Python: Concetti di base - Parte 1

Filippo Poltronieri

filippo.poltronieri@unife.it

Crediti a: Mattia Fogli

mattia.fogli@unife.it

Indice

1. Introduzione

1.1. Python

Un linguaggio di programmazione

- Orientato agli oggetti
- Dinamicamente tipizzato

L'**implementazione standard** è CPython, che:

- È tipicamente classificato come interpretato
- Fornisce una modalità interattiva

1.2. Programmazione orientata agli oggetti

Un paradigma di programmazione basato sul concetto di oggetti.

Un oggetto consiste di:

- Dati sotto forma di campi, detti attributi
- Codice sotto forma di procedure, dette metodi

```
class MyClass:
    def __init__(self, a):
        self.a = a  # this is an attribute

    def f(self):  # this is a method
        return self.a

1.3. Tipizzazione statica vs dinamica
1.3.1. Linguaggi staticamente tipizzati

    I tipi sono associati alle variabili

    I tipi vengono verificati al momento della compilazione

java
```

```
String s = "abcd";

s sarà per sempre una String.
1.3.2. Linguaggi dinamicamente tipizzati

    I tipi sono associati ai valori

    I tipi vengono verificati a runtime

python

s = "abcd"
s = 1
s = ["a", 2, "c", 4]

s era prima una str, poi un int, e infine una list.
1.4. Compilazione vs interpretazione

Non esistono linguaggi compilati o interpretati.

    Qualsiasi linguaggio di programmazione può essere implementato in
entrambi i modi

    Compilatori e interpreti sono metodi di implementazione

Sì, questo significa che esistono:

    Interpreti per C

    Compilatori per Python

là fuori...
1.4.1. Compilazione

Un programma scritto in un linguaggio viene tradotto in un programma
scritto in un altro linguaggio.

    Linguaggio sorgente → linguaggio oggetto

    Il compilatore fa questa traduzione (→)

    Il programma tradotto significa la stessa cosa del programma originale

bash

$ gcc hello.c    # gcc è il compilatore
$ ./a.out        # a.out è il programma tradotto
hello           # output di a.out

1.4.2. Interpretazione

L'interprete esegue operazioni per conto del programma che viene eseguito
per farlo funzionare.
```

```
    Un interprete è solo un altro programma

bash

$ python hello.py   # python è l'interprete
hello              # output di hello.py

1.4.3. CPython è compilato o interpretato?

Tecnicamente entrambi, anche se tipicamente classificato come interpretato.

CPython compila il codice sorgente in bytecode e lo interpreta.
1.5. Modalità interattiva

Una modalità interattiva, detta anche read-eval-print loop (REPL):

    Prende singoli input utente (read)

    Li esegue (eval)

    Restituisce il risultato all'utente (print)

Tipicamente, gli interpreti hanno una modalità interattiva. Tuttavia:

    Esistono interpreti che non sono interattivi

    Esistono compilatori che sono interattivi

1.5.1. CPython è interattivo?

Sì.
bash

$ python           # abilita la modalità interattiva
>>> print("hello") # read (R) e eval (E)
hello             # print (P)
>>>               # in attesa di un prompt... (L)

1.6. Ambiente di sviluppo
Software    Versione
Ubuntu Desktop  24.04.1 LTS
Python  3.13.2
VS Code ultima

Questo tutorial su come configurare Python in VS Code.
2. Tipi, valori ed espressioni
2.1. Operatori aritmetici

Simboli che denotano operazioni aritmetiche.
Simbolo Significato
+   Addizione
-   Sottrazione
*   Moltiplicazione
/   Divisione
```

```
//  Divisione intera
**  Elevazione a potenza
^   XOR
```

2.2. Tipi di numeri

Un tipo è una categoria di valori.

Tipo	Classe	Significato
Intero	int	Numeri senza parte decimale
Virgola mobile	float	Numeri con parti decimali

Per esempio:

+, -, *, o // tra interi, il risultato è un intero

/ tra interi, il risultato è un numero in virgola mobile

2.3. Espressioni

Combinazioni di variabili, valori e operatori. Ogni espressione ha un valore.

Ordine delle operazioni:

** → * e / → + e -

Le parentesi influenzano l'ordine

python

```
>>> 12 + 5 * 6
42
>>> (12 + 5) * 6
102
```

2.4. Stringhe

Sequenze di caratteri.

Ci sono solo due operatori che funzionano con le stringhe:

+ unisce due stringhe (concatenazione)

* crea copie multiple e le concatena

python

```
>>> 'Well, ' + "it's a small " + 'world.'
"Well, it's a small world."
>>> 'Spam, ' * 4
'Spam, Spam, Spam, Spam, '
```

2.5. Valori e tipi

Ogni valore ha un tipo.

```
Valore  Tipo    Classe
2   Intero  int
42.0    Numero virgola mobile   float
"Hello, World!" Stringa str

La funzione type restituisce il tipo di qualsiasi valore.
python

>>> type(2)
<class 'int'>
>>> type(42.0)
<class 'float'>
>>> type("Hello, World!")
<class 'str'>

int, float, e str possono essere usati come funzioni per convertire valori.
python

>>> int(42.9)
42
>>> float(42)
42.0
>>> str(42.0)
'42.0'

3. Variabili, istruzioni e moduli
3.1. Variabili

Nomi che si riferiscono a valori.

Le istruzioni di assegnamento creano variabili. Queste consistono di:

    Un nome di variabile a sinistra (es. n)

    L'operatore di uguaglianza (=)

    Un'espressione a destra (es. 42)

python

>>> n = 42

Nota che:

    Un'istruzione di assegnamento non ha output

    La variabile creata può poi essere usata come espressione

python

>>> n
42
>>> n * 2
84
```

3.2. Nomi di variabili

Cosa puoi fare:

- Lunghezza arbitraria
- Lettere e numeri

Cosa non puoi fare:

- Iniziare con un numero
- Punteggiatura, eccetto _
- Keywords (es. class)

Cosa dovresti fare (convenzione):

- Lettere minuscole
- _ come separatore di parole

python

```
var = "hello, world"      # buono
multiple_words = 42       # buono
multipleWords = 84        # cattivo
```

3.3. Moduli

Collezioni di variabili e funzioni (e classi).

Le istruzioni import rendono disponibili variabili e funzioni definite in altri moduli.

- Usa tali variabili o funzioni con l'operatore punto (.)

python

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sqrt(25)
5.0
```

3.4. Istruzioni

Unità di codice che hanno un effetto, ma non un valore.
Istruzione Effetto
Assegnamento Crea una variabile e le assegna un valore
Import Importa un modulo

Valutazione vs esecuzione:

```
    Valutazione è calcolare il valore di un'espressione

    Esecuzione è eseguire un'istruzione
```

3.5. Errori

```
Errore  Riguarda    Risultato
Sintassi    Struttura del programma e regole al riguardo    Python non
esegue nemmeno il programma
Runtime Eccezioni che occorrono durante l'esecuzione    Python mostra un
messaggio d'errore e ferma il programma
Semantico   Significato Il programma non fa ciò che intendevi, ma nessun
messaggio d'errore
python

>>> n! = 42
  File "<stdin>", line 1
    n! = 42
    ^
SyntaxError: invalid syntax

python

>>> '126' / 3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

4. Funzioni

4.1. Argomenti

```
Valori forniti a una funzione quando la funzione viene chiamata.
python

>>> int('101')
101

int è la funzione chiamata e '101' è l'argomento.

Alcune funzioni possono prendere qualsiasi numero di argomenti.
python

>>> print('Any', 'number', 'of', 'arguments')
Any number of arguments
```

4.2. Definizioni di funzioni

```
Una definizione di funzione specifica il nome della funzione e le
istruzioni che vengono eseguite quando la funzione viene chiamata.
python

def f():
    print("hello")
    print("world")
```

```
    def indica l'inizio di una definizione di funzione

    f è il nome della funzione (stesse regole dei nomi di variabile)

    () dopo f indica che f non prende argomenti
```

La prima riga di una definizione di funzione è l'header, che deve terminare con due punti.

```
python

def f():
```

Il resto è il body (deve essere indentato).

```
python

    print("hello")
    print("world")
```

Per convenzione, l'indentazione è sempre di 4 spazi.

Una definizione di funzione crea un oggetto funzione.

```
python

>>> def f():
...     print("hello")
...     print("world")
...
>>> f
<function f at 0x102fd5e40>
>>> f()
hello
world
```

4.3. Parametri

Nomi di variabili usati all'interno di una funzione per riferirsi ai valori passati come argomenti.

```
python

def f(p):
    print(p)
```

f prende un singolo argomento, il cui valore è assegnato a p.

I parametri e le variabili definite all'interno di una funzione sono locali.

4.4. Argomenti keyword

Argomenti che includono il nome dei parametri.

```
python

>>> def f(p1, p2):
...     print(p1)
```

```
...     print(p2)
...
>>> f(p2="world", p1="hello")
hello
world
```

4.5. Valori di ritorno

Risultati che le funzioni restituiscono.

L'istruzione return è usata per restituire il risultato di una funzione.

```python
def repeat(word, n):
    return word * n
```

Se non c'è un'istruzione return, la funzione restituisce None.

```python
def repeat(word, n):
    print(word * n)
```

4.6. Docstring

Stringhe all'inizio delle funzioni che spiegano le interfacce.

Per convenzione, le docstring sono stringhe tra triple virgolette.

Una docstring dovrebbe:

- Spiegare concisemente cosa fa la funzione (non come funziona)
- Spiegare l'effetto dei parametri sul comportamento della funzione
- Indicare i tipi dei parametri (se non ovvi)

```python
def add(num1, num2):
    """
    Somma due numeri

    num1 : primo numero da sommare
    num2 : secondo numero da sommare
    """
    return num1 + num2
```

4.7. Traceback

Lista di funzioni che stanno eseguendo, stampata quando si verifica un'eccezione.

Quando si verifica un errore runtime, Python visualizza:

```
    Il nome della funzione che stava eseguendo

    Il nome della funzione che l'ha chiamata

    E così via...

python

>>> def f1():
...     print(p)
...
>>> def f2():
...     f1()
...
>>> f2()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in f2
  File "<stdin>", line 2, in f1
NameError: name 'p' is not defined

5. Condizionali
5.1. Operatori relazionali

Operatori che confrontano i loro operandi.
Simbolo Significato
==  Uguale a
!=  Diverso da
>   Maggiore di
<   Minore di
>=  Maggiore o uguale a
<=  Minore o uguale a
5.2. Espressioni booleane

Espressioni i cui valori sono True o False.
python

>>> 5 == 5
True

True e False sono di tipo bool (non str).
5.3. Operatori logici

Operatori che combinano espressioni booleane.
Operazione  Risultato
x or y  Se x è True, allora x, altrimenti y
x and y Se x è False, allora x, altrimenti y
not x   Se x è False, allora True, altrimenti False

Le operazioni booleane restituiscono sempre:

    0 o False per falso

    1 o True per vero
```

```
or e and restituiscono sempre uno dei loro operandi.
python

>>> 42 and True
True
>>> 1 and True
1
>>> True and 1
True

5.4. Istruzioni condizionali

Istruzioni che controllano il flusso di esecuzione in base a qualche
condizione.
python

if x > 0:
    print('x è positivo')

Le istruzioni if hanno la stessa struttura delle definizioni di funzione:

    if x > 0: è l'header

    print('x è positivo') è un blocco

Se la condizione (x > 0) è vera, allora esegui il blocco.
5.5. Condizionali concatenati

Istruzioni condizionali con una serie di rami alternativi.
python

if x < y:
    print('x è minore di y')
elif x > y:
    print('x è maggiore di y')
else:
    print('x e y sono uguali')

Non c'è limite al numero di clausole elif.

Se c'è una clausola else, deve essere alla fine.

Le condizioni sono controllate in ordine:

    Se più di una condizione è vera, viene eseguito solo il primo ramo vero

Glossario
Termine Significato
Argomento   Un valore fornito a una funzione quando la funzione viene
chiamata
Operatore aritmetico    Un simbolo che denota un'operazione aritmetica
Istruzione di assegnamento  Un'istruzione che assegna un valore a una
variabile
```

```
Blocco  Una o più istruzioni indentate per indicare che fanno parte di
un'altra istruzione
Espressione booleana    Un'espressione il cui valore è True o False
Ramo    Una delle sequenze alternative di istruzioni in un'istruzione
condizionale
Bytecode    Un insieme di istruzioni progettate per un'esecuzione
efficiente da parte di un interprete
Condizionali concatenati    Un'istruzione condizionale con una serie di
rami alternativi
Compilazione    Un programma scritto in un linguaggio viene tradotto in un
programma scritto in un altro linguaggio
Compilatore Un programma che traduce un linguaggio (linguaggio sorgente) in
un altro (linguaggio oggetto)
Concatenazione  Unire due stringhe estremità a estremità
Condizione  L'espressione booleana in un'istruzione condizionale che
determina quale ramo viene eseguito
Istruzione condizionale Un'istruzione che controlla il flusso di esecuzione
in base a qualche condizione
Docstring   Una stringa all'inizio di una funzione che spiega l'interfaccia
Operatore punto L'operatore usato per accedere a una variabile o a una
funzione definita in un altro modulo
Linguaggio dinamicamente tipizzato  Un linguaggio di programmazione in cui
i tipi sono associati ai valori e verificati a runtime
Valutazione Eseguire le operazioni in un'espressione per calcolare un
valore
Eccezione   Un errore che viene rilevato mentre il programma è in
esecuzione
Esecuzione  Eseguire un'istruzione
Espressione Una combinazione di variabili, valori e operatori.
Un'espressione ha un valore
Virgola mobile  Numeri con parti decimali
Funzione    Una sequenza nominata di istruzioni che esegue qualche
operazione. Una funzione può o meno prendere argomenti e può o meno
produrre un risultato
Corpo della funzione    La sequenza di istruzioni all'interno di una
definizione di funzione
Definizione di funzione Un'istruzione che crea una funzione
Header della funzione   La prima riga di una definizione di funzione
Oggetto funzione    Un valore creato da una definizione di funzione. Il
nome della funzione è una variabile che si riferisce a un oggetto funzione
Istruzione import   Un'istruzione che legge un file modulo e rende
disponibili le variabili e le funzioni che contiene
Intero  Numeri senza parte decimale
Interpretazione L'interprete esegue operazioni per conto del programma che
viene eseguito per farlo funzionare
Interprete  Un programma che esegue direttamente istruzioni scritte in un
linguaggio di programmazione senza richiedere compilazione
Keyword Una parola speciale usata per specificare la struttura di un
programma
Argomento keyword   Un argomento che include il nome del parametro
Variabile locale    Una variabile definita all'interno di una funzione e
che può essere acceduta solo all'interno di quella funzione
Operatore logico    Un operatore che combina espressioni booleane
Modulo  Una collezione di variabili, funzioni e classi
```

```
Oggetto Una struttura dati consistente di attributi e metodi
Programmazione orientata agli oggetti   Un paradigma di programmazione
basato sul concetto di oggetti
Operando    Uno dei valori su cui opera un operatore
Parametro   Un nome di variabile usato all'interno di una funzione per
riferirsi al valore passato come argomento
Operatore relazionale   Un operatore che confronta i suoi operandi
Valore di ritorno   Il risultato di una funzione
Errore runtime  Un errore che causa la visualizzazione di un messaggio
d'errore e l'uscita del programma
Errore semantico    Un errore che causa al programma di non fare ciò che il
programmatore intendeva (nessun messaggio d'errore visualizzato)
Istruzione  Un'unità di codice che ha un effetto, ma non un valore
Linguaggio staticamente tipizzato   Un linguaggio di programmazione in cui
i tipi sono associati alle variabili e verificati al momento della
compilazione
Stringa Una sequenza di caratteri
Errore di sintassi  Un errore che si riferisce alla struttura di un
programma e alle regole al riguardo
Traceback   Una lista delle funzioni che stanno eseguendo, stampata quando
si verifica un'eccezione
Tipo    Una categoria di valori
Variabile   Un nome che si riferisce a un valore
```