Logica booleana, Costrutto IF

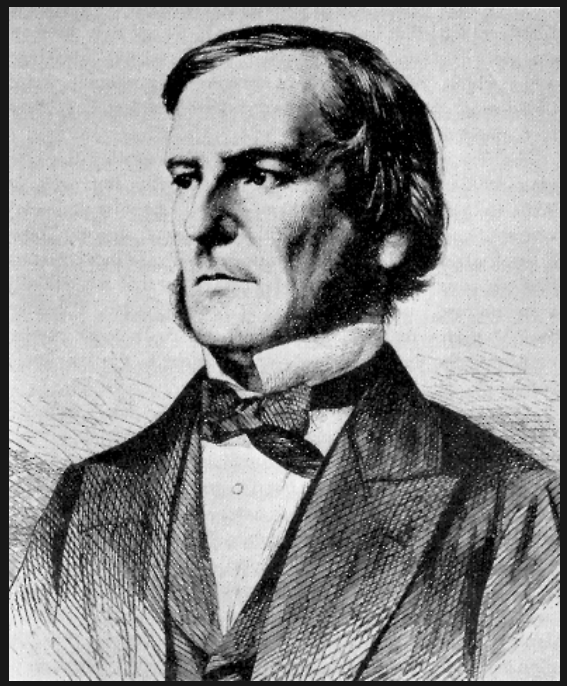
Informatica@SEFA 2017/2018 - Lezione 4

Massimo Lauria <massimo.lauria@uniroma1.it>
http://massimolauria.net/courses/infosefa2017/

Mercoledì, 4 Ottobre 2017

La logica booleana

George Boole (1815–1864)



Fondatore della logica matematica

- studio formale dei ragionamenti usati in matematica
- uso di manipolazioni algebriche per concetti logici

Variabile booleana

Python ha due valori, `True` e `False`, di tipo **booleano**.

```
type(True)
type(False)
bocciato = False
type(bocciato)
str(False)
str(True)
false
```

```
<class 'bool'>
<class 'bool'>
<class 'bool'>
'False'
'True'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'false' is not defined
```

Operatori relazionali

I valori booleani possono essere usati per rappresentare il risultato di relazioni logiche

```
1 >= 2
```

```
False
```

```
1 == (2 - 1)
```

```
True
```

```
'ia' in 'ciao'
```

```
True
```

Uguaglianza e assegnamenti

- L'operatore booleano == determina se i due operandi sono uguali.
- Il simbolo = indica un assegnamento di variable

```
variabile =   "valore assegnato"   # no output                1
variabile ==  "altra stringa"      # output interattivo       2
print(variabile)                   # output                   3
```

```
False
valore assegnato
```

Operatori booleani

Descriviamo i tre più importanti.

	Matematica	Python
negazione	$\neg x$	`not x`
congiunzione	$x \wedge y$	`x and y`
disgiunzione	$x \vee y$	`x or y`

Negazione logica $\neg x$

Assume il valore opposto della variable x

x	not x
False	True
True	False

```
porta_chiusa = False                                    1
porta_aperta = not porta_chiusa                         2
print(porta_aperta)                                     3
```

```
True
```

Domanda: a cosa è uguale `not not x`?

Congiunzione logica $x \wedge y$

La congiunzione è vera quando x e y sono entrambi veri.

x	y	x and y
False	False	False
True	False	False
False	True	False
True	True	True

Domanda: qual è il valore di

```
a1 and a2 and a2 and a4 and a5                                    1
```

Esempio di congiunzione logica

```
vento = True                    1
neve  = True                    2
tormenta = vento and neve       3
print(tormenta)                 4
```

```
True
```

Disgiunzione logica $x \vee y$

La disgiunzione è vera quando **almeno uno** tra x e y è vero.

x	y	x or y
False	False	False
True	False	True
False	True	True
True	True	True

Domanda: qual è il valore di

```
a1 or a2 or a2 or a4 or a5                                    1
```

Esempio di disgiunzione logica

```
nuvoloso  = True                                    1
pioggia   = False                                   2
brutto_tempo = pioggia or nuvoloso                  3
print(brutto_tempo)                                 4
```

```
True
```

Differenze con il linguaggio naturale

Nel linguaggio naturale si usa `or` in modo diverso

vado al mare o in montagna

intendendo alternative **esclusive**.

Invece l'`or` logico funziona in maniera differente, ne senso che il risultato è vero anche se entrambe le opzioni sono vere.

Or esclusivo `x ^ y`

L'or esclusivo (XOR) è vero quando **esattamente uno** tra x e y è vero.

x	y	x ^ y
False	False	False
True	False	True
False	True	True
True	True	False

Domanda: qual è il valore di

```
a1 ^ a2 ^ a2 ^ a4 ^ a5                                    1
```

Associatività e Commutatività

Un operatore tra due operandi, chiamiamolo $\circ$, si dice

- associativo, quando $(a \circ b) \circ c = a \circ (b \circ c)$
- commutativo, quando $a \circ b = b \circ a$

Esercizio: dimostrare che se un operatore $\circ$ è associativo e commutativo, allora comunque vengano messe le parentesi o ordinati gli operandi nella sequente espressione

$$a_1 \circ a_2 \circ a_3 \cdots a_{n-1} \circ a_n$$

il valore dell'espressione non cambia.

Il `not` precede `and` che precede `or`

```
def exclusive_or(x,y):                                  1
    return (not x and y or x and not y)                 2
                                                        3
print(exclusive_or(False,False))                        4
print(exclusive_or(True,False))                         5
print(exclusive_or(False,True))                         6
print(exclusive_or(True,True))                          7
```

```
False
True
True
False
```

Esercizi

Esercizio: Verificare che addizione e moltiplicazione sono commutativi e associativi.

Esercizio: Dimostrare che $\wedge$, $\vee$ e XOR sono associativi e commutativi.

Tabelle di verità

Formula booleana: formula di variabili booleane e operatori booleani.

$$(x \vee \neg y) \vee (\neg x \wedge y)$$

x	y	(x or (not y)) or ((not x) and y)
False	False	True
True	False	True
False	True	True
True	True	True

Distributività

$x \wedge (y \vee z)$ è uguale a $(x \wedge y) \vee (x \wedge z)$

ed anche

$x \vee (y \wedge z)$ è uguale a $(x \vee y) \wedge (x \vee z)$

Esercizio: verificare usando le tabelle di verità

- scrivere le tabelle delle quattro formule
- ogni formula ha tre variabli: la tabella ha 8 righe

Regole di de Morgan

$\neg(x \vee y)$ è uguale a $\neg x \wedge \neg y$

ed anche

$\neg(x \wedge y)$ è uguale a $\neg x \vee \neg y$

Esercizio: verificare usando le tabelle di verità

- scrivere le tabelle delle quattro formule
- ogni formula ha due variabli: la tabella ha 4 righe

Terminologia

Una formula booleana è detta

- Soddisfacibile: vera per almeno un assegnamento
- Contraddizione/Insoddisfacibile: sempre falsa
- Tautologia: sempre vera
- Falsificabile: falsa per almeno un assegnamento

Ad esempio la formula vista prima è una **tautologia**

$$(x_1 \vee \neg x_2) \vee (\neg x_1 \wedge x_2)$$

Problema SAT

data in input una formula F fatta da

- variabili booleane (i.e. `True` / `False`)
- operatori logici `and`, `or`, `not`

trovare un algoritmo **veloce** che determini se F è **soddisfacibile**

Premio: 1.000.000 di Dollari (Clay Institute)

Prendere decisioni

Scegliere le istruzioni da eseguire

É possibile eseguire delle istruzioni solo se una condizione si verifica

```
pioggia = False                                    1
nuvoloso = True                                    2
if pioggia or nuvoloso:                            3
    print("1. Prenderò l'ombrello")                4
    print("1. Prenderò le scarpe chiuse")          5
                                                   6
nuvoloso = False                                   7
if pioggia or nuvoloso:                            8
    print("2. Prenderò l'ombrello")                9
    print("2. Prenderò le scarpe chiuse")          10
```

```
1. Prenderò l'ombrello
1. Prenderò le scarpe chiuse
```

Sintassi del costrutto `if`

```
if condizione:                                                    1
    istruzione1                                                   2
    istruzione2                                                   3
    istruzione3                                                   4
    ...                                                           5
```

- `condizione` espressione dal valore booleano
- `istruzione1` **indentata** rispetto alla riga precedente
- le altre istruzioni allineate con `istruzione1`

Due alternative

Se `condizione` vera esegue il primo blocco, altrimenti il secondo.

```
pioggia = False                                    1
nuvoloso = False                                   2
if pioggia or nuvoloso:                            3
    print("Prenderò l'ombrello")                   4
else:                                              5
    print("Prenderò i sandali")                    6
```

```
Prenderò i sandali
```

Sintassi del costrutto `if else`

```
if condizione:                    1
    blocco1                       2
else:                             3
    blocco2                       4
```

oppure (anche se fa un po' schifo)

```
if condizione:                    1
      blocco1                     2
else:                             3
   blocco2                        4
```

L'indentazione dei due blocchi non deve essere uguale

Aumentiamo il numero di opzioni con `elif`

`elif` è un'abbreviazione di `else if`

```
def commenti_voto(voto):
    print("Il voto e'", voto)
    if voto < 18:
        print("mi dispiace")
    elif voto == 18:
        print("appena sufficiente")
    elif voto < 24:
        print("OK, ma potevi fare meglio")
    elif voto == 30:
        print("congratulazioni!")
    else:
        print("bene!")
```

Aumentiamo il numero di opzioni con `elif` (II)

```
commenti_voto(15)                                    1
commenti_voto(18)                                    2
commenti_voto(23)                                    3
commenti_voto(27)                                    4
commenti_voto(30)                                    5
```

```
Il voto e' 15
mi dispiace
Il voto e' 18
appena sufficiente
Il voto e' 23
OK, ma potevi fare meglio
Il voto e' 27
bene!
Il voto e' 30
congratulazioni!
```

`elif` aiuta la leggibilità del codice

Questa è una versione del codice precedente scritta senza `elif`

```
def commenti_voto(voto):                                    1
    print("Il voto e'", voto)                               2
    if voto < 18:                                           3
        print("mi dispiace")                                4
    else:                                                   5
        if voto == 18:                                      6
            print("appena sufficiente")                     7
        else:                                               8
            if voto < 24:                                   9
                print("OK, ma potevi fare meglio")          10
            else:                                           11
                if voto == 30:                              12
                    print("congratulazioni!")               13
                else:                                       14
                    print("bene!")                          15
```

Operatori booleani per le condizioni di `if`

Paragrafo 5.2: una serie di operatori

- tra numeri, stringhe, ecc…
- producono un valore booleano

Esempio:

- `==` (uguale) `!=` (diverso)
- `<`, `>`, `<=`, `>=` (comparazioni)
- `in` e `not in` (sotto stringhe)

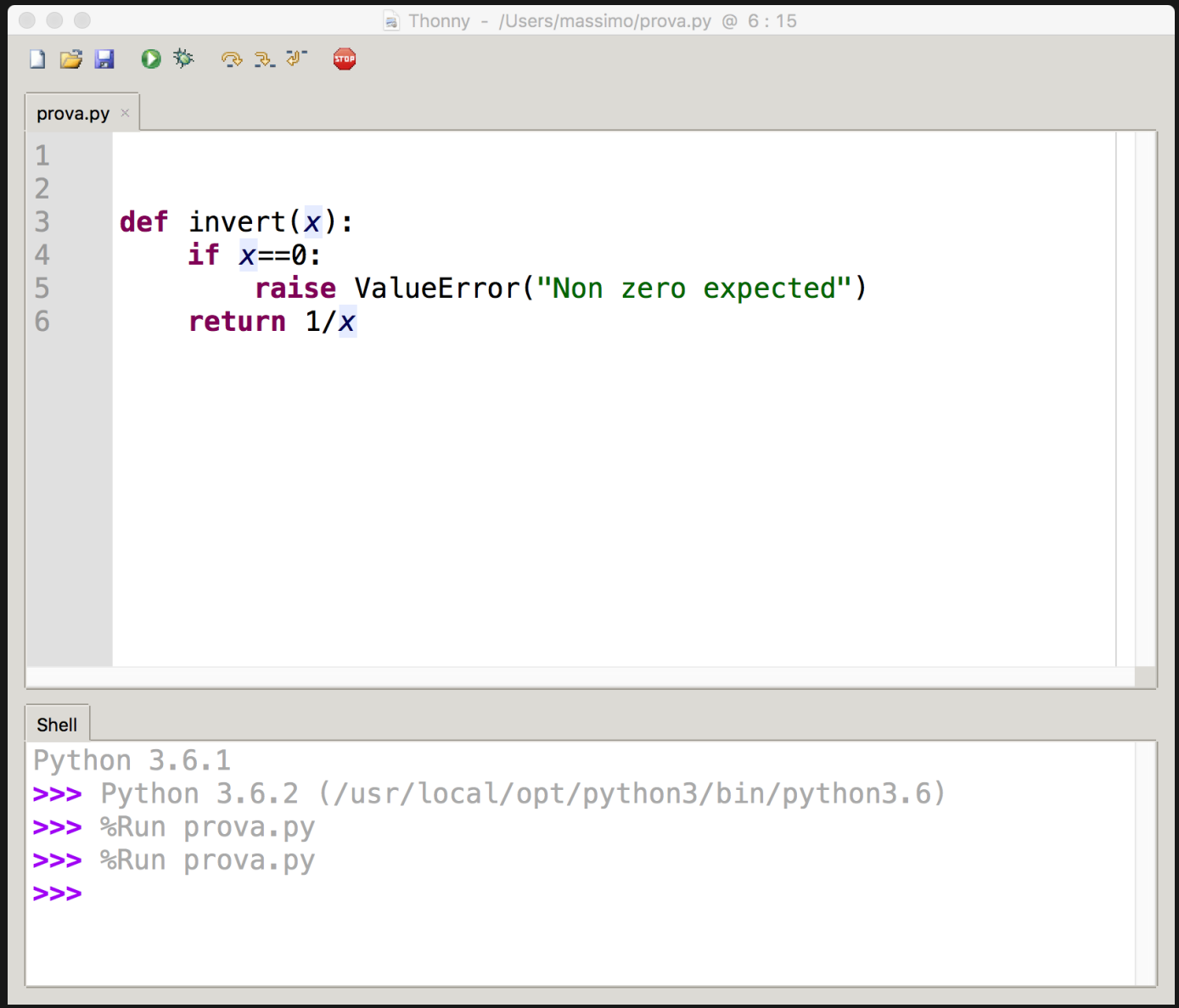
Concatenazione di operatori

```
print(0 < 9 <= 10 != 'ciao' in 'ciao a tutti')
```

```
1
```

```
True
```

Appendix 1: Thonny



Letture

Lezione di oggi:

- Capitoli 5.1–5.4