Python: Concetti di base - Parte 2

Filippo Poltronieri

filippo.poltronieri@unife.it

Crediti a: Mattia Fogli

mattia.fogli@unife.it

Indice

- Bibliografia
- Licenze

1. Iterazione

1.1. Loop

Istruzioni che eseguono una o più istruzioni ripetutamente.

```
for i in range(3):
    print(i)

    range crea una sequenza di valori, cioè [0, 1, 2]

    L'istruzione for (con i come variabile di loop):

        Assegna il valore successivo da range a i

        Esegue il corpo

        Torna all'header

1.2. Iterare su una stringa
python

def has_e(word):
    """
    Verifica se una parola contiene la lettera 'e'

    word : stringa da verificare
    """
    for letter in word:
        if letter == 'E' or letter == 'e':
            return True
    return False

1.3. Iterare su un file
python

for line in open("words.txt"):
    print(line)

I loop for su words.txt riga per riga.

    open restituisce un oggetto file

    words.txt è questo file

python
```

```
>>> f = open("words.txt")
>>> f.readline()
'aah'
>>> f.readline()
'aah\n'
```

readline è un metodo di f:

    Legge caratteri dal file fino a quando incontra un newline (\n)

    Restituisce il risultato come str

1.4. Iterare e contare
python

```
total = 0
for line in open("words.txt"):
    total += 1
```

total = 0 è un'inizializzazione di variabile:

    Crea una nuova variabile (total) e le assegna un valore (0)

total += 1 è un aggiornamento di variabile:

    Assegna un nuovo valore a una variabile che già esiste

    += è un operatore di assegnamento aumentato

Inizializzazione → aggiornamento
python

```
for line in open("words.txt"):
    total += 1    # ERRORE
```

total += 1 significa total = total + 1:

    Prende il valore corrente di total

    Aggiunge 1

    Assegna il risultato di nuovo a total

Python valuta prima total + 1, ma total non esiste.
1.5. Ricerca lineare

Un pattern computazionale che cerca attraverso una sequenza di elementi e si ferma quando trova ciò che sta cercando.
python

```
def uses_any(word, letters):
    for letter in word.lower():
        if letter in letters.lower():
            return True
```

```
    return False

    lower è un metodo che può essere chiamato su str

    in è usato sia per l'iterazione che per la verifica di appartenenza
```

1.6. Doc test

Stringhe all'inizio di una funzione che testano le funzioni.

```
python

def uses_any(word, letters):
    """
    Verifica se una parola usa una qualsiasi di una lista di lettere

    >>> uses_any('banana', 'aeiou')
    True
    >>> uses_any('apple', 'xyz')
    False
    """
    for letter in word.lower():
        if letter in letters.lower():
            return True
    return False
```

Ogni test consiste di due righe:

```
    >>> seguito da un'espressione

    Il valore che l'espressione dovrebbe avere se la funzione funziona correttamente
```

Per esempio:

```
    >>> uses_any('apple', 'xyz')

    False
```

poiché apple non usa nessuna di xyz.

uses_any.py (vedi qui):

```
python

def uses_any(word, letters):
    """
    Verifica se una parola usa una qualsiasi di una lista di lettere

    >>> uses_any('banana', 'aeiou')
    True
    >>> uses_any('apple', 'xyz')
    False
    """
    for letter in word.lower():
        if letter in letters.lower():
```

```
            return True
    return False

if __name__ == "__main__":
    import doctest
    doctest.testmod()

L'interprete cambia __name__ come segue:

    Se il programma viene eseguito come script, __name__ è uguale a
"__main__"

    Se il programma viene importato, __name__ è uguale al nome del modulo

python

if __name__ == "__main__":
    import doctest
    doctest.testmod()

I test vengono eseguiti solo quando il programma viene eseguito come
script.

La convenzione è di posizionare questo if in fondo per chiarezza.
bash

$ python uses_any.py -v
Trying:
    uses_any('banana', 'aeiou')
Expecting:
    True
ok
Trying:
    uses_any('apple', 'xyz')
Expecting:
    False
ok
1 items had no tests:
    __main__
1 items passed all tests:
    2 tests in __main__.uses_any
2 tests in 2 items.
2 passed and 0 failed.
Test passed.

-v stampa:

    Un log di ciò che il modulo doctest sta provando

    Un riepilogo alla fine

2. Stringhe
2.1. Le stringhe sono sequenze
```

Una stringa è una sequenza di caratteri. In altre parole:

    Una stringa è una collezione ordinata di caratteri

    Ogni carattere è identificato da un indice intero

python

```
>>> fruit = 'apple'
>>> fruit[1]
'p'
>>> fruit[-1]
'e'
```

2.2. Slicing di stringhe

Parti di una stringa specificate da intervalli di indici.
python

```
>>> fruit = 'apple'
>>> fruit[0:3]
'app'
```

L'operatore [n:m]:

    Restituisce dall'ennesimo all'emmesimo carattere

    Includendo l'ennesimo, ma escludendo l'emmesimo

python

```
>>> fruit[1:3]
'pp'
>>> fruit[3:]
'le'
>>> fruit[3:3]
''
```

    Se [:m] → lo slice inizia dall'inizio

    Se [n:] → lo slice va fino alla fine

    Se n == m → stringa vuota

2.3. Le stringhe sono immutabili

Gli elementi delle stringhe non possono essere cambiati.
python

```
>>> fruit = 'apple'
>>> fruit[0] = 'b'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

```
>>> new_fruit = 'b' + fruit[1:]
>>> new_fruit
'bpple'
>>> fruit
'apple'
```

2.4. Metodi delle stringhe

Diversi metodi possono essere chiamati sugli oggetti stringa.
python

```
>>> fruit = 'apple'
>>> fruit.upper()
'APPLE'
>>> fruit
'apple'
```

L'operatore punto specifica:

Il nome del metodo (upper)

Il nome dell'oggetto str a cui applicare il metodo (fruit)

3. Liste
3.1. Le liste sono sequenze

Sequenze di valori, che possono essere di qualsiasi tipo.
python

```
>>> numbers = [42, 123]
>>> cheeses = ['Cheddar', 'Edam', 'Gouda']
>>> t = ['spam', 2.0, 5, [10, 20]]
```

t è una lista annidata, cioè una lista che è un elemento di un'altra lista.
3.2. Le liste sono mutabili

A differenza degli oggetti str, che sono immutabili.

Gli indici delle liste funzionano allo stesso modo degli indici delle stringhe.
python

```
>>> numbers = [42, 123]
>>> numbers[1] = 17
>>> numbers
[42, 17]
```

3.3. Slicing di liste

L'operatore slice funziona sulle liste come funziona sulle stringhe.
python

```
>>> letters = ['a', 'b', 'c', 'd']
>>> letters[1:3]
```

```
['b', 'c']
>>> letters[:]
['a', 'b', 'c', 'd']

    Se [:], lo slice è una copia dell'intera lista

3.4. Operazioni sulle liste

Ci sono solo due operatori che funzionano con le liste:

    + unisce due liste (concatenazione)

    * crea copie multiple e le concatena

Funzione built-in   Descrizione
sum Somma gli elementi
min Trova l'elemento più piccolo
max Trova l'elemento più grande
sorted  Ordina gli elementi di una lista
python

>>> numbers = [1, 2, 3, 4, 5]
>>> sum(numbers)
15
>>> min(numbers)
1
>>> max(numbers)
5
>>> scramble = ['c', 'a', 'b']
>>> sorted(scramble)
['a', 'b', 'c']

3.5. Metodi delle liste
Metodo  Descrizione
append(x)   Aggiunge x alla fine della lista
extend(l)   Aggiunge tutti gli elementi di l alla fine della lista
pop(i)  Rimuove l'elemento i-esimo e lo restituisce. Se non viene
specificato un indice, rimuove e restituisce l'ultimo elemento
remove(x)   Rimuove il primo elemento dalla lista il cui valore è uguale a
x
python

>>> numbers = [1, 2, 3, 4, 5]
>>> numbers.append(6)
>>> numbers
[1, 2, 3, 4, 5, 6]
>>> numbers.extend([7, 8])
>>> numbers
[1, 2, 3, 4, 5, 6, 7, 8]
>>> numbers.pop()
8
>>> numbers.remove(5)
>>> numbers
[1, 2, 3, 4, 6, 7]
```

```
3.6. Liste e stringhe

Lista di caratteri vs stringa.
python

>>> s = 'apple'
>>> type(s)
<class 'str'>
>>> s.pop()
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
AttributeError: 'str' object has no attribute 'pop'
>>> t = list(s)
>>> t.pop()
'e'

La maggior parte dei metodi delle liste modifica la lista originale e
restituisce None.

I metodi delle stringhe restituiscono una nuova stringa (le stringhe sono
immutabili).
python

>>> l = [1, 2, 3]
>>> l = l.remove(3)
>>> l.remove(2)
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
AttributeError: 'NoneType' object has no attribute 'remove'

l è NoneType perché remove restituisce None.
3.7. Iterare attraverso una lista
python

>>> numbers = [1, 2, 3, 4, 5]
>>> for n in numbers:
...     print(n)
...
1
2
3
4
5

Un loop for su una lista vuota [] non esegue mai il corpo.
3.8. Oggetti e valori

is verifica se due variabili si riferiscono allo stesso oggetto.
python

>>> a = 'apple'
>>> b = 'apple'
>>> a is b
```

```
True

a e b si riferiscono allo stesso oggetto str:

    Hanno lo stesso valore (equivalenti)

    Sono lo stesso oggetto (identici)

python

>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> a is b
False

a e b si riferiscono a due diversi oggetti lista:

    Hanno lo stesso valore (equivalenti)

    Ma non sono identici

3.9. Riferimenti e alias

Un riferimento è l'associazione di una variabile con un oggetto.
python

>>> a = [1, 2, 3]
>>> b = a
>>> a is b
True

L'oggetto [1, 2, 3] ha due riferimenti, cioè a e b.

Un oggetto con più di un riferimento è aliasato.

Se l'oggetto aliasato è mutabile, i cambiamenti fatti con un riferimento
influenzano l'altro.
python

>>> a = [1, 2, 3]
>>> b = a
>>> b[0] = 5
>>> a
[5, 2, 3]

In generale, evita l'aliasing quando lavori con oggetti mutabili.

Quando passi una lista a una funzione, la funzione ottiene un riferimento.
python

>>> def pop_first(l):
...     return l.pop(0)
...
>>> numbers = [1, 2, 3]
```

```
>>> pop_first(numbers)
1
>>> numbers
[2, 3]

l e numbers sono alias per lo stesso oggetto.
4. Dizionari
4.1. Un dizionario è un mapping

Oggetti che contengono coppie chiave-valore, dette anche elementi.
python

>>> numbers = {}
    # oppure numbers = dict()
>>> numbers['zero'] = 0
>>> numbers['one'] = 1
>>> numbers['two'] = 2
# numbers = {'zero': 0, 'one': 1, 'two': 2}
>>> numbers
{'zero': 0, 'one': 1, 'two': 2}

Ogni chiave (zero, one, e two) mappa a un valore (0, 1, e 2).
4.2. La ricerca per chiave è molto veloce

I dizionari sono implementati usando tabelle hash:

    Sovraccarico significativo di memoria

    Accesso veloce indipendentemente dalla dimensione (finché entra in
memoria)

Chiavi  Funzione hash   Buckets
John Smith  00  521-8976
    01  521-1234
Lisa Smith  03  :
    :   :
Sandra Dee  13  521-9655
    14  521-9655
    15

Fonte: Wikipedia
python

found = 0
for n in needles:
    if n in haystack:
        found += 1

    haystack contiene float

    needles è una lista di 1000 float (50% presi da haystack)

haystack    Fattore tempo dict  Fattore tempo list  Fattore
1K  1   0.000202s   1.00    0.010556s   1.00
```

```
10K 10  0.000140s   0.69    0.086586s   8.20
100K    100 0.000228s   1.13    0.871560s   82.57
1M  1K  0.000290s   1.44    9.189616s   870.56
10M 10K 0.000337s   1.67    97.948056s  9278.9

Con i dizionari, i tipi mutabili:

    Possono essere usati come valori

    Non possono essere usati come chiavi

python

>>> d = {'a': 1, 'b': [1, 2]}
>>> l = [3, 4]
>>> d[l] = 5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'

4.3. Iterazione e dizionari
python

>>> d = {'a': 1, 'b': 2, 'c': 3}
>>> for k in d:
...     print(k, d[k])
...
a 1
b 2
c 3

>>> d = {'a': 1, 'b': 2, 'c': 3}
>>> for v in d.values():
...     print(v)
...
1
2
3

5. Tuple
5.1. Le tuple sono come le liste

Come una lista, una tupla è:

    Una sequenza di valori

    Indicizzata da interi

python

>>> t = ('a', 'b', 'c', 'd')  # () sono opzionali
>>> type(t)
<class 'tuple'>
>>> t[1]
```

```
'b'

>>> s = ('a')
>>> type(s)
<class 'str'>
>>> t = ('a',)
>>> type(t)
<class 'tuple'>
```

Nota che un singolo valore tra parentesi non è una tupla.

La maggior parte degli operatori delle liste funziona anche con le tuple.
python

```
>>> t1 = ('h', 'e', 'l')
>>> t2 = ('l', 'o')
>>> t1 + t2
('h', 'e', 'l', 'l', 'o')
>>> t2 * 2
('l', 'o', 'l', 'o')
>>> sorted(t1 + t2)
['e', 'h', 'l', 'l', 'o']
```

5.2. Ma le tuple sono immutabili

L'operatore parentesi quadre non funziona.
python

```
>>> t = ('h', 'e', 'l', 'l', 'o')
>>> t[0] = 'a'
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Non ci sono metodi come append o remove.
python

```
>>> t = ('h', 'e', 'l', 'l', 'o')
>>> t.remove('l')
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'remove'
```

Poiché le tuple sono immutabili, sono hashable.
python

```
>>> d = dict()
>>> d[1, 2] = 'a'
>>> d[(3, 4)] = 'b'
>>> d
{(1, 2): 'a', (3, 4): 'b'}
```

5.3. Assegnazione di tuple

I valori sono assegnati alle variabili da sinistra a destra.
python

```
>>> a, b = 1, 2
>>> a
1
>>> b
2
>>> a, b
(1, 2)
```

Se il lato sinistro è una tupla, il lato destro può essere qualsiasi tipo di sequenza.
python

```
>>> email = "mattia.fogli@unife.it"
>>> username, domain = email.split("@")
>>> username, domain
('mattia.fogli', 'unife.it')
```

Utile per scambiare i valori di due variabili.
python

```
>>> a, b = 1, 2
>>> a, b
(1, 2)
>>> a, b = b, a
>>> a, b
(2, 1)
```

Questo funziona perché il lato destro viene valutato prima delle assegnazioni.

Utile anche per iterare attraverso i dizionari.
python

```
>>> d = {'zero': 0, 'one': 1, 'two': 2}
>>> for k, v in d.items():
...     print(k, '-->', v)
...
zero --> 0
one --> 1
two --> 2
```

5.4. Tuple come valori di ritorno

Le funzioni possono restituire solo un singolo valore, ma se quel valore è una tupla...
python

```
>>> quotient, remainder = divmod(10, 3)
>>> quotient, remainder
(3, 1)
```

```
5.5. Impacchettamento di argomenti

Raccogliere più argomenti in una tupla.
python

>>> def mean(*args):
...     return sum(args) / len(args)
...
>>> mean(1, 2)
1.5
>>> mean(1, 2, 3, 4, 5)
3.0

I parametri che iniziano con l'operatore * impacchettano.
python

>>> t = (10, 3)
>>> divmod(t)
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
TypeError: divmod expected 2 arguments, got 1
>>> divmod(*t)
(3, 1)

Gli argomenti che iniziano con l'operatore * spacchettano.
Glossario
Termine Significato
Aliasato    Se c'è più di una variabile che si riferisce a un oggetto,
l'oggetto è aliasato
Attributo   Uno dei valori nominati (variabili o metodi) associati a un
oggetto
Operatore di assegnamento aumentato Un operatore che aggiorna una variabile
in modo più conciso
Dizionario  Un oggetto che contiene coppie chiave-valore, anche chiamate
elementi
Doctest Una stringa all'inizio di una funzione che testa una funzione
Elemento    Uno dei valori in una lista o altra sequenza
Stringa vuota   Una stringa che non contiene caratteri e ha lunghezza 0
Equivalente Avere lo stesso valore
Oggetto file    Un oggetto che rappresenta un file aperto e tiene traccia
di quali parti del file sono state lette o scritte
Funzione hash   Una funzione che può essere usata per mappare dati di
dimensione arbitraria a valori di dimensione fissa
Tabella hash    Una collezione di coppie chiave-valore che usa una funzione
hash per calcolare un indice in un array di bucket, da cui può essere
trovato il valore desiderato. Durante la ricerca, la chiave viene hashata e
l'hash risultante indica dove è memorizzato il valore corrispondente
Identico    Essere lo stesso oggetto
Oggetto immutabile  Se gli elementi di un oggetto non possono essere
cambiati
Indice  Un valore intero usato per selezionare un elemento in una sequenza,
come un carattere in una stringa. In Python gli indici partono da 0
Invocazione Un'espressione, o parte di un'espressione, che chiama un metodo
Ricerca lineare Un pattern computazionale che cerca attraverso una sequenza
```

```
di elementi e si ferma quando trova ciò che sta cercando
Lista   Un oggetto mutabile che contiene una sequenza di valori
Loop    Un'istruzione che esegue una o più istruzioni, spesso ripetutamente
Variabile di loop   Una variabile definita nell'header di un loop for
Metodo  Una funzione che è associata a un oggetto e chiamata usando
l'operatore punto
Oggetto mutabile    Se gli elementi di un oggetto possono essere cambiati
Lista annidata  Una lista che è un elemento di un'altra lista
Impacchettamento    Raccogliere più argomenti in una tupla
Riferimento Un'associazione tra una variabile e il suo valore
Sequenza    Una collezione ordinata di valori dove ogni valore è
identificato da un indice intero
Slice   Una parte di una stringa specificata da un intervallo di indici
Tupla   Un oggetto immutabile che contiene una sequenza di valori
Spacchettamento Trattare una sequenza come più argomenti
Inizializzazione di variabile   Un'istruzione di assegnamento che crea una
nuova variabile e le assegna un valore
Aggiornamento di variabile  Un'istruzione di assegnamento che assegna un
nuovo valore a una variabile che già esiste, invece di creare una nuova
variabile
```