Commenting

Naming

- What are examples from online or your own experience of variables/categories where naming mattered? (e.g. non-descriptive, problematic, really well done)

Naming II

- When programming & doing data analysis you have to name many things
  - Many many variables
  - Data categories
  - Functions (when we get there)
  - Files
- Naming matters!

Naming III

- Can show assumptions (who/what gets counted, or not)
- Example: Binary gender categories in online forms, data categories, variables, etc.
  - Assumes that gender operates as a binary
  - Excludes other categories of gender
  - Homogenizes within each category
- More in other courses about what data tells us (or not)
  - E.g. Facebook gender options for users vs. advertisers
- These dilemmas extend to minute practice of naming in code

Naming IV

- Programming languages also have names & terminology
- Some terminology in how you refer to certain data structures, etc.
  - May vary by discipline, but also holds assumptions, values, and norms
- Some built in by the people who designed them
  - Keywords that you can’t reuse
  - Also holds assumptions, values, and norms

Power of Names & Words

- Names & words have power
  - They have an effect in the world and on people
  - They shape the way we conceptualize our programs & our data

MOTHERBOARD
TECH BY VICE

‘Master/Slave’ Terminology Was Removed from Python Programming Language

The terminology has been a point of contention in the tech community for nearly two decades and now it was just removed from one of the most popular programming languages in the world.

Naming practices

- Clear & descriptive
- Attentive to effects of name choices
- Explicit and reflexive about assumptions (also part of commenting)
- camelCase or under_score naming – be consistent
- Capitalization matters

Commenting Practices

- Usually comment above the code being explained
  - Can sometimes use inline comments
- Make comments readable (e.g. space after #, align inline comments, line spacing)
- Good naming practices can reduce the need for comments
- Be concise, precise, and clear to explain what code *does*
- Can also use comments to document & make explicit assumptions and decisions
  - E.g. Cases your code does not deal with, choice of categories

See https://towardsdatascience.com/the-art-of-writing-efficient-code-comments-692213ed71b1

Lists & Strings (recap +)

Lists (recap)

- A `list` is another (more complex) data type
- A `list` is a container of sequential elements that can be accessed by their index
- List methods:
  - append(x): add an element to the end of a list
  - remove(x): remove an element from the list
  - Change elements at specific indices
- Lots more!
  - sort, min, max, pop, etc.
  - https://docs.python.org/3/tutorial/datastructures.html

More Strings I

Recall:

- Strings are sequences of characters
- Strings can be added and modified using string methods
- Strings are like lists in many ways

Indices with Strings

- Use character indices like a list
- You can use a range of indices
  - `a_string[3:5]` would retrieve the 4th and 5th element
  - You can do this with lists too!

```
my_string = "I'm a string!"
```

What's the result of `my_string[3]` and `my_string[1:3]`?

String methods for “splitting”

Documentation break – look up these methods, what do they do?

- str.rpartition(sep)
- str.rsplit(sep, maxsplit)
- str.lsplit([chars])
- str.split(sep, maxsplit)

https://docs.python.org/3/library/stdtypes.html#string-methods

Example

Example:

`my_url = “https://realpython.com/python-lists-tuples/”`

`my_url.split(“/”)` to split with the “/” character

Output: ['https:', '', 'realpython.com', 'python-lists-tuples', '']

Dictionaries

Dictionaries I

- A type of container accessed by keys rather than an index
  - Uses {key: value} pairs
  - Values can be many things
  - Keys must be unique
  - Dictionaries are changeable, duplicate keys NOT allowed

Creating Dictionaries

- Use Curly Brackets, colon :, and comma ,

```
empty_dictionary = {}

my_dictionary = {'Samantha': 'Anthropology',
                 'Friedo': 'Sociology',
                 'Greg': 'Political Science'}
```

Accessing Dictionaries

- Access dictionary values by the keys within square [] brackets
- Use dictionary methods to access keys and values

```
my_dictionary = {'Samantha': 'Anthropology',
                 'Friedo': 'Sociology',
                 'Greg': 'Political Science'}
```

What is the result of print(`my_dictionary['Greg']`)?

If you say `my_dictionary['Greg'] = 'SDS'` what is the result now?
How can you check?

Adding to Dictionaries

- Dictionaries can be updated
- Add elements by assigning key, value pairs OR update() method

```
my_dictionary = {'Samantha': 'Anthropology',
                 'Friedo': 'Sociology',
                 'Greg': 'Political Science'}

my_dictionary.update({'Asger':'SDS'})
my_dictionary['Hjalmar'] = 'Sociology'
```

What should you think about in terms of readability, etc.?

Removing from Dictionaries

- You can remove everything with clear()
- Use pop() to remove specific key-value

```
my_dictionary = {'Samantha': 'Anthropology',
                 'Friedo': 'Sociology',
                 'Greg': 'Political Science'}

my_dictionary.pop('Greg')
```

How many key-value pairs are in your dictionary now? How would you find out? What if you had already run the code on the previous slide?

Making Dictionaries from Lists

- You can make dictionaries out of lists
  - zip(list) to zip two lists together (like a zipper)
  - NOTE: Order matters!!

Example:

```
keys = ['Samantha', 'Friedo', 'Greg']
ages = [103, 78, 85]
key_values = list(zip(keys, ages))

my_dictionary2 = dict(key_values)
```

What's the output of: `my_dictionary2['Ulf']`

Dictionaries

- Can get complicated!
- Often used for storing and structuring data

Tuples

Tuples

- A tuple is a list that can't be changed
  - my_tuple = ('Samantha', 'Anthropology', '16.0.05')
- Key features: ordered, unchangeable, allows duplicates
- You can have tuples within lists, lists within tuples, tuples within tuples, etc.
- Accessing tuple elements is like accessing lists
- Access nested lists and tuples with double indices.
  - E.g. nested_list[0][0] would retrieve the first element of the first list/tuple

Example

```
instructors = [('Samantha', 'Anthropology', '16.0.05'),
               ('Friedo', 'Sociology', ''),
               ('Hjalmar', 'Sociology', '')]
```

How would you access the second value of the second tuple (i.e. Friedo's discipline?)

Could you update the second tuple with Friedo's office number?

Value of Tuples

- Why use a tuple?
  - To preserve relations between data
  - To preserve a dataset

Single Element Tuple Example

```
firsttuple = (“carrot",)
type(firsttuple)


secondtuple = (“potato")
type(secondtuple)
```

What is the type of firsttuple and secondtuple?

Sets

Sets

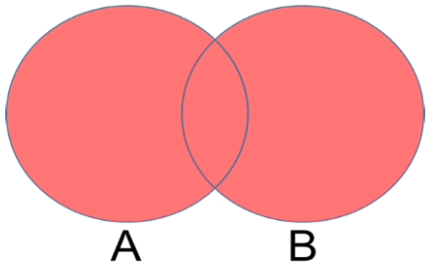
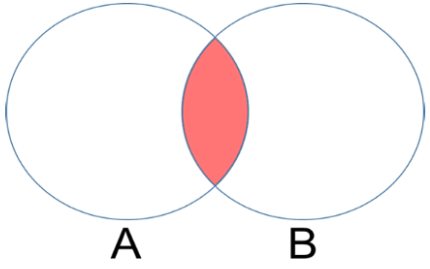
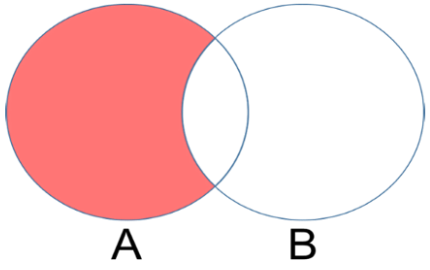
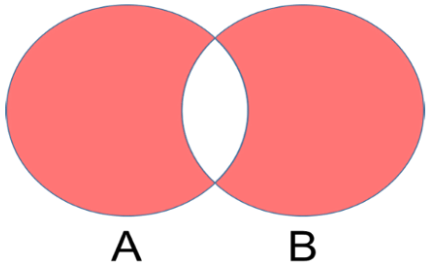
- A set is used to store multiple items
  - basecamp_teachers = {“Samantha”, “Friedo”, “Greg”, “Asger”}
- Key features: A set is unordered and unchangeable and does not allow duplicates

Short coding break

- Try making a set and printing it out – what happens?
- Try making a set with duplicates and then printing – what happens?
- Try making a set that contains a list – what happens?
- Try making a set that contains multiple basic types (string, float, int, Boolean)

Recall sets from math class

- Set methods allow for
  - Creating a new set that is a union of sets
  - Creating a new set that is the intersection of sets
  - Creating a new set that is the difference of sets, or removing the items that are the same as another set
  - Symmetric difference
  - And more
- Remember these terms for merging data frames!

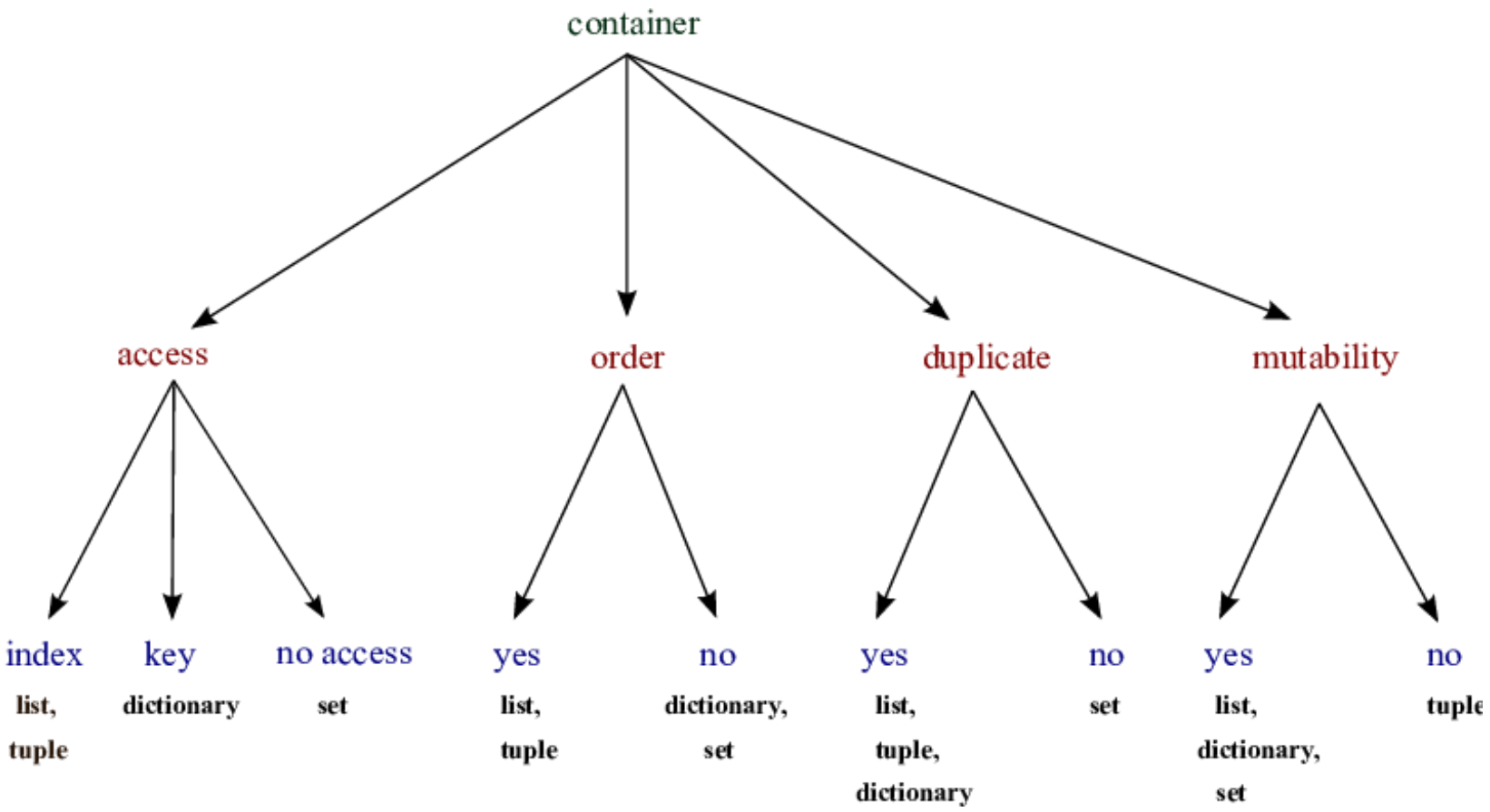
Set Operation	Venn Diagram	Interpretation
Union		$A \cup B$, is the set of all values that are a member of A, or B, or both.
Intersection		$A \cap B$, is the set of all values that are members of both A and B.
Difference		$A \setminus B$, is the set of all values of A that are not members of B
Symmetric Difference		$A \triangle B$, is the set of all values which are in one of the sets, but not both.

Summary

Summary

- Lists are defined with square brackets [ ] and commas ,
  - my_list = [“element 1”, 34, 1.6]
  - Lists can also contain multiple types
  - Lists are mutable (can be modified)
- Strings are defined with quotes (‘, “, or “”“)
  - They all work, but be consistent in which you use
  - Strings are immutable (can’t be modified)
- Dictionaries are defined with curly brackets { }, colons :, and commas
  - my_dictionary = {‘key1’: ‘value1’, ‘key2’: ‘value2’, ‘key3’:’value3’}
  - Dictionaries are mutable (can be modified)
- Tuples are defined with parentheses ( ) and commas ,
  - my_tuple = (11, ‘aa’, 1.5, ‘another string’)
  - Note that tuples can contain multiple types
  - Tuples are immutable (can’t be modified)
- Sets are defined with curly brackets { } and commas ,
  - my_set = {11, ‘sdf’, 3.1, False}
  - Sets are immutable, can contain multiple types (but not lists etc.), and cannot contain duplicates

Summary cont'd



(Maghdid 2016)