Applied Data Science with Python

Istvan Lengyel

ilengyel@eit.ac.nz

Senior Lecturer, EIT

Kamini Garg
kaminigarg32@gmail.com

Lead Data Scientist, UPC Switzerland

This lecture will teach .....

- Functions
- Lists
- Dictionaries & Tuples

Why Functions ?

- Functions allow the same piece of code to run multiple times
- Functions break long programs up into smaller components
- Functions can be shared and used by other programmers

Functions

Named sequence of statements that performs a computation

1. Type-conversion functions

Built-in functions in Python to convert values from one type to another.

```
>>> int('32')
32
>>> int('Hello')
 ValueError: invalid literal for int(): Hello
```

Others are : float, str

Functions

2. Math Functions

Python has a “math” module for mathematical functions.

Need to import it using **import math**

```
>>> math.sqrt(2) / 2.0
0.707106781187
```

3. Adding own functions

Easier to read, test, fix, improve and develop.

Leads to more structured programming.

Functions

```
def sum(value1, value2) :
    total_sum = 0
    total_sum = value1+value2
    return total_sum
```

Example: sum(10,12)
Python makes it easier to provide default argument values

Python module help

Powerful help tools

- Most objects, functions, modules, ... can be inspected

```
>>> help(math)
>>> help(math.cos)
>>> a = [1,2,3]
>>> help(a)
```

Lists

- Lists store a sequence of data which supports indexing.
- Examples

['hydrogen', 'helium', 'lithium', 'beryllium','boron', ..., 'thorium', 'protactinium', 'uranium']

[-3.141592653589793, -1.5707963267948966, 0.0, 1.5707963267948966, 3.141592653589793]

[2, 3, 5, 7, 11, 13, 17, 19]

Lists

```
L = [2, 4, 6, 8]
print(L[3]) # prints ’8’
print (L[-1]) # prints ’8’
print (L[4]) # error
L.append(10)
print (L[4]) # prints ’10’
```

Similar to strings, list support negative indexing.

Lists

- Lists can be non-homogeneous and nested.
- new_list = [2, [3, 5, 6], ’helium’, -6, ’sodium’]

```
>>> print(new_list[2])
>>> ’helium’
```

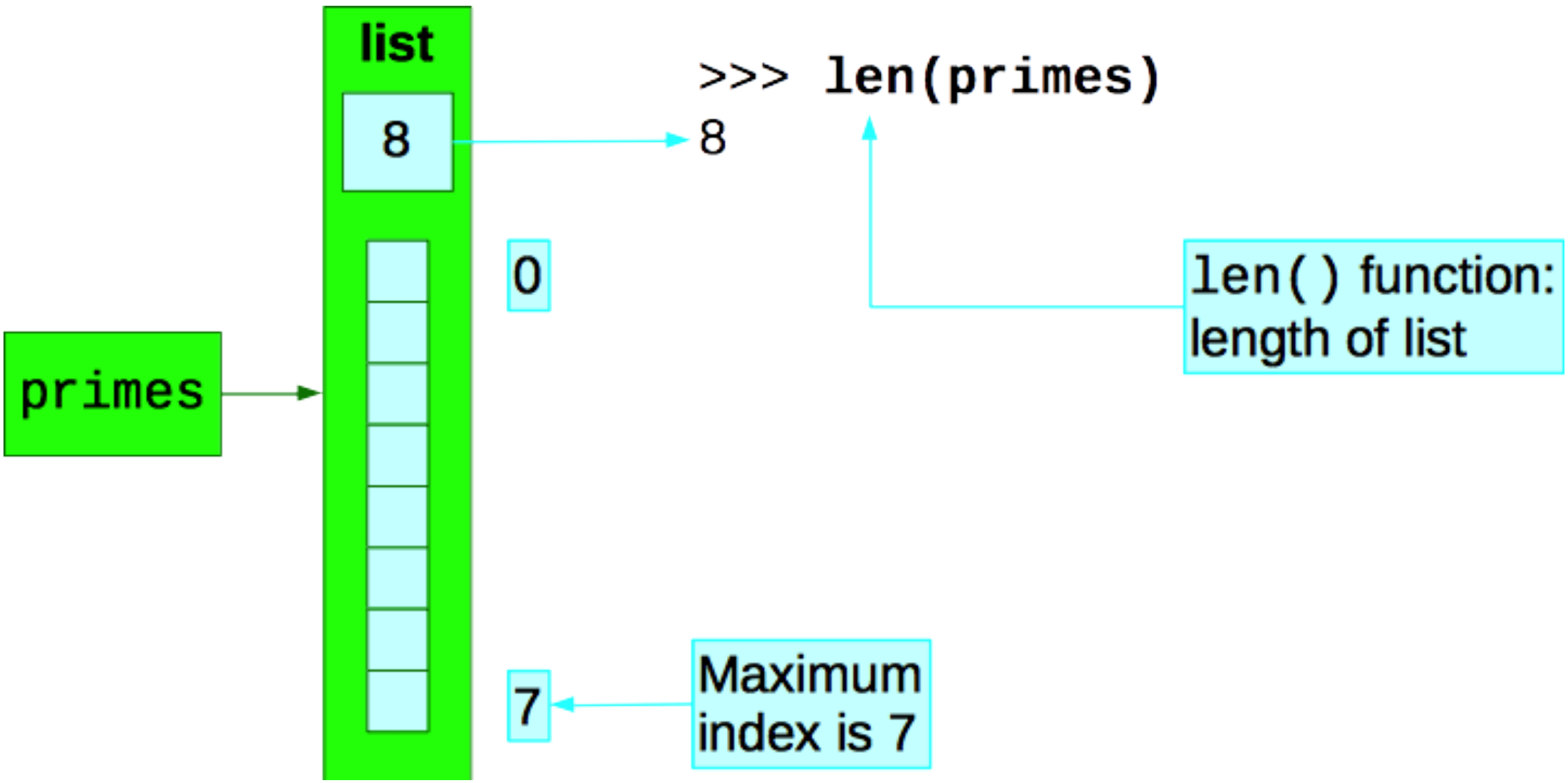
Creating a List

```
>>> primes = [ 2, 3, 5, 7, 11, 13, 17, 19]
```

```
>>> primes
[2, 3, 5, 7, 11, 13, 17, 19]
```

```
>>> type(primes)
<class 'list'>
```

Length of a List



Changing a value in a List

```
>>> data = ['alpha', 'beta', 'gamma']

>>> data[2]
'gamma'

>>> data[2] = 'G'

>>> data[2]
'G'

>>> data
['alpha', 'beta', 'G']
```

Appending to a List

```
>>> primes
[2, 3, 5, 7, 11, 13, 17, 19]
```

A function built into a list

```
>>> primes.append(23)
```

```
>>> primes
[2, 3, 5, 7, 11, 13, 17, 19,
```

The list is now updated

"." a full stop should be placed here

23] should be displayed here

Appending to a List

```
>>> primes
[2, 3, 5, 7, 11, 13, 17, 19]
```

Other methods: sort(), reverse(), insert(), remove(), pop(), min(), max()

Add two lists together

```
L1 = [2, 4, 6, 8, 10]

L2 = [3, 5]

L3 = l1 + l2

L3 = ???

L3 = [2, 4, 6, 8, 10, 3, 5]
```

List slicing and manipulation

```
L = [2, 4, 6, 8]
print(L[0:2]) # prints [2, 4]
L[1:3] = [7, 7]  # L = [2, 7, 7, 8]
print (L[2:]) # prints [7, 8]
L*= 2
print (L[3:6]) # prints [8, 2, 7]
```

List comprehensions

Create new lists by manipulating old ones.

```
my_num = [3, 17, 22, 46, 71, 8]

even_num = [n**2 for n in my_num if n%2 == 0]
```

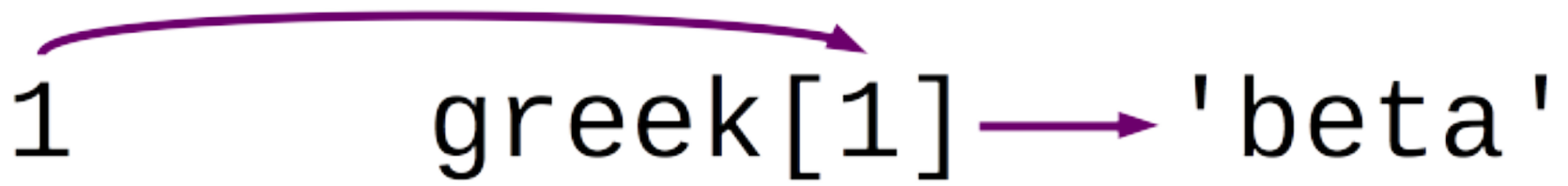
Tuples

Tuples are similar to lists but they are immutable i.e cannot be modified.

```
t1 = ('apple', 1.2, -3.0, 7)
t2 = ('orange', -7.3, -8, -0.0001)
t1[1] = 100   # error
print(t2[0])   ???
```

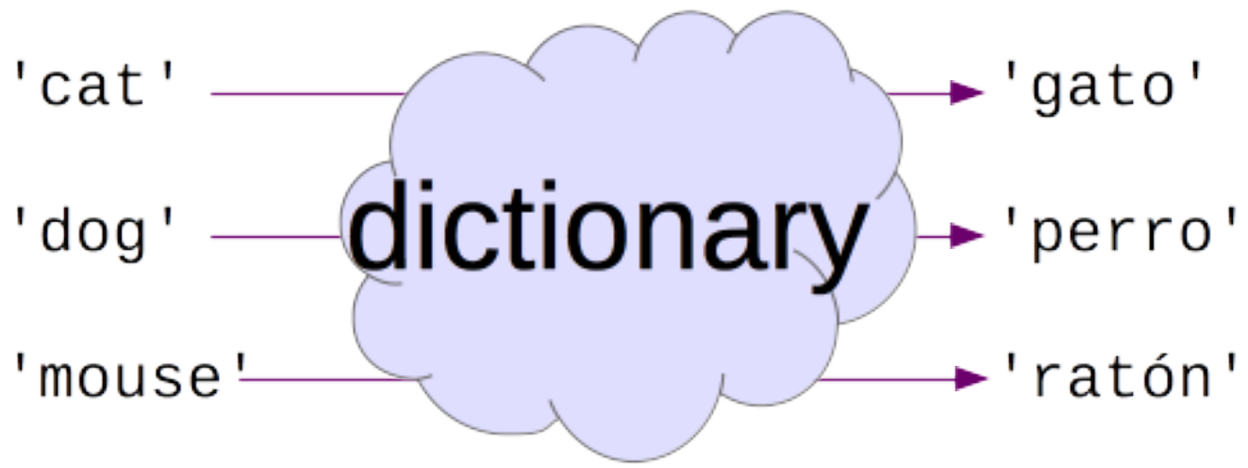
Python Dictionaries

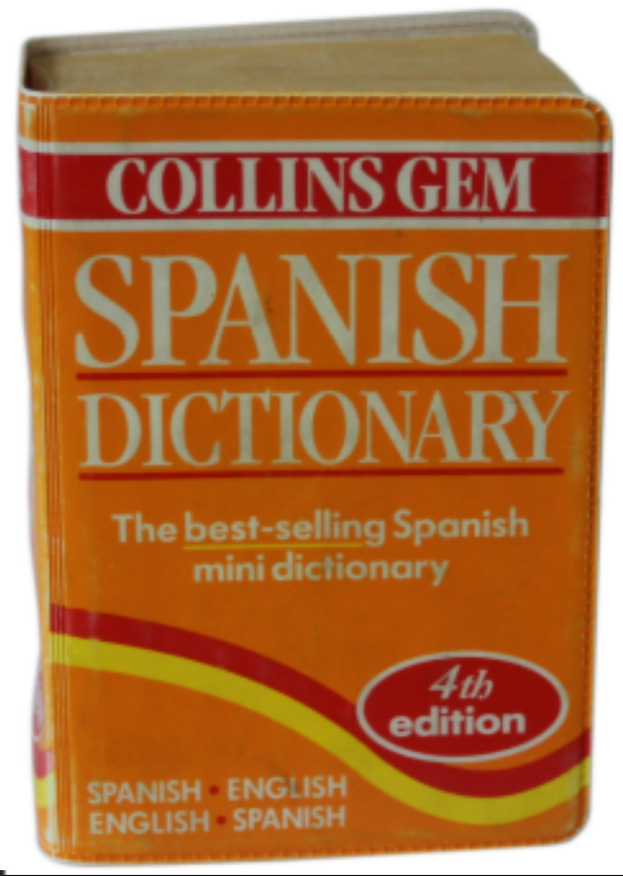
"Index in — Value out"



Python Dictionaries

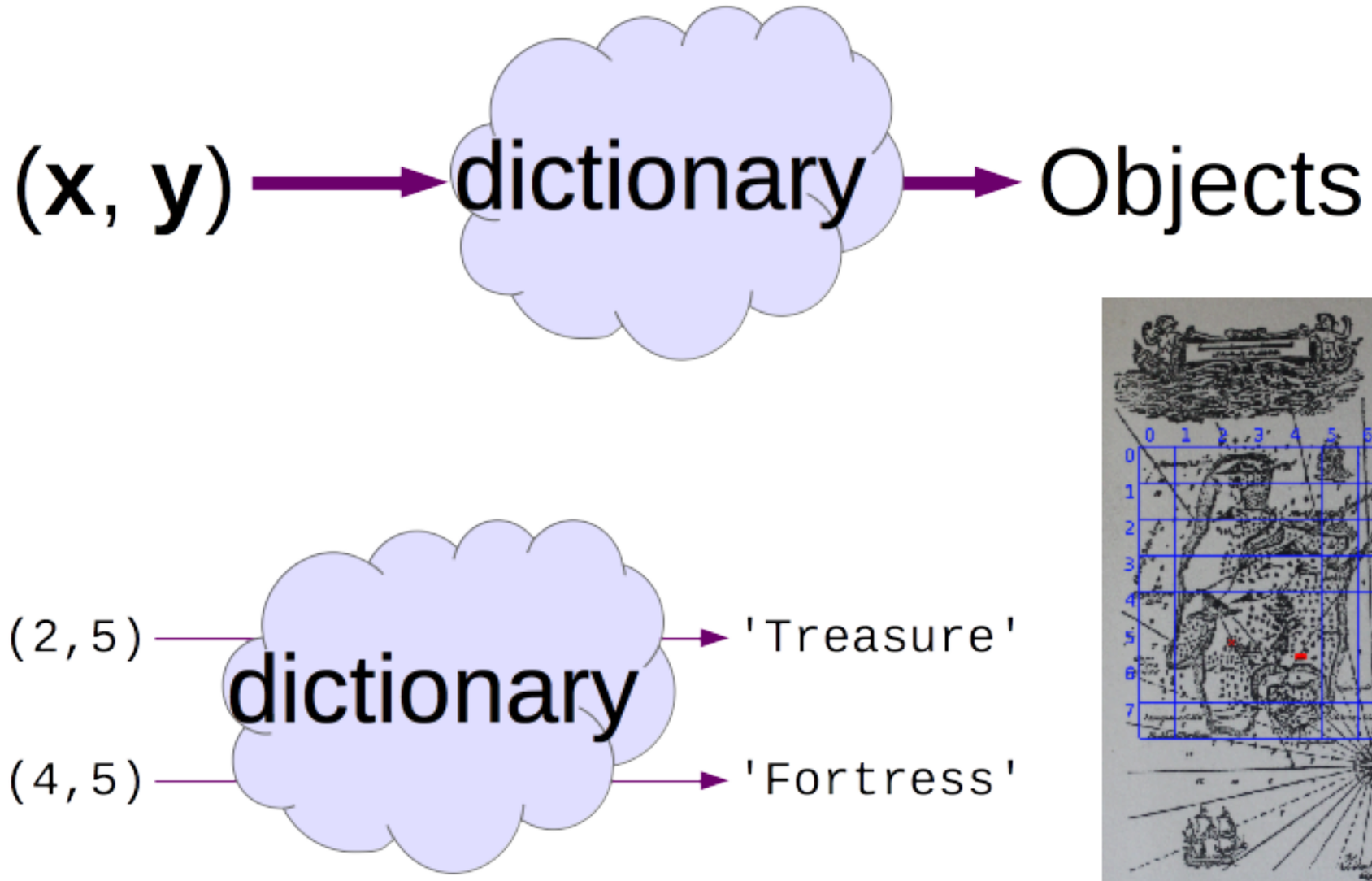
Other “indices”: Strings?





Python Dictionaries

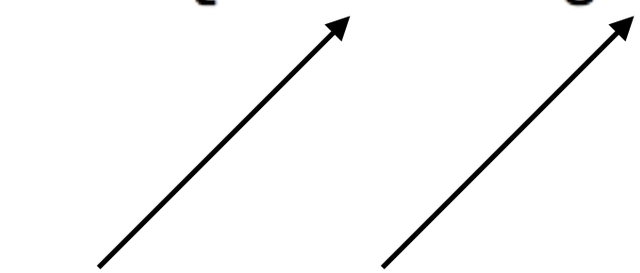
Other “indices”: Tuples?



Dictionaries are maps from a set of keys to a set of values.

Python Dictionaries

```
>>> en_to_es = { 'cat':'gato' , 'dog':'perro' }
```



What's in dictionary ?

```
>>> en_to_es
{'mouse': 'ratón', 'dog': 'perror', 'cat': 'gato'}
```

```
>>> en_to_es.keys()
dict_keys(['mouse', 'dog', 'cat'])
```

```
>>> en_to_es.values()
dict_values(['ratón', 'perro', 'gato'])
```

What's in dictionary ?

```
>>> en_to_es.items()
```

Most useful method

```
dict_items([('mouse', 'ratón'), ('dog', 'perro'),
('cat', 'gato')])
```

(Key,Value) pairs/tuples

```
>>> list(en_to_es.items())
[('mouse','ratón'), ('dog','perro'),('cat','gato')]
```

Python Dictionaries

Accessing keys:

```
en_to_es['cat'] = 'gato'
```

Overwriting keys:

```
en_to_es['cat'] = 'Katze'
```

Removing key-value pair:

```
del en_to_es['cat']
```

Python Dictionaries : in and for

Print all keys:

```
for key in en_to_es.keys():
    print(key)
```

Check if key k exists:

```
if k not in en_to_es.keys():
    print(“Key does not exist!!”)
```

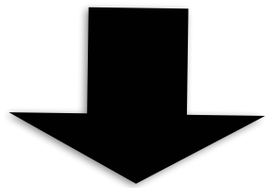
Exercises

1. Write a function for adding elements in a list called *numbers* and print the result.

numbers = [45, 76, -23, 90, 15]

2. Create histogram of words:

```
words = [‘the’,’cat','sat','on','the','mat']
```



```
counts = {‘the':2,'cat':1,'sat':1,'on':1,'mat':1}
```