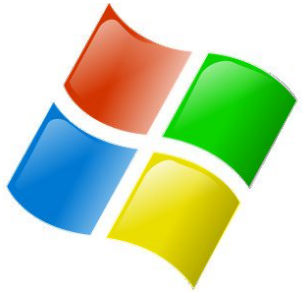
Corso di STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI

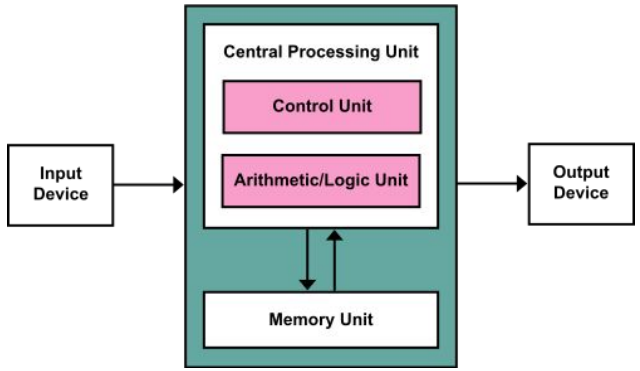
UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Modulo di Sistemi di Elaborazione delle Informazioni

Stringhe

Docente:
Domenico Daniele Bloisi

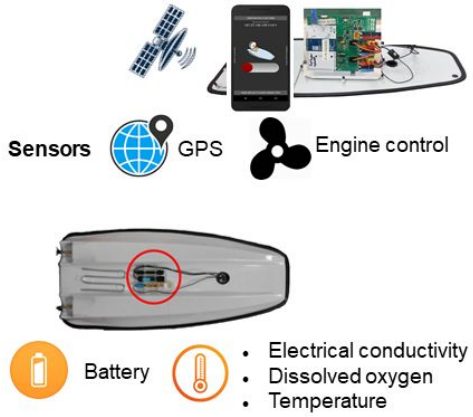


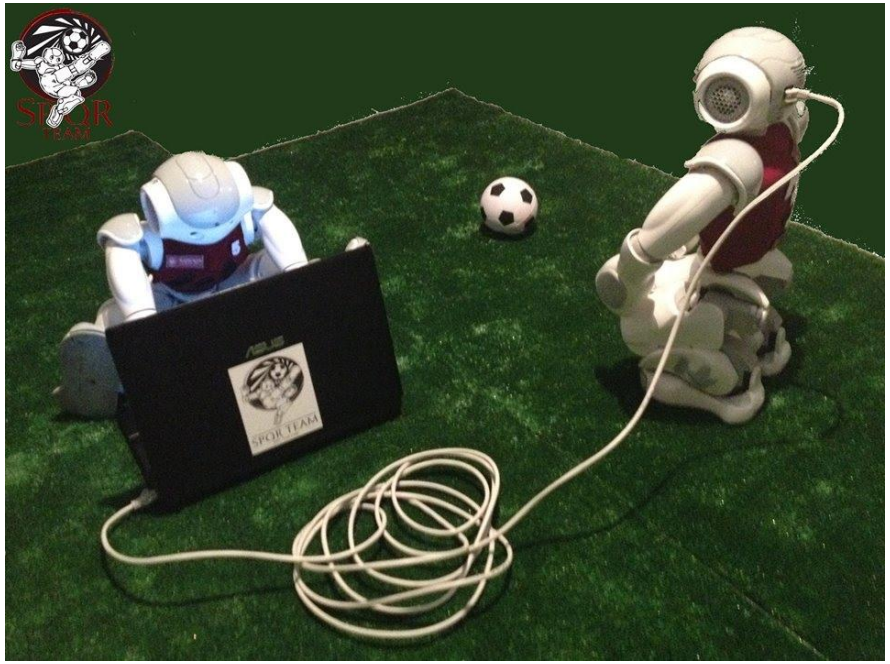


Domenico Daniele Bloisi

- 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



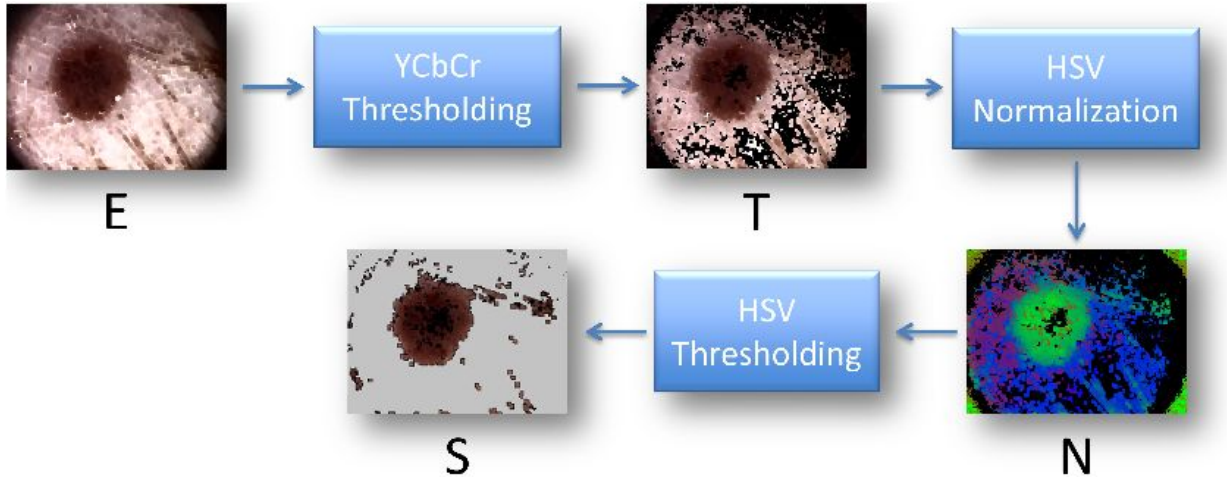


Interessi di ricerca

- 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

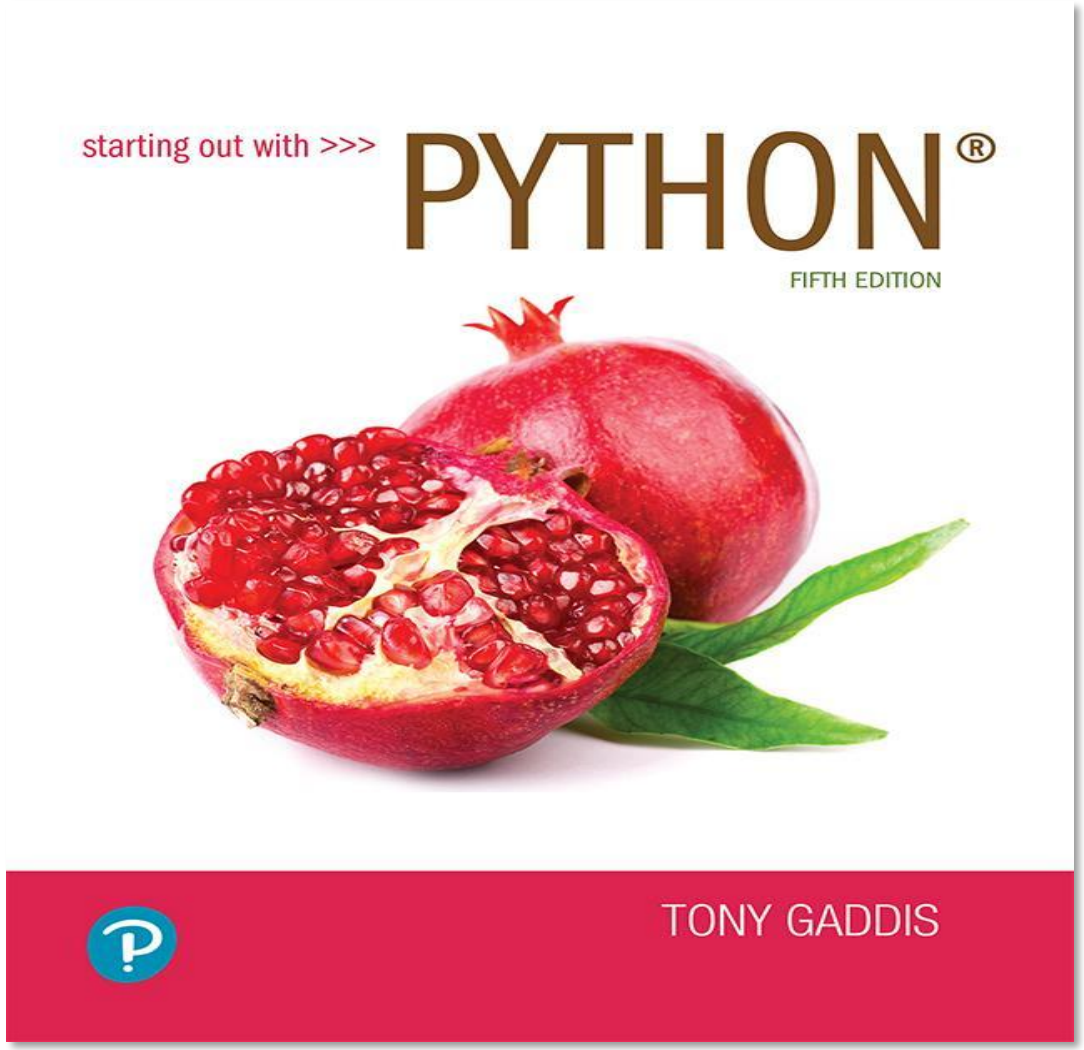
Ricevimento Bloisi

- 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

Starting out with Python

Fifth Edition



Chapter 8

More About Strings

Topics

- Basic String Operations
- String Slicing
- Testing, Searching, and Manipulating Strings

Basic String Operations

- Many types of programs perform operations on strings
- In Python, many tools for examining and manipulating strings
  - Strings are sequences, so many of the tools that work with sequences work with strings

Accessing the Individual Characters in a String (1 of 4)

- To access an individual character in a string:
  - Use a `for` loop
    - Format: `for character in string:`
    - Useful when need to iterate over the whole string, such as to count the occurrences of a specific character
  - Use indexing
    - Each character has an index specifying its position in the string, starting at 0
    - Format: `character = my_string[i]`

```
name = 'Juliet'

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

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

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

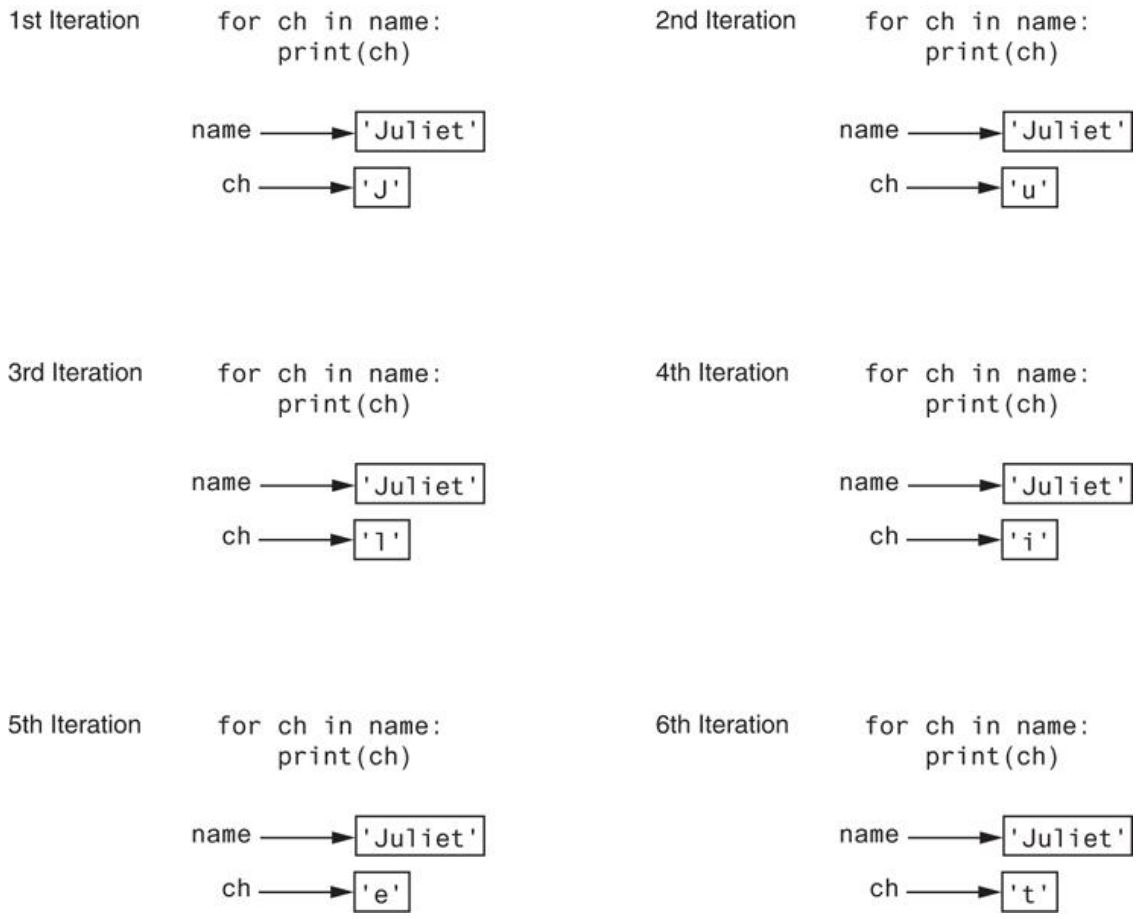


Figure 8-1 Iterating over the string 'Juliet'

```python
# Questo programma conta il numero di volte
# che la lettera T (maiuscola o minuscola)
# compare in una stringa.

def main():
    # Crea una variabile da utilizzare per tenere il conto.
    # La variabile viene inizializzata a 0.
    count = 0

    # Ottiene una stringa dall'utente.
    my_string = input('Inserisci una frase: ')

    # Conta le T.
    for ch in my_string:
        if ch == 'T' or ch == 't':
            count += 1

    # Stampa il risultato.
    print(f'La lettera T compare {count} volte.')

# Chiama la funzione main.
if __name__ == '__main__':
    main()
```

```
Inserisci una frase: Trentatré trentini
La lettera T compare 5 volte.
```

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)

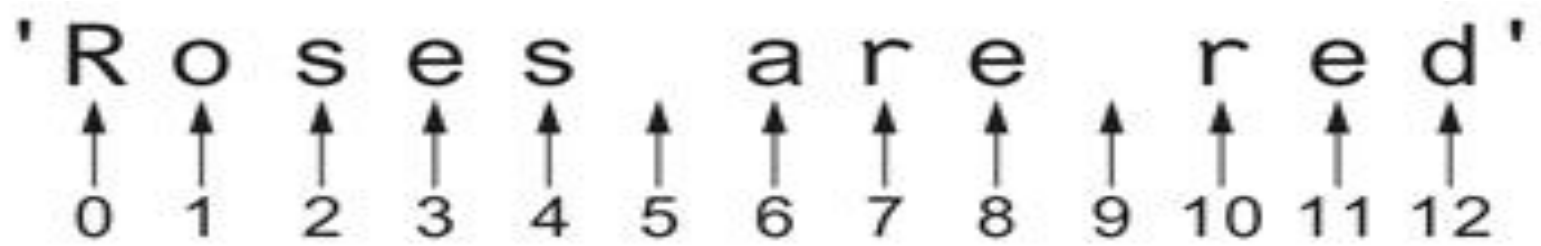


Figure 8-2 String indexes

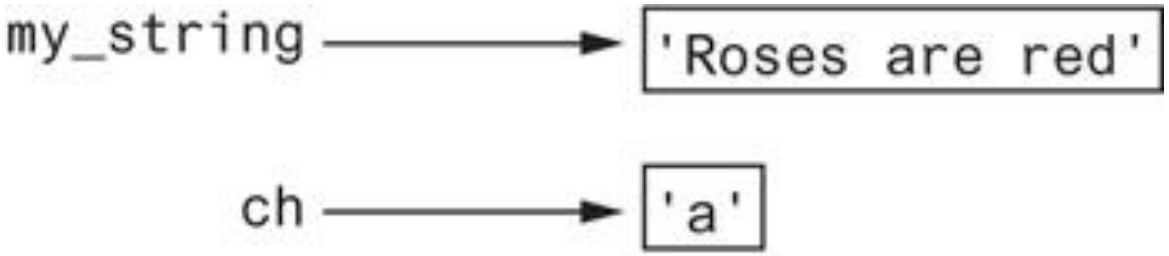


Figure 8-3 Getting a copy of a character from a string

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

- `IndexError` exception will occur if:
  - You try to use an index that is out of range for the string
    - Likely to happen when loop iterates beyond the end of the string
- `len(`*`string`*`)` function can be used to obtain the length of a string
  - Useful to prevent loops from iterating beyond the end of a string

```
city = 'Boston'
index = 0
while index < 7:
  print(city[index])
  index += 1
```

```
B
o
s
t
o
n
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-4-2b091005aafb> in <module>
      2 index = 0
      3 while index < 7:
----> 4   print(city[index])
      5   index += 1

IndexError: string index out of range
```

SEARCH STACK OVERFLOW

```
city = 'Boston'
index = 0
while index < len(city):
  print(city[index])
  index += 1
```

```
B
o
s
t
o
n
```

String Concatenation

- Concatenation: appending one string to the end of another string
  - Use the + operator to produce a string that is a combination of its operands
  - The augmented assignment operator += can also be used to concatenate strings
    - The operand on the left side of the += operator must be an existing variable; otherwise, an exception is raised

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

    print(message)
```

```
HelloWorld
```

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

print(message)
```

```
Hello World
```

Strings Are Immutable (1 of 2)

- Strings are immutable
  - Once they are created, they cannot be changed
    - Concatenation doesn't actually change the existing string, but rather creates a new string and assigns the new string to the previously used variable
  - Cannot use an expression of the form
  - `string[index] = new_character`
    - Statement of this type will raise an exception

```
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

```
# Questo programma concatena delle stringhe.

def main():
    name = 'Carmen'
    print(f'Il nome è: {name}')
    name = name + ' Brown'
    print(f'Ora il nome è {name}')

# Chiama la funzione main.
if __name__ == '__main__':
    main()
```

```
Il nome è: Carmen
Ora il nome è Carmen Brown
```

Strings Are Immutable (2 of 2)

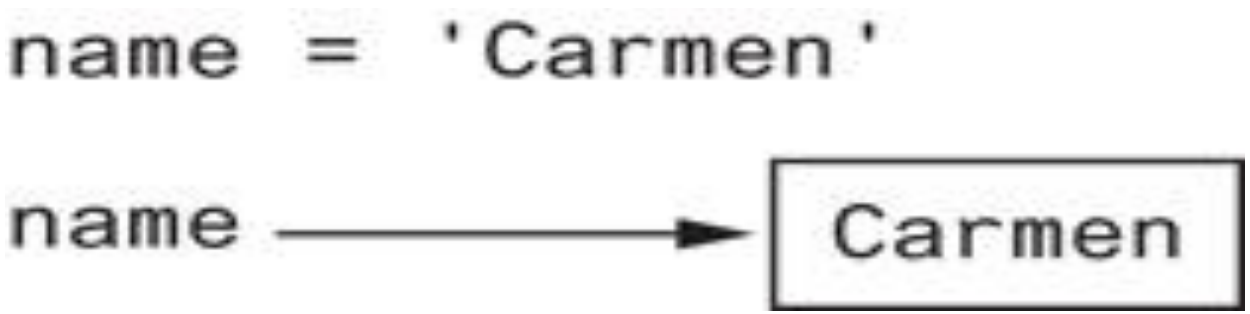


Figure 8-4 The string 'Carmen' assigned to name

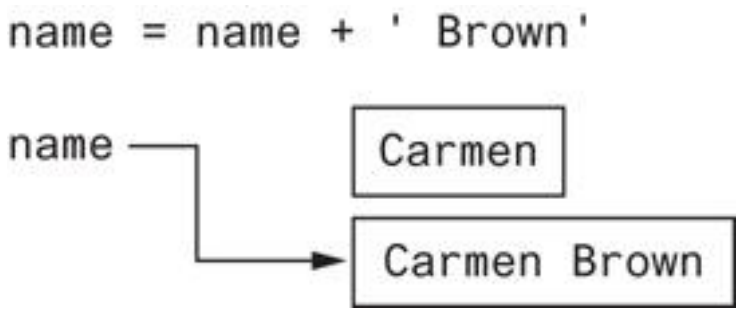


Figure 8-5 The string 'Carmen Brown' assigned to name

String Slicing

- Slice: span of items taken from a sequence, known as *substring*
  - Slicing format: `string[start : end]`
    - Expression will return a string containing a copy of the characters from `start` up to, but not including, `end`
    - If `start` not specified, `0` is used for start index
    - If `end` not specified, `len(string)` is used for end index
  - Slicing expressions can include a step value and negative indexes relative to end of string

```
[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
```

Testing, Searching, and Manipulating Strings

- You can use the `in` operator to determine whether one string is contained in another string
  - General format: *`string1`* `in` *`string2`*
    - *`string1`* and *`string2`* can be string literals or variables referencing strings
- Similarly you can use the `not in` operator to determine whether one string is not contained in another string

```
[14] text = "Nel mezzo del cammin di nostra vita..."

     if "nos" in text:
       print("stringa trovata!")
     else:
       print("stringa NON trovata!")
```

```
stringa trovata!
```

```
text = "Nel mezzo del cammin di nostra vita..."

if "nos" not in text:
  print("stringa NON trovata!")
else:
  print("stringa trovata!")
```

```
stringa trovata!
```

String Methods (1 of 7)

- Strings in Python have many types of methods, divided into different types of operations
  - General format:

mystring`.`*method*`(`*arguments*`)`

- Some methods test a string for specific characteristics
  - Generally Boolean methods, that return `True` if a condition exists, and `False` otherwise

```
[18] string1 = '1200'

     if string1.isdigit():
       print(f"la stringa {string1} contiene solo numeri!")
     else:
       print(f"la stringa {string1} contiene anche caratteri non numerici!")
```

```
la stringa 1200 contiene solo numeri!
```

```
string2 = '12a34c'

if string2.isdigit():
  print(f"la stringa {string2} contiene solo numeri!")
else:
  print(f"la stringa {string2} contiene anche caratteri non numerici!")
```

```
la stringa 12a34c contiene anche caratteri non numerici!
```

String Methods (2 of 7)

Table 8-1 Some string testing methods

Method	Description
`isalnum()`	Returns true if the string contains only alphabetic letters or digits and is at least one character in length. Returns false otherwise.
`isalpha()`	Returns true if the string contains only alphabetic letters and is at least one character in length. Returns false otherwise.
`isdigit()`	Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.
`islower()`	Returns true if all of the alphabetic letters in the string are lowercase, and the string contains at least one alphabetic letter. Returns false otherwise.
`isspace()`	Returns true if the string contains only whitespace characters and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (`\n`), and tabs (`\t`).
`isupper()`	Returns true if all of the alphabetic letters in the string are uppercase, and the string contains at least one alphabetic letter. Returns false otherwise.

Posta in arrivo | 14-stringhe - | string.ipynb - | 14-stringhe.pp | 08_Gaddis Pyt | Posta in arrivo | Posta in arrivo | (39) WhatsAp | nel cammin di

colab.research.google.com/drive/1Nt61nKigU0xndoiELZGqV9nPtgnnKrRg#scrollTo=G6kGQXnJ6y61

string.ipynb

File Modifica Visualizza Inserisci Runtime Strumenti Guida Tutte le modifiche sono state salvate

Commenta Condividi

+ Codice + Testo

RAM Disco

Modifica

```
# Questo programma dimostra diversi metodi per il test delle stringhe.
def main():
    # Riceve una stringa dall'utente.
    user_string = input('Inserisci una stringa: ')

    print('Ecco che cosa ho scoperto sulla stringa:')

    # Testa la stringa.
    if user_string.isalnum():
       print('La stringa è alfanumerica.')
    if user_string.isdigit():
       print('La stringa contiene solo numeri.')
    if user_string.isalpha():
       print('La stringa contiene solo caratteri alfabetici.')
    if user_string.isspace():
       print('La stringa contiene solo spazi bianchi.')
    if user_string.islower():
       print('Le lettere di questa stringa sono tutte minuscole.')
    if user_string.isupper():
       print('Le lettere di questa stringa sono tutte maiuscole.')

# Chiama la funzione main.
if __name__ == '__main__':
    main()
```

```
Inserisci una stringa: tamburo
Ecco che cosa ho scoperto sulla stringa:
La stringa è alfanumerica.
La stringa contiene solo caratteri alfabetici.
Le lettere di questa stringa sono tutte minuscole.
```

Battery saver

Battery saver is on
Consider plugging in your device.

13 s data/ora di completamento: 23:06

String Methods (3 of 7)

- Some methods return a copy of the string, to which modifications have been made
  - Simulate strings as mutable objects
- String comparisons are case-sensitive
  - Uppercase characters are distinguished from lowercase characters
  - `lower` and `upper` methods can be used for making case-insensitive string comparisons

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 (5 of 7)

- Programs commonly need to search for substrings
- Several methods to accomplish this:
  - `endswith(`*`substring`*`)`: checks if the string ends with *`substring`*
    - Returns `True` or `False`
  - `startswith(`*`substring`*`)`: checks if the string starts with *`substring`*
    - Returns `True` or `False`

```
filename = input("inserire il nome del file: ")
if filename.endswith(".txt"):
  print("si tratta di un file di testo")
elif filename.endswith(".jpg"):
  print("si tratta di un file immagine")
elif filename.endswith(".py"):
  print("si tratta di un file sorgente di Python")
else:
  print("tipo di file sconosciuto")
```

```
inserire il nome del file: img.jpg
si tratta di un file immagine
```

String Methods (6 of 7)

- Several methods to accomplish this (cont'd):
  - `find(substring)`: searches for *substring* within the string
    - Returns lowest index of the substring, or if the substring is not contained in the string, returns -1
  - `replace(substring, new_string)`:
    - Returns a copy of the string where every occurrence of *substring* is replaced with *new_string*

```python
string = "Nel mezzo del cammin di nostra vita..."

position = string.find("nel")

if position != -1:
  print(f"la parola 'nel' è stata trovata nella posizone {position}")
else:
  print(f"la parola 'nel' non è stata trovata")
```

```
la parola 'nel' non è stata trovata
```

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*.

The Repetition Operator

- Repetition operator: makes multiple copies of a string and joins them together
  - The * symbol is a repetition operator when applied to a string and an integer
    - String is left operand; number is right
  - General format: *string_to_copy * n*
  - Variable references a new string which contains multiple copies of the original string

```python
# Questo programma dimostra l'uso dell'operatore di ripetizione.
def main():
    # Stampa nove righe di larghezza man mano più ampia.
    for count in range(1, 10):
        print('Z' * count)

    # Stampa nove righe di larghezza man mano più stretta.
    for count in range(8, 0, -1):
        print('Z' * count)

# Chiama la funzione main.
if __name__ == '__main__':
    main()
```

```
Z
ZZ
ZZZ
ZZZZ
ZZZZZ
ZZZZZZ
ZZZZZZZ
ZZZZZZZZ
ZZZZZZZZZ
ZZZZZZZZ
ZZZZZZZ
ZZZZZZ
ZZZZZ
ZZZZ
ZZZ
ZZ
Z
```

Splitting a String (1 of 2)

- `split` method: returns a list containing the words in the string
  - By default, uses space as separator
  - Can specify a different separator by passing it as an argument to the `split` method

Splitting a String (2 of 2)

- Examples:

```
>>> my_string = 'One two three four'
>>> word_list = my_string.split()
>>> word_list
['One', 'two', 'three', 'four']
>>>
```

```
>>> my_string = '1/2/3/4/5'
>>> number_list = my_string.split('/')
>>> number_list
['1', '2', '3', '4', '5']
>>>
```

String Tokens (1 of 4)

- Sometimes a string contains substrings that are separated by a special character
  - Example:

```
'peach raspberry strawberry vanilla'
```

  - This string contains the substrings *peach*, *raspberry*, *strawberry*, and *vanilla*
  - The substrings are separated by the space character
  - The substrings are known as *tokens* and the separating character is known as the *delimiter*

String Tokens (2 of 4)

- Example:

```
'17;92;81;12;46;5'
```

  - This string contains the tokens 17, 92, 81, 12, 46, and 5
  - The delimiter is the ; character

String Tokens (3 of 4)

- *Tokenizing* is the process of breaking a string into tokens
- When you tokenize a string, you extract the tokens and store them as individual items
- In Python you can use the split method to tokenize a string

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']
>>>
```

Summary

- This chapter covered:
  - String operations, including:
    - Methods for iterating over strings
    - Repetition and concatenation operators
    - Strings as immutable objects
    - Slicing strings and testing strings
    - String methods
    - Splitting a string

Corso di STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI

UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Modulo di Sistemi di Elaborazione delle Informazioni

Stringhe

Docente:
Domenico Daniele Bloisi