ISLab Python Course

Session 2: Data Types in Python

Presenters:

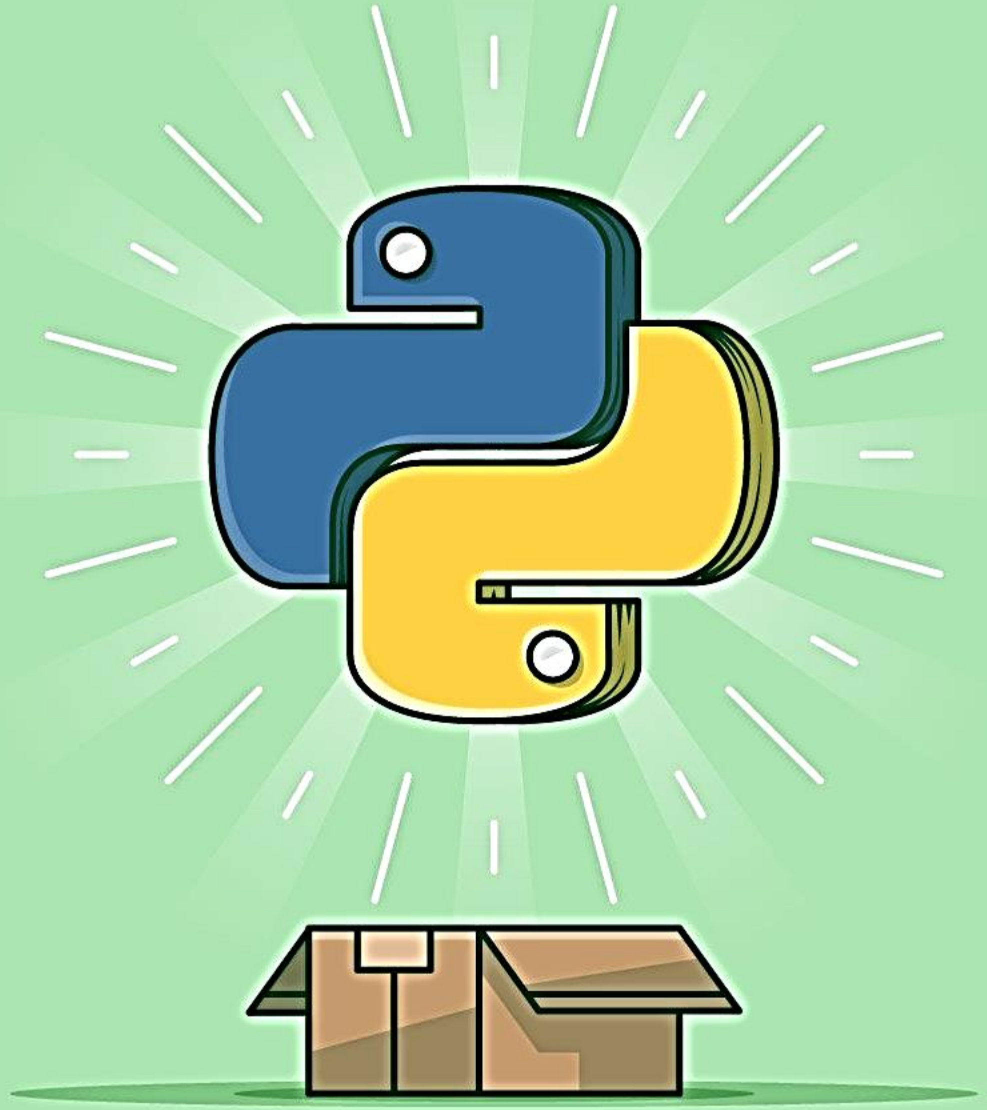
Shahrzad Shashaani

Hamed Homaei Rad

Saeed Samimi

Summer 2023

K.N.Toosi University of Technology



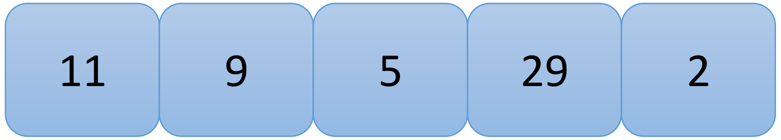
Composite Data Types

✗

```
var_1 = 11
var_2 = 9
var_3 = 5
var_4 = 29
...
var_20 = 2
```

✓

```
variables = [11,9,5,29,2]
```



11	9	5	29	2

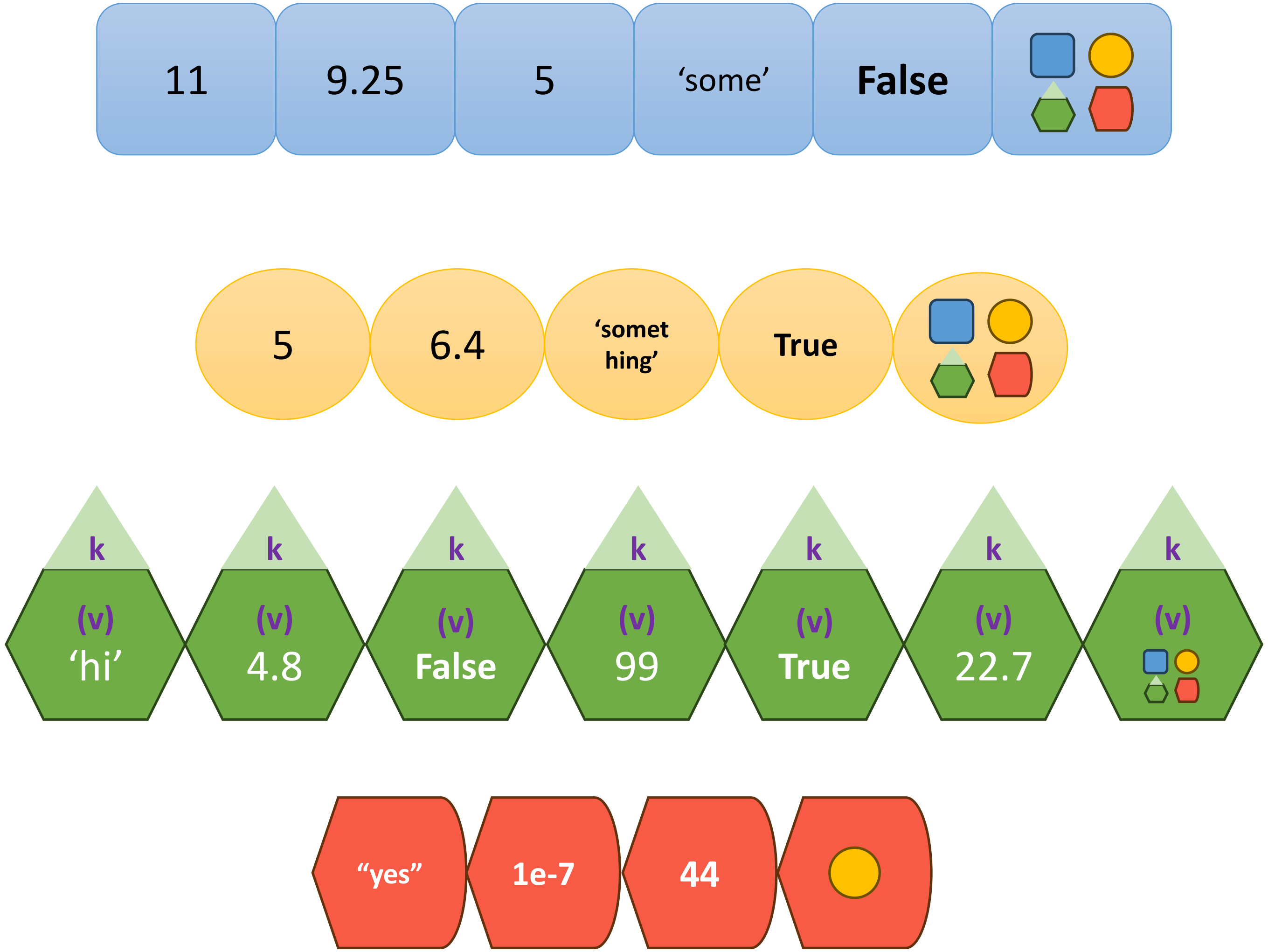
Composite Data Types

- Python has a variety of data storage notions
- Each data structure has its unique properties

- The combination of properties that make these data types unique:
  - **Ordered / Unordered**
  - **Mutable / Immutable**

- What are these Array-like structures?
  - Lists ☆ ☆ ☆ ☆ ☆
  - Tuples ☆ ☆ ☆
  - Dictionaries ☆ ☆ ☆ ☆ ☆
  - Sets ☆

Composite Data Types



List

11 9.25 5 ‘some’ False

Tuple

5 6.4 ‘somet hing’ True

Dictionary

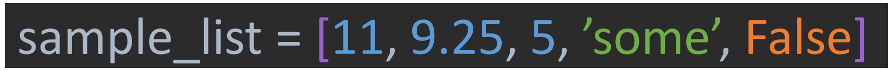
k k k k k k k

(v) ‘hi’ (v) 4.8 (v) False (v) 99 (v) True (v) 22.7 (v)

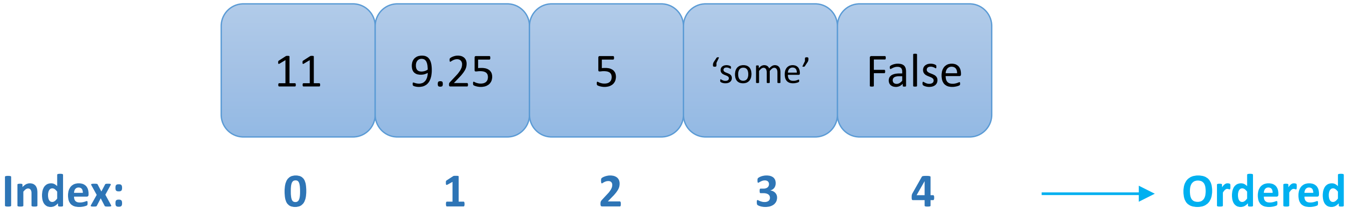
Set

“yes” 1e-7 44

Lists



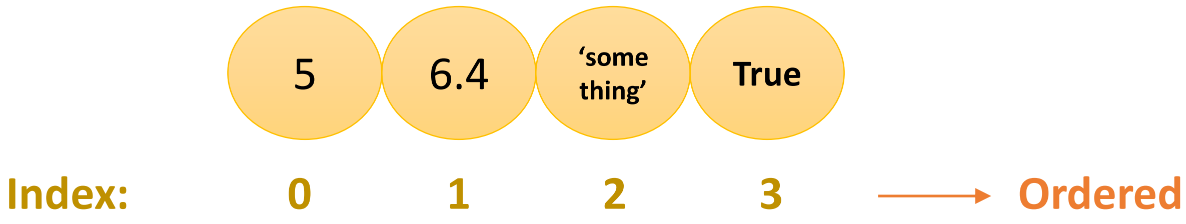
```
sample_list = [11, 9.25, 5, ’some’, False]
```



- Lists are usually known as Arrays in other programming languages
- Lists are declared by **brackets []**
- Lists are **ordered** and **mutable**
- Lists are indexed with integer numbers
- List members (elements) could be of any form (basic data types, composite data types, data structure & objects)

Tuples

```
sample_tuple = (5, 6.4, ’something’, True)
```



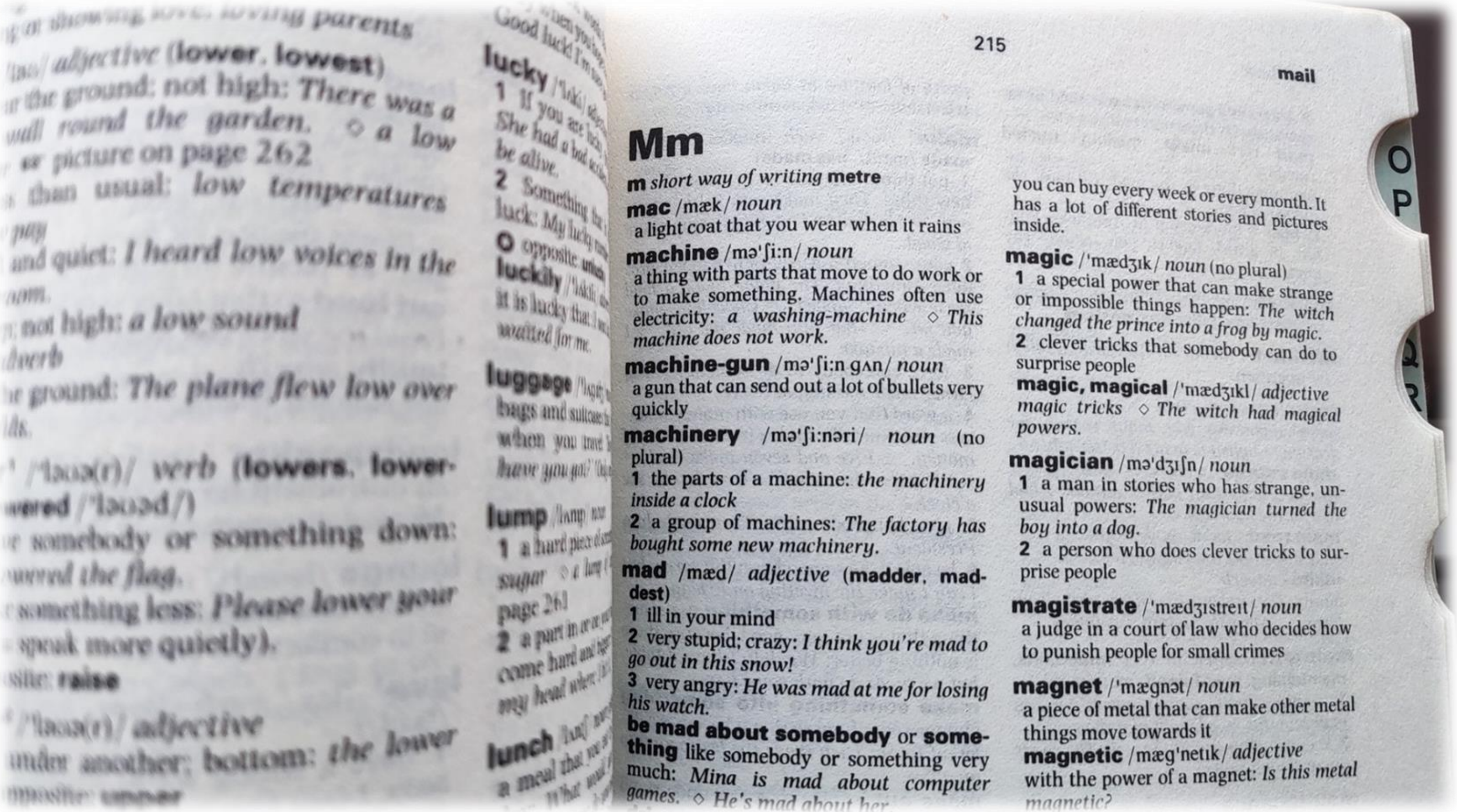
- Tuples are declared by **parenthesis ()**
- Tuples are **ordered** like Lists, but **immutable** unlike them
- Tuples are indexed with integer numbers
- Tuple members (elements) could be of any form (basic data types, composite data types, data structure & objects)
- Tuples are often used for functions that have multiple return values or config variables that shouldn’t be altered

Tuples

- We can't add elements to a Tuple because of its immutable property
  - There's no **append()** or **extend()** method for Tuples
  - However, we can still join two Tuples with a “+” operator

- We can't remove elements from a Tuple, also because of its immutability
  - Tuples have no **remove()** or **pop()** method

Dictionaries



215

mail

Mm

m *short way of writing* **metre**

mac /mæk/ *noun*
a light coat that you wear when it rains

machine /mə'ʃi:n/ *noun*
a thing with parts that move to do work or to make something. Machines often use electricity: *a washing-machine* ◇ *This machine does not work.*

machine-gun /mə'ʃi:n gʌn/ *noun*
a gun that can send out a lot of bullets very quickly

machinery /mə'ʃi:nəri/ *noun* (no plural)
1 the parts of a machine: *the machinery inside a clock*
2 a group of machines: *The factory has bought some new machinery.*

mad /mæd/ *adjective* (**madder**, **maddest**)
1 ill in your mind
2 very stupid; crazy: *I think you're mad to go out in this snow!*
3 very angry: *He was mad at me for losing his watch.*
be mad about somebody or **something** like somebody or something very much: *Mina is mad about computer games.* ◇ *He's mad about her.*

you can buy every week or every month. It has a lot of different stories and pictures inside.

magic /'mædʒɪk/ *noun* (no plural)
1 a special power that can make strange or impossible things happen: *The witch changed the prince into a frog by magic.*
2 clever tricks that somebody can do to surprise people

magic, magical /'mædʒɪkl/ *adjective*
magic tricks ◇ *The witch had magical powers.*

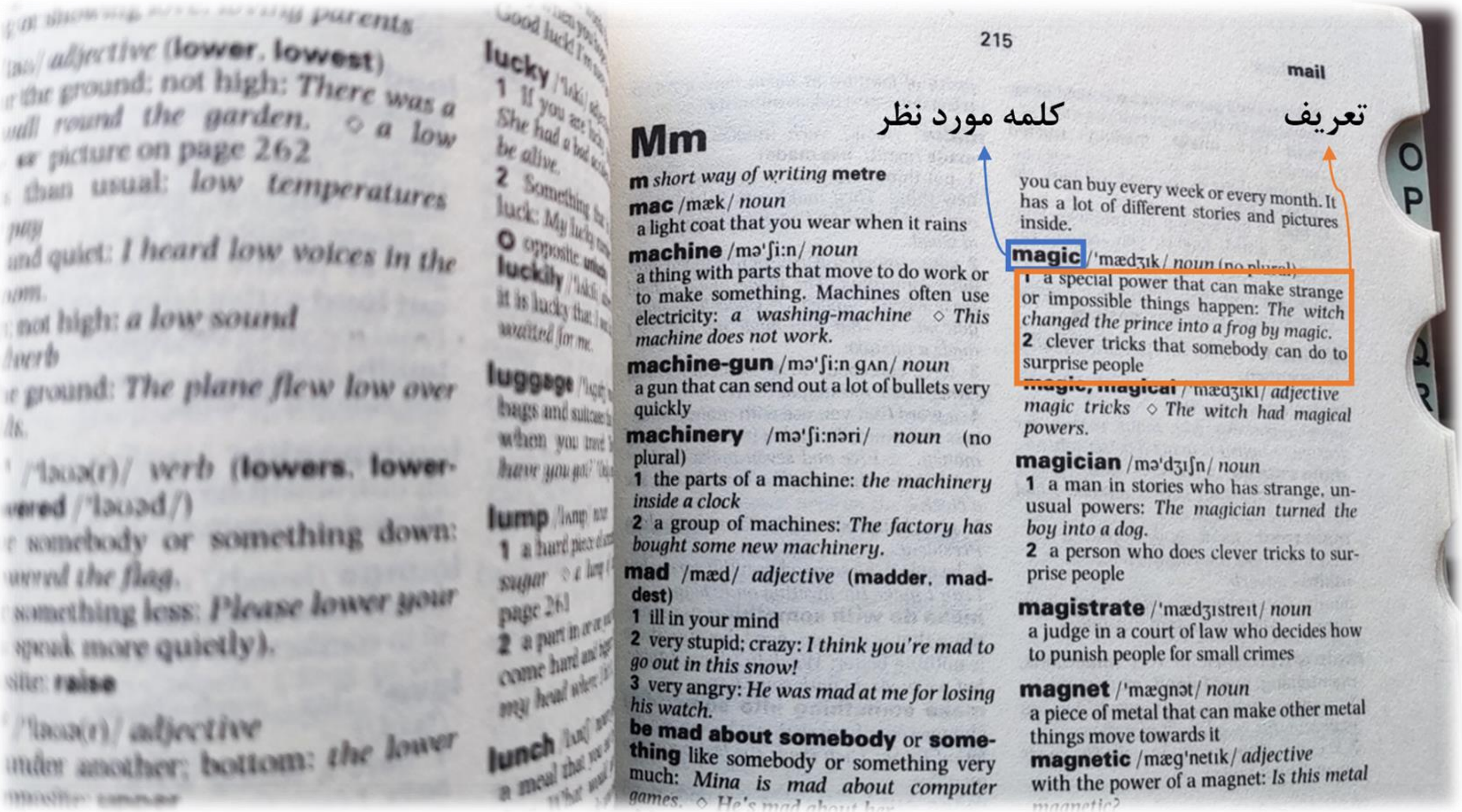
magician /mə'dʒɪʃn/ *noun*
1 a man in stories who has strange, unusual powers: *The magician turned the boy into a dog.*
2 a person who does clever tricks to surprise people

magistrate /'mædʒɪstreɪt/ *noun*
a judge in a court of law who decides how to punish people for small crimes

magnet /'mægnət/ *noun*
a piece of metal that can make other metal things move towards it

magnetic /mæg'netɪk/ *adjective*
with the power of a magnet: *Is this metal magnetic?*

Dictionaries



...adjective (lower, lowest)

...the ground; not high: There was a ...wall round the garden. ◇ a low ...picture on page 262

...than usual: low temperatures

...and quiet: I heard low voices in the ...room.

...not high: a low sound

...verb

...ground: The plane flew low over ...

...¹ /ˈləʊə(r)/ verb (lowers, lowered /ˈləʊəd/)

...somebody or something down: ...wered the flag.

...something less: Please lower your ...speak more quietly).

...site: raise

...² /ˈləʊə(r)/ adjective

...under another; bottom: the lower

lucky /ˈlʌki/ adjective

luckily

luggage

lump

lunch

215

mail

Mm

m short way of writing metre

mac /mæk/ noun
a light coat that you wear when it rains

machine /məˈʃiːn/ noun
a thing with parts that move to do work or to make something. Machines often use electricity: *a washing-machine* ◇ *This machine does not work.*

machine-gun /məˈʃiːn ɡʌn/ noun
a gun that can send out a lot of bullets very quickly

machinery /məˈʃiːnəri/ noun (no plural)
1 the parts of a machine: *the machinery inside a clock*
2 a group of machines: *The factory has bought some new machinery.*

mad /mæd/ adjective (**madder**, **maddest**)
1 ill in your mind
2 very stupid; crazy: *I think you're mad to go out in this snow!*
3 very angry: *He was mad at me for losing his watch.*
be mad about somebody or **something** like somebody or something very much: *Mina is mad about computer games.* ◇ *He's mad about her.*

you can buy every week or every month. It has a lot of different stories and pictures inside.

magic /ˈmædʒɪk/ noun (no plural)
1 a special power that can make strange or impossible things happen: *The witch changed the prince into a frog by magic.*
2 clever tricks that somebody can do to surprise people

magic, magical /ˈmædʒɪkl/ adjective
magic tricks ◇ *The witch had magical powers.*

magician /məˈdʒɪʃn/ noun
1 a man in stories who has strange, unusual powers: *The magician turned the boy into a dog.*
2 a person who does clever tricks to surprise people

magistrate /ˈmædʒɪstreɪt/ noun
a judge in a court of law who decides how to punish people for small crimes

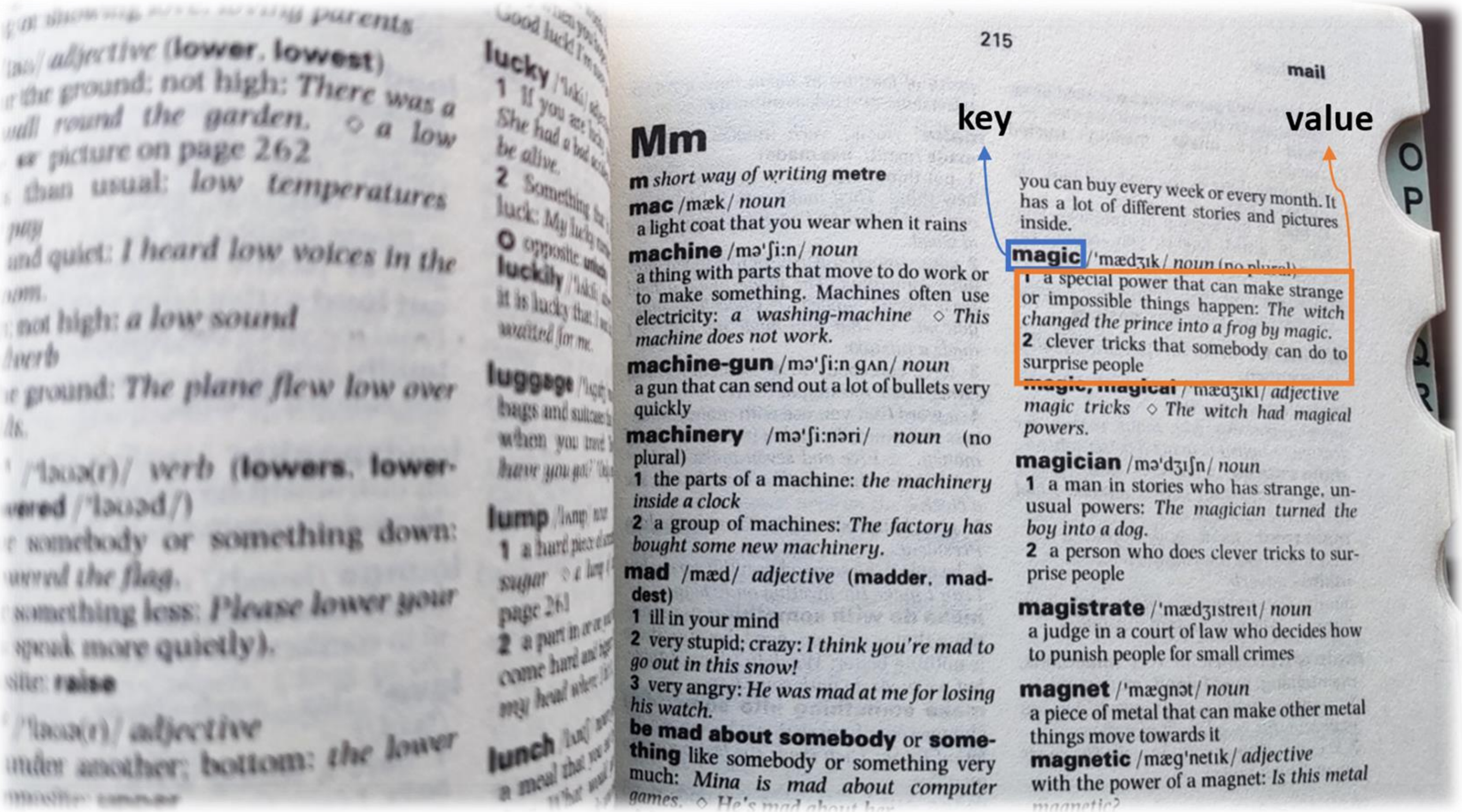
magnet /ˈmæɡnət/ noun
a piece of metal that can make other metal things move towards it

magnetic /mæɡˈnetɪk/ adjective
with the power of a magnet: *Is this metal magnetic?*

کلمه مورد نظر

تعریف

Dictionaries



215

mail

Mm

m *short way of writing* **metre**

mac /mæk/ *noun*
a light coat that you wear when it rains

machine /mə'ʃi:n/ *noun*
a thing with parts that move to do work or to make something. Machines often use electricity: *a washing-machine* ◇ *This machine does not work.*

machine-gun /mə'ʃi:n gʌn/ *noun*
a gun that can send out a lot of bullets very quickly

machinery /mə'ʃi:nəri/ *noun* (no plural)
1 the parts of a machine: *the machinery inside a clock*
2 a group of machines: *The factory has bought some new machinery.*

mad /mæd/ *adjective* (**madder**, **maddest**)
1 ill in your mind
2 very stupid; crazy: *I think you're mad to go out in this snow!*
3 very angry: *He was mad at me for losing his watch.*
be mad about somebody or **something** like somebody or something very much: *Mina is mad about computer games.* ◇ *He's mad about her.*

you can buy every week or every month. It has a lot of different stories and pictures inside.

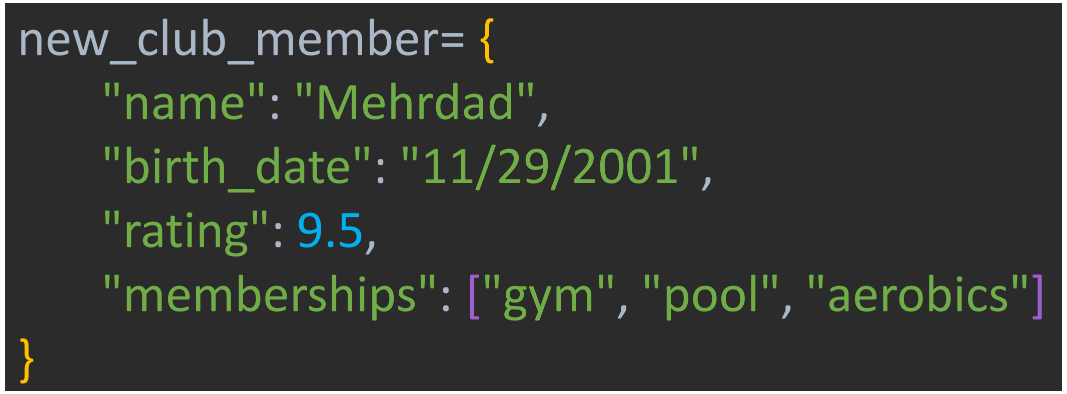
magic /'mædʒɪk/ *noun* (no plural)
1 a special power that can make strange or impossible things happen: *The witch changed the prince into a frog by magic.*
2 clever tricks that somebody can do to surprise people
magic, magical /'mædʒɪkl/ *adjective*
magic tricks ◇ *The witch had magical powers.*

magician /mə'dʒɪʃn/ *noun*
1 a man in stories who has strange, unusual powers: *The magician turned the boy into a dog.*
2 a person who does clever tricks to surprise people

magistrate /'mædʒɪstreɪt/ *noun*
a judge in a court of law who decides how to punish people for small crimes

magnet /'mægnət/ *noun*
a piece of metal that can make other metal things move towards it
magnetic /mæg'netɪk/ *adjective*
with the power of a magnet: *Is this metal magnetic?*

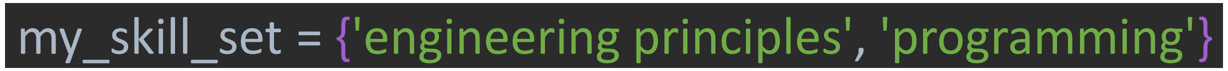
Dictionaries



```
new_club_member= {
     "name": "Mehrdad",
     "birth_date": "11/29/2001",
     "rating": 9.5,
     "memberships": ["gym", "pool", "aerobics"]
}
```

- Dictionaries are declared by **curly braces {}**
- Dictionaries are a set of **<key:value>** pairs
- Dictionaries are **unordered** and **mutable**
- Dictionaries **values** are accessed by **keys** (as index)
- Dictionary **keys are unique** in the entire set (**No duplicate key exists**)
- **Strings** and **Numbers** are the two most commonly used data types as dictionary keys
- Dictionary members (elements) could be of any form (basic data types, composite data types, data structure & objects)

Sets



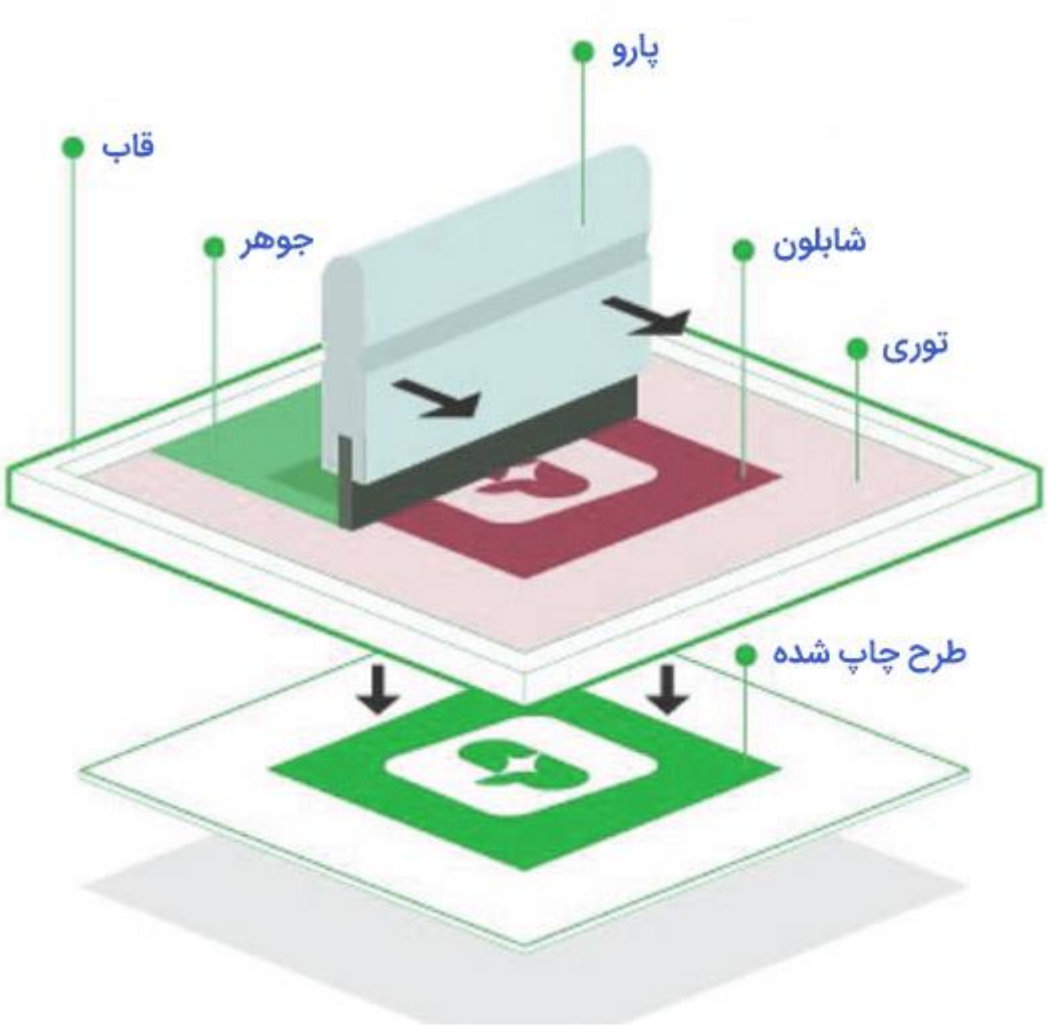
```
my_skill_set = {'engineering principles', 'programming'}
```

- Sets are also declared by **curly braces {}**, just like Dictionaries
- Sets are **unordered** and **mutable**, just like Dictionaries; But they can only hold **unique values** (**No duplicate value exists**)
- You cannot access items in a Set by referring to an index, since Sets are unordered and the items have no index
- Set items can be accessed by using a **for loop**, or ask if a specified value is present in a Set, by using the **in** keyword
- Elements of a set can only be of basic data types, such as integers, floats, and strings, or immutable objects like Tuples
- Sets are highly useful to **efficiently remove duplicate values** from a collection like a List, and to perform common math operations like **unions** and **intersections**

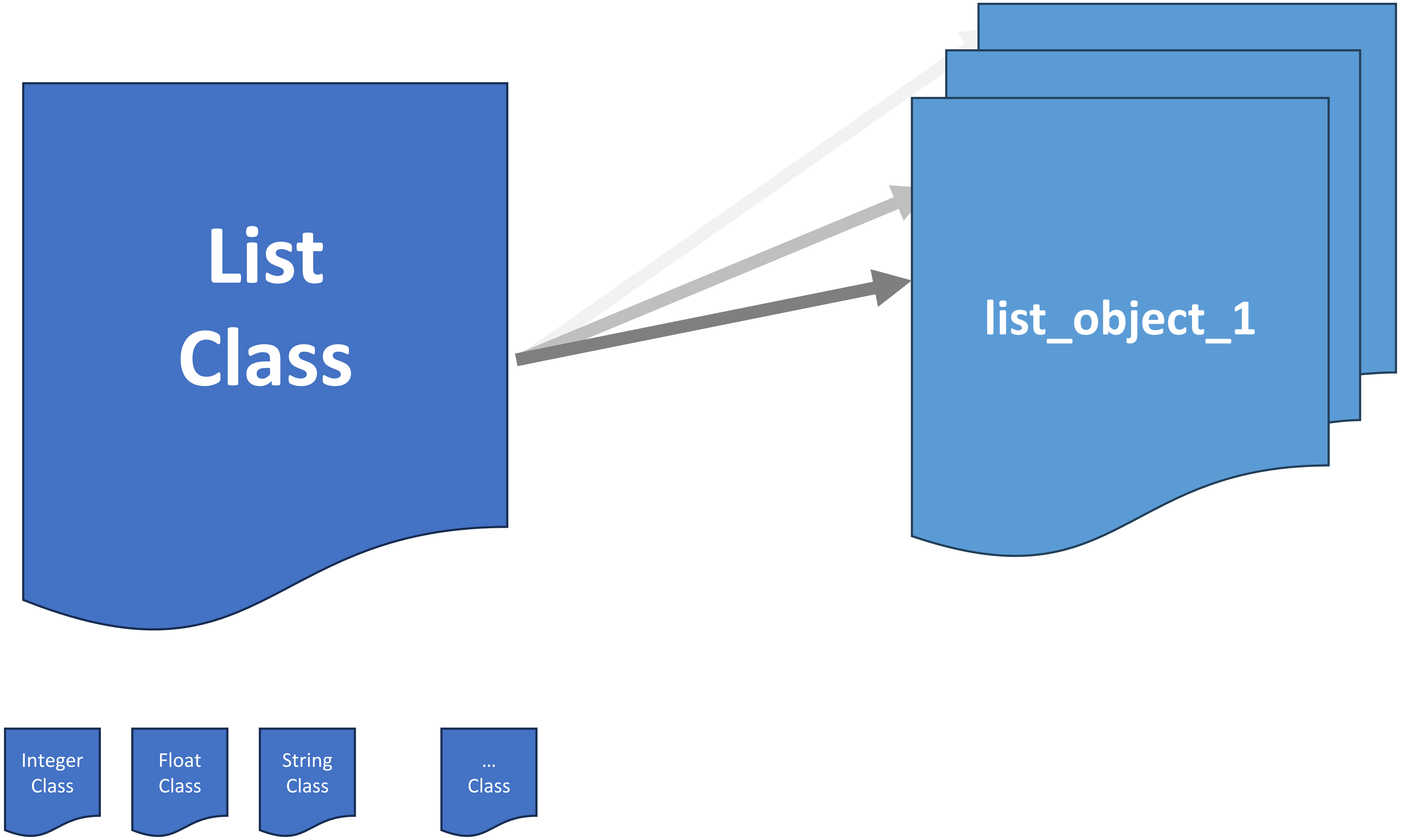
Python Data Types Comparison

Data Type	Declaration Syntax	Ordered	Mutable	Indexed	Can Have Duplicate Keys/IDs	Can Have Duplicate Values	Value Type
List	[]	✓	✓	✓ (int ID)	X	✓	Variable Types Data Types Data Structures
Tuple	()	✓	X	✓ (int ID)	X	✓	Variable Types Data Types Data Structures
Dictionary	{}	X	✓	✓ (immutable Key)	X	✓	Variable Types Data Types Data Structures
Set	{}	X	✓	X	-	X	Variable Types Only + Tuple

What are Objects?



What are Objects?



List Class

list_object_1

Integer Class

Float Class

String Class

… Class

Lists Functions in Python

Function	Description
append()	Adds an element to the end of the list
extend()	Adds all elements of a list to another list
insert()	Inserts an item at the defined index
remove()	Removes an item from the list
clear()	Removes all items from the list
pop()	Removes and returns the last value from the List or the given index value
index()	Returns the index of the first matched item
count()	Returns the count of the number of items passed as an argument
sort()	Sorts items in a list in ascending order
reverse()	Reverses the order of items in the list
copy()	Returns a copy of the list

Lists Functions in Python

Function	Description
reduce()	Applies a particular function passed in its argument, to all of the list elements the list elements mentioned in the sequence passed along
sum()	Sums up the numbers in an iterable
ord()	Returns an integer representing the Unicode code point of the given Unicode character
max()	Returns the maximum value of elements in an iterable
min()	Returns the minimum value of elements in an iterable
all()	Returns True if all elements of an iterable are True (even if an empty one is given)
any()	Returns True if any element of an iterable is True (returns False if an empty one is given)
len()	Returns length/size of the object
enumerate()	Returns enumerate object of the iterable
accumulate()	Applies a particular function passed in its argument to all of the list elements, and returns a list containing the intermediate results
filter()	Tests if each element of a list is True or not
map()	Returns a list of the results after applying the given function to each item of a given iterable
lambda()	This function can have any number of arguments but only one expression, which is evaluated and returned

Tuples Functions in Python

Function	Description
index()	Finds an element in an iterable and returns the index of where it’s located
count()	Returns the frequency of occurrence of a specified value

Function	Description
all()	Returns True if all elements of an iterable are True (even if an empty one is given)
any()	Returns True if any element of an iterable is True (returns False if an empty one is given)
len()	Returns length/size of the object
enumerate()	Returns enumerate object of the iterable
max()	Returns the maximum value of elements in an iterable
min()	Returns the minimum value of elements in an iterable
sum()	Sums up the numbers in an iterable
sorted()	Returns a sorted list of an iterable
tuple()	Converts an iterable to a tuple

Dictionaries Functions in Python

Method	Description
dict.clear()	Removes all the elements from the dictionary
dict.copy()	Returns a copy of the dictionary
dict.get(key, default = “None”)	Returns the value of specified key
dict.items()	Returns a list containing a tuple for each key-value pair
dict.keys()	Returns a list containing dictionary’s keys
dict.update(dict2)	Updates dictionary with specified key-value pairs
dict.values()	Returns a list containing all the values of dictionary
dict.pop()	Removes the element with a specified key
dict.popItem()	Removes the last inserted key-value pair
dict.setdefault(key,default= “None”)	Used to set the key to the default value if the key is not specified in the dictionary
dict.get(key, default = “None”)	Used to get the value specified for the passed key

Sets Functions in Python

Operators	Notes
key in s	containment check
key not in s	non-containment check
s1 == s2	s1 is equivalent to s2
s1 != s2	s1 is not equivalent to s2
s1 <= s2	s1 is subset of s2
s1 < s2	s1 is proper subset of s2
s1 >= s2	s1 is superset of s2
s1 > s2	s1 is proper superset of s2
s1 | s2	the union of s1 and s2
s1 & s2	the intersection of s1 and s2
s1 – s2	the set of elements in s1 but not s2
s1 ˆ s2	the set of elements in precisely one of s1 or s2

Tuples vs. Lists in Python

Similarities	Differences
Functions that can be used for both Lists and Tuples: len(), max(), min(), sum(), any(), all(), sorted()	Methods that cannot be used for Tuples: append(), insert(), remove(), pop(), clear(), sort(), reverse()
Methods that can be used for both Lists and Tuples: count(), Index()	we *generally* use ‘Tuples’ for heterogeneous (different) data types and ‘Lists’ for homogeneous (similar) data types
Tuples can be stored in Lists	Iterating through a ‘Tuple’ is faster than in a ‘List’
Lists can be stored in Tuples	‘Lists’ are mutable whereas ‘Tuples’ are immutable
Both ‘Tuples’ and ‘Lists’ can be nested	Tuples that contain immutable elements can be used as a key for a Dictionary

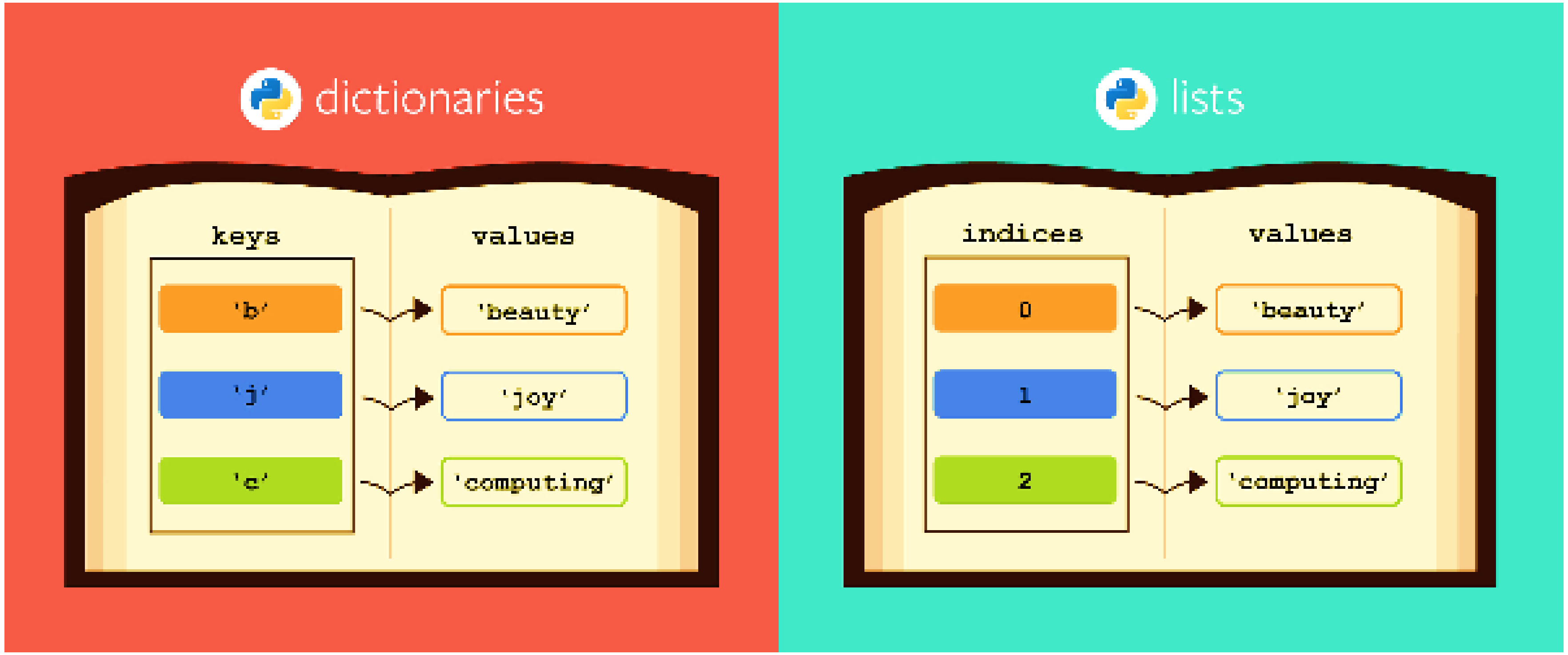
Dictionaries VS Lists in Python

Unordered

Has Keys

Ordered

Indexed



keys	values
'b'	'beauty'
'j'	'joy'
'c'	'computing'

indices	values
0	'beauty'
1	'joy'
2	'computing'

When to Use Each Data Type?

Lists VS Tuples	
Similarity	• Tuples and Lists are both used to store the collection of data • Tuples and Lists are both heterogeneous data types, meaning that you can store any kind of data type/structure inside them • Tuples and Lists are both ordered means the order in which you put the items is kept • Tuples and Lists are both sequential data types so you can iterate over the items contained • Items of both Tuples and Lists can be accessed by an integer index operator
Difference	• Need mutable and ordered data structure ⇨ List / Need immutable and ordered data structure ⇨ Tuple • A container to hold multiple objects as one ⇨ Tuple is preferred over List (if the data shouldn't change) • Tuples can be used as dictionary keys • Tuples are faster than Lists (Tuples have a slight advantage over the Lists especially when we consider lookup value) • Tuples are more memory efficient than the Lists

When to Use Each Data Type?

	Lists VS Dictionaries
Similarity	• Dictionaries and Lists are both used to store the collection of data with duplicate values (not keys) • Dictionaries and Lists are both heterogeneous data types, meaning that you can store any kind of data type/structure inside them
Difference	• Need mutable and ordered data structure ⇨ List / Need mutable and unordered data structure ⇨ Dictionary • Lists are used to store the data, which should be ordered and sequential. On the other hand, Dictionary is used to store large amounts of data for easy and quick access. • The indices of the Lists are integers starting from 0. The keys of the Dictionaries can be of any **immutable** data type (strings, numbers, or tuples). • The Lists elements are accessed via indices. The Dictionaries elements are accessed via key-value pairs. • The order of the elements entered in Lists is maintained. There is no guarantee for maintaining the order of elements entered in Dictionaries. • Dictionaries are faster than Lists for the lookup of elements because it takes less time to traverse in the dictionary than a list. • It will take more time to fetch a single element in a List than that in a Dictionary.

When to Use Each Data Type?

Lists VS Sets	
Similarity	• Lists and Sets are both used to store the collection of data • Lists and Sets are both heterogeneous data types: • Lists can store any kind of data type inside them (Basic Data Types, Composite Data Types, Data Structures) • Sets can store any kind of immutable types (Basic Data Types, Tuple)
Difference	• Need mutable and ordered data structure ⇨ List / Need mutable and unordered data structure ⇨ Set • Sets cannot have duplicate values. All values must be unique. Lists can have duplicate values. • Items of a Set can’t be changed but can be added to and removed from it. Items of a List can be changed, added to, and removed from it. • The Lists elements are accessed via indices. • The order of the elements entered in Lists is maintained. There is no guarantee for maintaining order of elements entered in Sets. • Sets are significantly faster when it comes to determining if an object is present in it than iterating through Lists.

When to Use Each Data Type?

	Tuples VS Dictionaries
Similarity	• Tuples and Dictionaries are both used to store the collection of data with duplicate values (not keys) • Tuples and Dictionaries are both heterogeneous data types, meaning that you can store any kind of data type/structure inside them
Difference	• Need immutable and ordered data structure ⇨ Tuple / Need mutable and unordered data structure ⇨ Dictionary • Tuples are used to store the data which is not intended to change. The values of Dictionaries can be changed. • The indices of the Tuples are integers starting from 0. The keys of the Dictionaries can be of any immutable data type (strings, numbers, or tuples). • The Tuples elements are accessed via indices. The Dictionaries elements are accessed via key-value pairs. • The order of the elements entered in Tuples is maintained. There is no guarantee for maintaining the order of elements entered in Dictionaries.

When to Use Each Data Type?

Tuples VS Sets	
Similarity	• Tuples and Sets are both used to store the collection of data • Tuples and Sets are both heterogeneous data types: ◦ Tuples can store any kind of data type inside them (Basic Data Types, Composite Data Types, Data Structures) ◦ Sets can store any kind of immutable types (Basic Data Types, Tuple)
Difference	• Need immutable and ordered data structure ⇨ Tuples / Need mutable and unordered data structure ⇨ Set • Sets cannot have duplicate values. All values must be unique. Tuples can have duplicate values. • Items of a Set can't be changed but can be added to and removed from it. Items of a Tuple can't be changed, added to, and removed from it. • The Tuples elements are accessed via indices. • The order of the elements entered in Tuples is maintained. There is no guarantee for maintaining order of elements entered in Sets. • Sets are significantly faster when it comes to determining if an object is present in it than iterating through Tuples.

When to Use Each Data Type?

	Dictionaries VS Sets
Similarity	• Dictionaries and Sets are both used to store the collection of data • Dictionaries and Sets are both heterogeneous data types: • Dictionaries can store any kind of data type inside them (Basic Data Types, Composite Data Types, Data Structures) • Sets can store any kind of immutable types (Basic Data Types, Tuple) • Dictionaries and Sets are both mutable and unordered data structure
Difference	• Sets cannot have duplicate values. All values must be unique. Dictionaries can have duplicate values but cannot have duplicate keys.