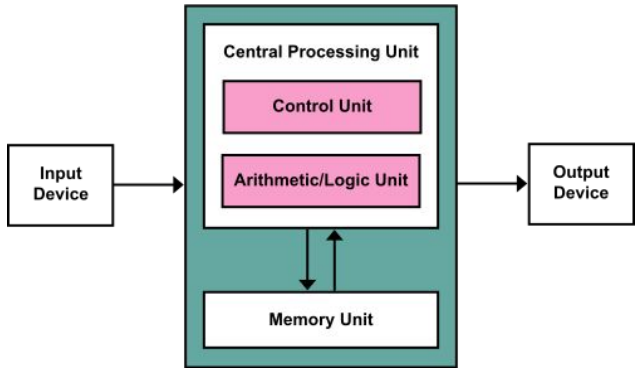
Corso di STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI

UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Modulo di Sistemi di Elaborazione delle Informazioni

Dizionari e Set

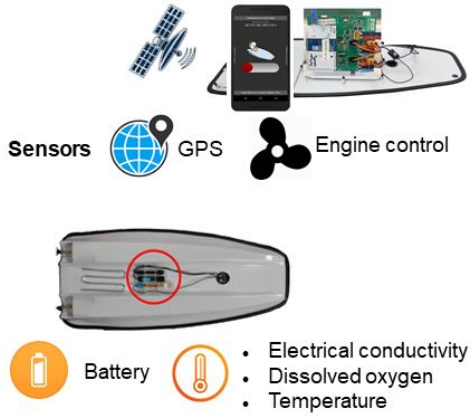
Docente:
Domenico Daniele Bloisi

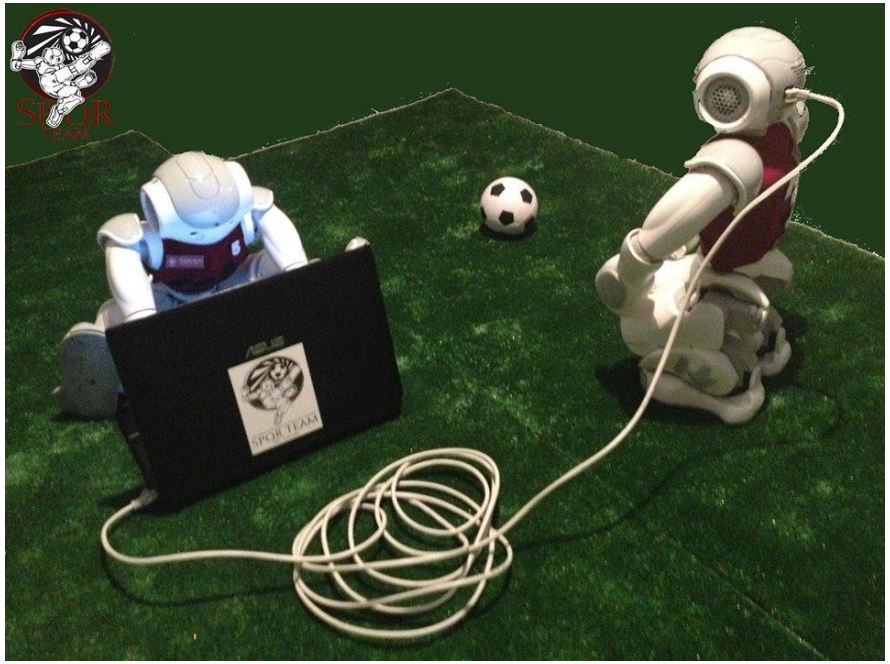


Domenico Daniele Bloisi

- Professore Associato
  Dipartimento di Matematica, Informatica ed Economia
  Università degli studi della Basilicata
  http://web.unibas.it/bloisi

- SPQR Robot Soccer Team
  Dipartimento di Informatica, Automatica e Gestionale Università degli studi di Roma “La Sapienza”
  http://spqr.diag.uniroma1.it



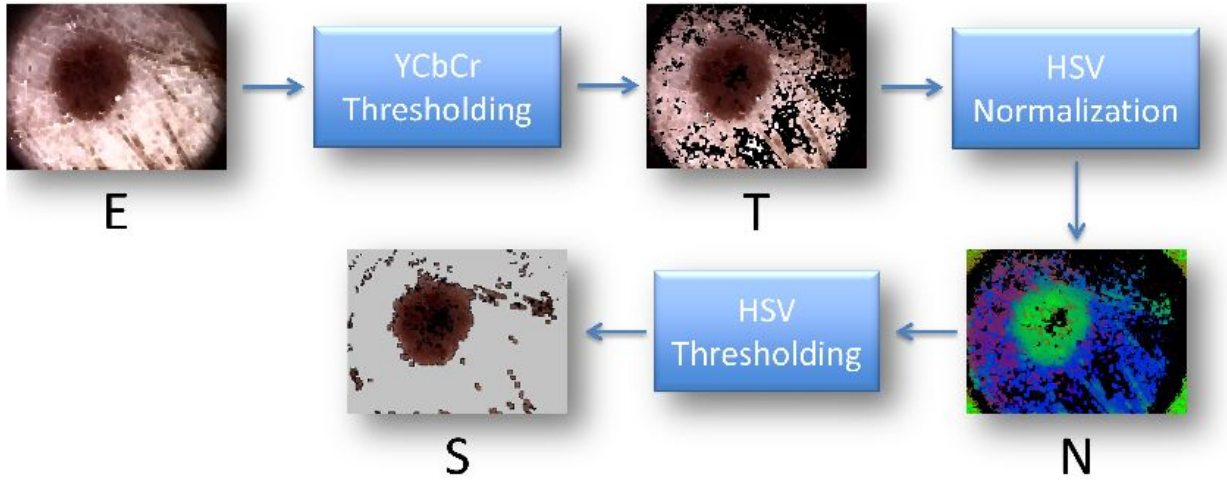


Interessi di ricerca

- Intelligent surveillance
- Robot vision
- Medical image analysis

https://youtu.be/2KHNZX7UIWQ

https://youtu.be/9a70Ucgbi_U



UNIBAS Wolves https://sites.google.com/unibas.it/wolves

- UNIBAS WOLVES is the robot soccer team of the University of Basilicata. Established in 2019, it is focussed on developing software for NAO soccer robots participating in RoboCup competitions.

- UNIBAS WOLVES team is twinned with SPQR Team at Sapienza University of Rome

https://youtu.be/ji0OmkaWh20

Informazioni sul corso

Il corso di STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI

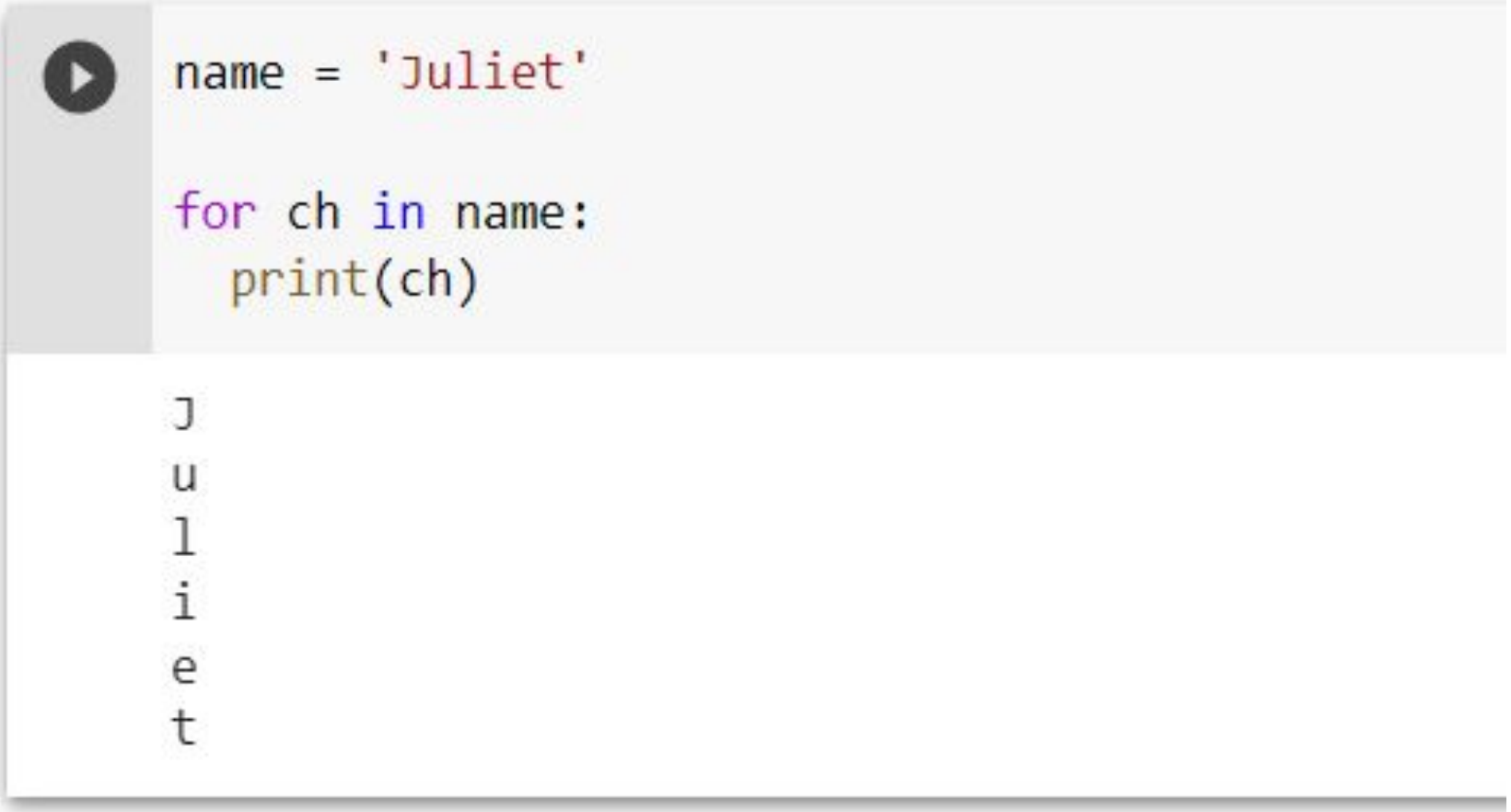
- include 3 moduli:
  - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI
    (il martedì - docente: Domenico Bloisi)
  - INFORMATICA
    (il mercoledì - docente: Enzo Veltri)
  - PROBABILITA' E STATISTICA MATEMATICA
    (il giovedì - docente: Antonella Iuliano)
- Periodo: I semestre ottobre 2022 – gennaio 2023

Ricevimento Bloisi

- In presenza, durante il periodo delle lezioni:
  Lunedì dalle 17:00 alle 18:00
  presso Edificio 3D, II piano, stanza 15
  Si invitano gli studenti a controllare regolarmente la bacheca degli avvisi per eventuali variazioni
- Tramite google Meet e al di fuori del periodo delle lezioni:
  da concordare con il docente tramite email

Per prenotare un appuntamento inviare una email a
domenico.bloisi@unibas.it

Recap



```
name = 'Juliet'

for ch in name:
  print(ch)
```

```
J
u
l
i
e
t
```

Accessing the Individual Characters in a String (2 of 4)

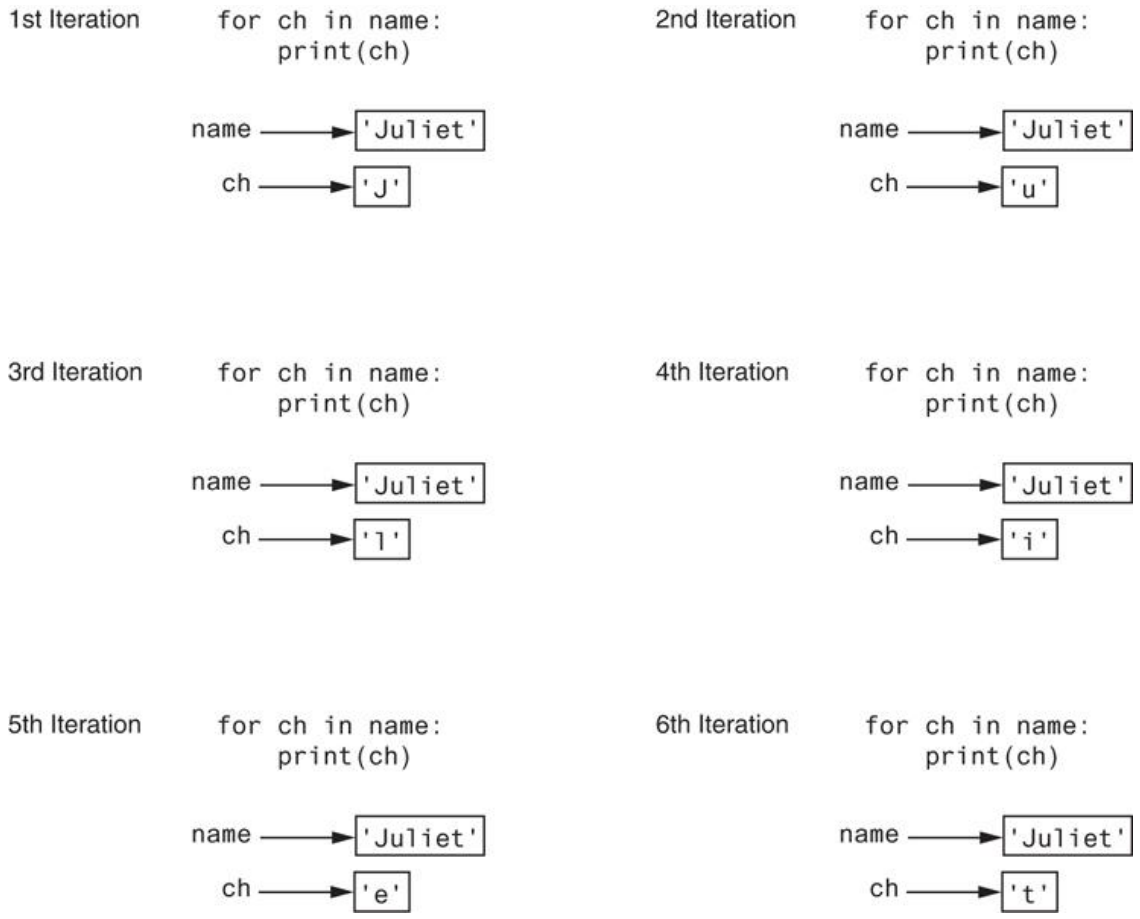


Figure 8-1 Iterating over the string 'Juliet'

Accessing the Individual Characters in a String (3 of 4)

```
my_string = "Roses are red"

ch = my_string[6]
```

Accessing the Individual Characters in a String (3 of 4)

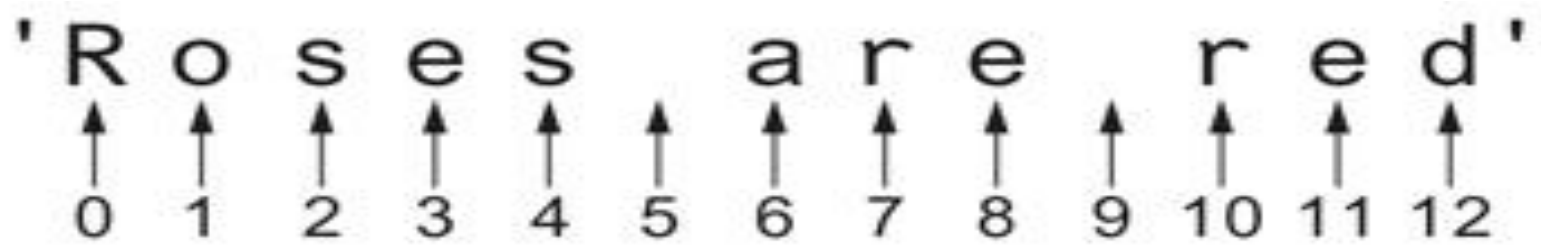


Figure 8-2 String indexes

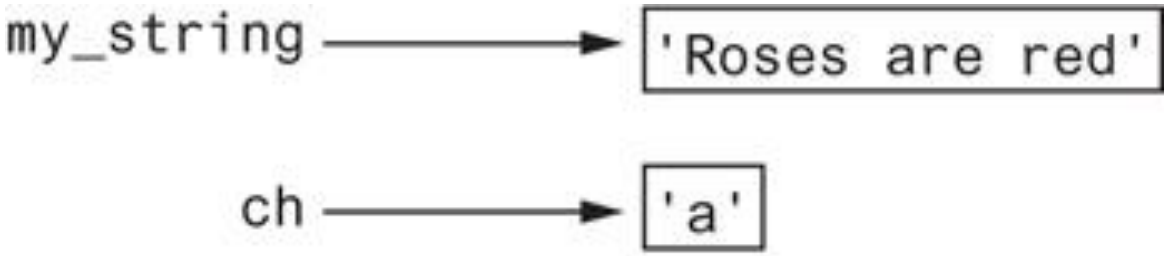


Figure 8-3 Getting a copy of a character from a string

```
[6] message = "Hello" + "World"

    print(message)
```

```
HelloWorld
```

```
message = "Hello " + "World"

print(message)
```

```
Hello World
```

```
s = "Mario"

ch = s[2]

print(ch)

s[2] = "V"
```

```
r
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-ba3e49c0a7b9> in <module>
      5 print(ch)
      6
----> 7 s[2] = "V"

TypeError: 'str' object does not support item assignment
```

SEARCH STACK OVERFLOW

```
[11] full_name = "Patty  Lynn Smith"

     middle_name = full_name[6:10]

     print(middle_name)
```

```
 Lyn
```

```
[12] first_name = full_name[:5]

     print(first_name)
```

```
Patty
```

```
last_name = full_name[11:]

print(last_name)
```

```
 Smith
```

String Methods (4 of 7)

Table 8-2 String Modification Methods

Method	Description
`lower()`	Returns a copy of the string with all alphabetic letters converted to lowercase. Any character that is already lowercase, or is not an alphabetic letter, is unchanged.
`lstrip()`	Returns a copy of the string with all leading whitespace characters removed. Leading whitespace characters are spaces, newlines (`\n`), and tabs (`\t`) that appear at the beginning of the string.
`lstrip(`*`char`*`)`	The *`char`* argument is a string containing a character. Returns a copy of the string with all instances of *`char`* that appear at the beginning of the string removed.
`rstrip()`	Returns a copy of the string with all trailing whitespace characters removed. Trailing whitespace characters are spaces, newlines (`\n`), and tabs (`\t`) that appear at the end of the string.
`rstrip(`*`char`*`)`	The *`char`* argument is a string containing a character. The method returns a copy of the string with all instances of *`char`* that appear at the end of the string removed.
`strip()`	Returns a copy of the string with all leading and trailing whitespace characters removed.
`strip(`*`char`*`)`	Returns a copy of the string with all instances of *`char`* that appear at the beginning and the end of the string removed.
`upper()`	Returns a copy of the string with all alphabetic letters converted to uppercase. Any character that is already uppercase, or is not an alphabetic letter, is unchanged.

```
[21] letters = "WSXE"

     print(letters.lower())
```

```
wsxe
```

```
     letters = "yhnj"

     print(letters.upper())
```

```
YHNJ
```

String Methods (7 of 7)

Table 8-3 Search and replace methods

Method	Description
`endswith(substring)`	The *substring* argument is a string. The method returns true if the string ends with *substring*.
`find(substring)`	The *substring* argument is a string. The method returns the lowest index in the string where *substring* is found. If *substring* is not found, the method returns −1.
`replace(old, new)`	The *old* and *new* arguments are both strings. The method returns a copy of the string with all instances of *old* replaced by *new*.
`startswith(substring)`	The *substring* argument is a string. The method returns true if the string starts with *substring*.

String Tokens (4 of 4)

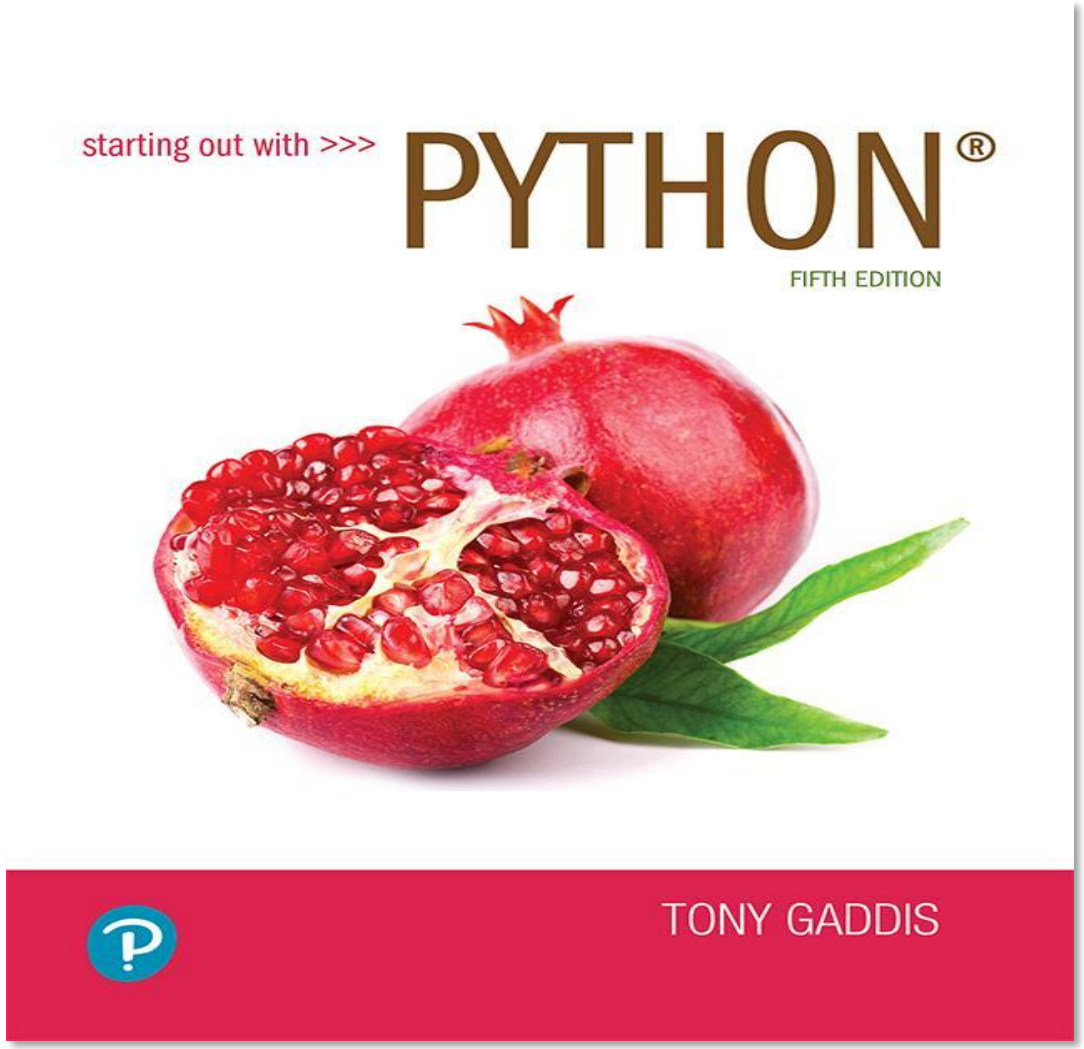
- Examples:

```
>>> str = 'peach raspberry strawberry vanilla'
>>> tokens = str.split()
>>> tokens
['peach', 'raspberry', 'strawberry', 'vanilla']
>>>
```

```
>>> my_address = 'www.example.com'
>>> tokens = my_address.split('.')
>>> tokens
['www', 'example', 'com']
>>>
```

Starting out with Python

Fifth Edition



Chapter 9

Dictionaries and Sets

Topics

- Dictionaries
- Sets
- Serializing Objects

Dictionaries

- Dictionary: object that stores a collection of data
  - Each element consists of a *key* and a *value*
    - Often referred to as *mapping* of key to value
    - Key must be an immutable object
  - To retrieve a specific value, use the key associated with it
  - Format for creating a dictionary

```
dictionary =
    {key1:val1, key2:val2}
```

Dictionaries

```
rubrica = {"Antonio":"323573", "Giuseppe":"322955", "Marina":"3449007"}
```

Retrieving a Value from a Dictionary

- Elements in dictionary are unsorted
- General format for retrieving value from dictionary: *`dictionary[key]`*
  - If `key` in the dictionary, associated value is returned, otherwise, `KeyError` exception is raised
- Test whether a key is in a dictionary using the `in` and `not in` operators
  - Helps prevent `KeyError` exceptions

Retrieving a Value from a Dictionary

```
[1] rubrica = {"Antonio":"323573", "Giuseppe":"322955", "Marina":"3449007"}
```

```
[2] rubrica
```

```
{'Antonio': '323573', 'Giuseppe': '322955', 'Marina': '3449007'}
```

```
if "Laura" in rubrica:
  print("Laura c'è")
else:
  print("Laura non c'è")
```

```
Laura non c'è
```

Retrieving a Value from a Dictionary

```
[11] print(rubrica["Marina"])

     3449007
```

```
[5]  print(rubrica[2])

     ---------------------------------------------------------------------------
     KeyError                                  Traceback (most recent call last)
     <ipython-input-5-1443c816d35a> in <module>
     ----> 1 print(rubrica[2])

     KeyError: 2

     SEARCH STACK OVERFLOW
```

```
     print(rubrica["Mario"])

     ---------------------------------------------------------------------------
     KeyError                                  Traceback (most recent call last)
     <ipython-input-12-6526382855a0> in <module>
     ----> 1 print(rubrica["Mario"])

     KeyError: 'Mario'

     SEARCH STACK OVERFLOW
```

Retrieving a Value from a Dictionary

```
[12] print(rubrica["Mario"])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-12-6526382855a0> in <module>
----> 1 print(rubrica["Mario"])

KeyError: 'Mario'
```

SEARCH STACK OVERFLOW

```
if "Mario" not in rubrica:
  print("Mario non si trova.")
  print("Lo vuoi aggiungere?")
```

```
Mario non si trova.
Lo vuoi aggiungere?
```

Adding Elements to an Existing Dictionary

- Dictionaries are mutable objects
- To add a new key-value pair:

```
dictionary[key] = value
```

  - If key exists in the dictionary, the value associated with it will be changed

```
[15] rubrica["Mario"] = "392356"
     rubrica["Laura"] = "339247"
```

```
[16] rubrica
```

```
{'Antonio': '323573',
 'Giuseppe': '322955',
 'Marina': '3449007',
 'Mario': '392356',
 'Laura': '339247'}
```

```
[17] rubrica["Antonio"] = "322111"
```

```
rubrica
```

```
{'Antonio': '322111',
 'Giuseppe': '322955',
 'Marina': '3449007',
 'Mario': '392356',
 'Laura': '339247'}
```

Deleting Elements From an Existing Dictionary

- To delete a key-value pair:

```
del dictionary[key]
```

  - If key is not in the dictionary, `KeyError` exception is raised

Deleting Elements From an Existing Dictionary

```
[19] del rubrica["Laura"]
```

```
rubrica
```

```
{'Antonio': '322111',
 'Giuseppe': '322955',
 'Marina': '3449007',
 'Mario': '392356'}
```

Getting the Number of Elements and Mixing Data Types

- `len` function: used to obtain number of elements in a dictionary
- Keys must be immutable objects, but associated values can be any type of object
  - One dictionary can include keys of several different immutable types
- Values stored in a single dictionary can be of different types

Getting the Number of Elements and Mixing Data Types

```
[20] rubrica
```

```
{'Antonio': '322111',
 'Giuseppe': '322955',
 'Marina': '3449007',
 'Mario': '392356'}
```

```
len(rubrica)
```

```
4
```

Getting the Number of Elements and Mixing Data Types

```
[23] rubrica["Ethan"] = ["323500","336599"]
```

```
rubrica
```

```
{'Antonio': '322111',
 'Giuseppe': '322955',
 'Marina': '3449007',
 'Mario': '392356',
 'Ethan': ['323500', '336599']}
```

Getting the Number of Elements and Mixing Data Types

```
[26] mix = {'abc':1, 999:'ciao', (3, "a", 6):["a",3,"c"]}
```

```
mix
```

```
{'abc': 1, 999: 'ciao', (3, 'a', 6): ['a', 3, 'c']}
```

Creating an Empty Dictionary and Using `for` Loop to Iterate Over a Dictionary

- To create an empty dictionary:
  - Use `{}`
  - Use built-in function `dict()`
  - Elements can be added to the dictionary as program executes
- Use a `for` loop to iterate over a dictionary
  - General format: `for` *`key`* `in` *`dictionary`*`:`

Creating an Empty Dictionary and Using `for` Loop to Iterate Over a Dictionary

```
[28] rubrica = {}
```

```
[29] rubrica
```

```
{}
```

```
[31] rubrica["Lorenzo"] = "345098"
     rubrica["Miriana"] = "333678"
```

```
rubrica
```

```
{'Lorenzo': '345098', 'Miriana': '333678'}
```

Creating an Empty Dictionary and Using `for` Loop to Iterate Over a Dictionary

```
[32] rubrica

     {'Lorenzo': '345098', 'Miriana': '333678'}

[33] for chiave in rubrica:
       print(chiave)

     Lorenzo
     Miriana

     for chiave in rubrica:
       print(chiave + ":" + rubrica[chiave])

     Lorenzo:345098
     Miriana:333678
```

Some Dictionary Methods (1 of 5)

- `clear` method: deletes all the elements in a dictionary, leaving it empty
  - Format: *`dictionary`*`.clear()`
- `get` method: gets a value associated with specified key from the dictionary
  - Format: *`dictionary`*`.get(`*`key`*`, `*`default`*`)`
    - *`default`* is returned if *`key`* is not found
  - Alternative to `[]` operator
    - Cannot raise `KeyError` exception

Some Dictionary Methods (1 of 5)

```
[35] rubrica.get("Miriana","Non la trovo")

     '333678'

[36] rubrica.clear()

     rubrica.get("Miriana","Non la trovo")

     'Non la trovo'
```

Some Dictionary Methods (2 of 5)

- `items` method: returns all the dictionaries keys and associated values
  - Format: *`dictionary`*`.items()`
  - Returned as a *dictionary view*
    - Each element in dictionary view is a tuple which contains a key and its associated value
    - Use a `for` loop to iterate over the tuples in the sequence
      - Can use a variable which receives a tuple, or can use two variables which receive key and value

```
[38] rubrica["GianPio"] = "325298"
     rubrica["Nicole"] = "332628"
```

```
[39] rubrica.items()
```

```
dict_items([('GianPio', '325298'), ('Nicole', '332628')])
```

```
[42] for chiave, valore in rubrica.items():
       print(chiave+":"+valore)
```

```
GianPio:325298
Nicole:332628
```

```
for chiave, valore in rubrica:
  print(chiave+":"+valore)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-43-813750bf24f3> in <module>
----> 1 for chiave, valore in rubrica:
      2   print(chiave+":"+valore)

ValueError: too many values to unpack (expected 2)
```

SEARCH STACK OVERFLOW

Some Dictionary Methods (3 of 5)

- `keys` method: returns all the dictionaries keys as a sequence
  - Format: *`dictionary`*`.keys()`
- `pop` method: returns value associated with specified key and removes that key-value pair from the dictionary
  - Format: *`dictionary`*`.pop(`*`key`*`, `*`default`*`)`
    - *`default`* is returned if *`key`* is not found

Some Dictionary Methods (4 of 5)

- `popitem` method: Returns, as a tuple, the key-value pair that was last added to the dictionary. The method also removes the key-value pair from the dictionary.
  - Format: *`dictionary`*`.popitem()`
  - Key-value pair returned as a tuple
- `values` method: returns all the dictionaries values as a sequence
  - Format: *`dictionary`*`.values()`
  - Use a `for` loop to iterate over the values

Some Dictionary Methods (5 of 5)

Table 9-1 Some of the dictionary methods

Method	Description
`Clear`	Clears the contents of a dictionary.
`get`	Gets the value associated with a specified key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.
`items`	Returns all the keys in a dictionary and their associated values as a sequence of tuples.
`keys`	Returns all the keys in a dictionary as a sequence of tuples.
`pop`	Returns the value associated with a specified key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.
`popitem`	Returns, as a tuple, the key-value pair that was last added to the dictionary. The method also removes the key-value pair from the dictionary.
`values`	Returns all the values in the dictionary as a sequence of tuples.

Sets

- Set: object that stores a collection of data in same way as mathematical set
  - All items must be unique
  - Set is unordered
  - Elements can be of different data types

Creating a Set

- `set` function: used to create a set
  - For empty set, call `set()`
  - For non-empty set, call `set(`*`argument`*`)` where *`argument`* is an object that contains iterable elements
    - e.g., *`argument`* can be a list, string, or tuple
    - If *`argument`* is a string, each character becomes a set element
      - For set of strings, pass them to the function as a list
    - If *`argument`* contains duplicates, only one of the duplicates will appear in the set

```
[44] myset = set()
```

```
[45] myset
```

```
set()
```

```
[46] altro_set = set(['a','b','c'])
```

```
[47] altro_set
```

```
{'a', 'b', 'c'}
```

```
[48] altro_ancora = set('abbccc')
```

```
altro_ancora
```

```
{'a', 'b', 'c'}
```

Esercizio

Come posso creare un set contenente i 3 elementi "uno", "due e "tre"?

Esercizio

Come posso creare un set contenente i 3 elementi "uno", "due e "tre"?

```
risp = set("uno", "due", "tre")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-50-9191c8fc6c4a> in <module>
----> 1 risp = set("uno", "due", "tre")

TypeError: set expected at most 1 argument, got 3
```

SEARCH STACK OVERFLOW

Esercizio

```
[1]  risp = set("uno" "due" "tre")
```

```
[2]  risp
```

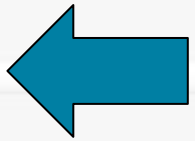
```
{'d', 'e', 'n', 'o', 'r', 't', 'u'}
```

```
[3]  risp = set("uno due tre")
```

```
[4]  risp
```

```
{' ', 'd', 'e', 'n', 'o', 'r', 't', 'u'}
```

```
[5]  risp = set(["uno","due","tre"])
```



```
risp
```

```
{'due', 'tre', 'uno'}
```

Getting the Number of and Adding Elements

- `len` function: returns the number of elements in the set
- Sets are mutable objects
- `add` method: adds an element to a set
- `update` method: adds a group of elements to a set
  - Argument must be a sequence containing iterable elements, and each of the elements is added to the set

Getting the Number of and Adding Elements

```
[7]  myset = set()
     myset.add(1)
     myset.add(2)
     myset.add(3)

     len(myset)
```

```
3
```

```
myset.update([5,6,7])

print(myset)

len(myset)
```

```
{1, 2, 3, 5, 6, 7}
6
```

Deleting Elements From a Set

- `remove` and `discard` methods: remove the specified item from the set
  - The item that should be removed is passed to both methods as an argument
  - Behave differently when the specified item is not found in the set
    - `remove` method raises a `KeyError` exception
    - `discard` method does not raise an exception
- `clear` method: clears all the elements of the set

Using the `for` Loop, `in`, and `not in` Operators With a Set

- A `for` loop can be used to iterate over elements in a set
  - General format: `for` *`item`* `in` *`set`*`:`
  - The loop iterates once for each element in the set
- The `in` operator can be used to test whether a value exists in a set
  - Similarly, the `not in` operator can be used to test whether a value does not exist in a set

Finding the Union of Sets

- Union of two sets: a set that contains all the elements of both sets
- To find the union of two sets:
  - Use the `union` method
    - Format: *`set1`*`.union(`*`set2`*`)`
  - Use the `|` operator
    - Format: *`set1 | set2`*
  - Both techniques return a new set which contains the union of both sets

Finding the Intersection of Sets

- Intersection of two sets: a set that contains only the elements found in both sets
- To find the intersection of two sets:
  - Use the `intersection` method
    - Format: *`set1`*`.intersection(`*`set2`*`)`
  - Use the `&` operator
    - Format: *`set1 & set2`*
  - Both techniques return a new set which contains the intersection of both sets

Finding the Difference of Sets

- Difference of two sets: a set that contains the elements that appear in the first set but do not appear in the second set
- To find the difference of two sets:
  - Use the `difference` method
    - Format: *`set1`*`.difference(`*`set2`*`)`
  - Use the – operator
    - Format: *`set1 - set2`*

Finding the Symmetric Difference of Sets

- Symmetric difference of two sets: a set that contains the elements that are not shared by the two sets
- To find the symmetric difference of two sets:
  - Use the `symmetric_difference` method
    - Format: *`set1`*`.symmetric_difference(`*`set2`*`)`
  - Use the `^` operator
    - Format: *`set1 ^ set2`*

Finding Subsets and Supersets (1 of 2)

- Set A is subset of set B if all the elements in set A are included in set B
- To determine whether set A is subset of set B
  - Use the `issubset` method
    - Format: *`setA`*`.issubset(`*`setB`*`)`
  - Use the <= operator
    - Format: *`setA <= setB`*

Finding Subsets and Supersets (2 of 2)

- Set A is superset of set B if it contains all the elements of set B
- To determine whether set A is superset of set B
  - Use the `issuperset` method
    - Format: *`setA`*`.issuperset(`*`setB`*`)`
  - Use the `>=` operator
    - Format: *`setA >= setB`*

Summary (1 of 2)

- This chapter covered:
  - Dictionaries, including:
    - Creating dictionaries
    - Inserting, retrieving, adding, and deleting key-value pairs
    - `for` loops and `in` and `not in` operators
    - Dictionary methods

Summary (2 of 2)

- This chapter covered (cont'd):
  - Sets:
    - Creating sets
    - Adding elements to and removing elements from sets
    - Finding set union, intersection, difference and symmetric difference
    - Finding subsets and supersets

Corso di STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI

UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Modulo di Sistemi di Elaborazione delle Informazioni

Dizionari e Set

Docente:
Domenico Daniele Bloisi