Tipi scalari e non scalari, Stringhe

Oggetti Python

- I programmi Python manipolano cosiddetti **data objects**
- **In Python tutto è un oggetto**
- Gli oggetti hanno un **tipo** che definisce il genere di cose che il programma può fare su di essi
- I tipi possono essere
  - ***scalari*** (tipi atomici, non possono essere divisi, e.g., int, float, etc.)
  - ***non scalari*** (hanno una struttura interna che può essere acceduta, e.g., liste, tuple, etc.)

Tipi scalari

- `int` – rappresenta **interi**, e.g., `5`
- `float` – rappresenta **numeri reali**, e.g., `3.27`
- `bool` – rappresenta i valori **booleani** `True` e `False`
- `NoneType` – tipo **speciale** e assume un solo valore, `None`
- Possiamo usare `type()` per vedere il tipo di un oggetto

```
>>> type(5)
int
>>> type(3.0)
float
```

what you write into the Python shell

what shows after hitting enter

Conversione di tipo (*type casting*)

- Possiamo **convertire un oggetto di un tipo in un oggetto di altro tipo**
- `float(3)` converte l'intero `3` in un float `3.0`
- `int(3.9)` tronca il float `3.9` nell'intero `3`

Stampa verso la console

- Per poter mostrare l'output dal codice verso l'utente, possiamo usare il comando `print`

```
In [11]: 3+2
Out[11]: 5

In [12]: print(3+2)
5
```

"Out" tells you it's an interaction within the shell only

No "Out" means it is actually shown to a user, apparent when you edit/run files

Operazioni aritmetiche

- `i+j` → la **somma**
- `i-j` → la **differenza**
- `i*j` → il **prodotto**
- `i/j` → **divisione**

- Se entrambi sono interi, il risultato sarà un intero
- Se entrambi sono float, il risultato sarà un float

Il risultato è un float

- `i%j` → l'operazione di **modulo** quando `i` viene diviso per `j`
- `i**j` → **elevamento a potenza** di `i` a `j`

Stringhe

- Possono contenere lettere, caratteri speciali, spazi, numeri, etc.
- Sono definite utilizzando **doppio apice (*quotation marks*) o singolo apice (*single quotes*)**
  - `hi = "hello there"`
  - `hi2 = 'hello there again'`
- **E' possibile Concatenare** stringhe
  - `name = "pippo"`
  - `greet = hi + name`
  - `greeting = hi + " " + name`
- Eseguire **operazioni** su stringhe come definito nella documentazione Python
  - `silly = hi + " " + name * 3`

Stringhe

- Possiamo pensare alle stringhe come una **sequenza** case-sensitive di caratteri
- Possiamo comparare stringhe attraverso gli operatori ==, >, < etc.
- **`len()`** è una funzione utilizzata per ottenere la **lunghezza** di una stringa passata come primo argomento

```
s = "abc"
len(s)  → ritorna 3
```

Stringhe

- Le parentesi quadre sono usate per accedere in maniera **indicizzata** ad una stringa

```
s = "abc"
```

index: 0 1 2 ← indexing always starts at 0
index: -3 -2 -1 ← last element always at index -1

```
s[0]    → ritorna "a"
s[1]    → ritorna "b"
s[2]    → ritorna "c"
s[3]    → trying to index out of bounds, error
s[-1]   → ritorna "c"
s[-2]   → ritorna "b"
s[-3]   → ritorna "a"
```

Stringhe

- Possiamo *affettare* (**slice**) le stringhe utilizzando la notazione `[start:stop:step]`
- Con [`start:stop`], `step=1` by default
- Possiamo anche utilizzare solo la notazione ::

```
s = "abcdefgh"
```

`s[3:6]` → ritorna "`def`", ovvero `s[3:6:1]`

`s[3:6:2]` → ritorna "`df`"

`s[::]` → ritorna "`abcdefgh`", ovvero `s[0:len(s):1]`

`s[::-1]` → ritorna "`hgfedbca`", ovvero `s[-1:-(len(s)+1):-1]`

`s[4:1:-2]` → ritorna "`ec`"

If unsure what some command does, try it out in your console!

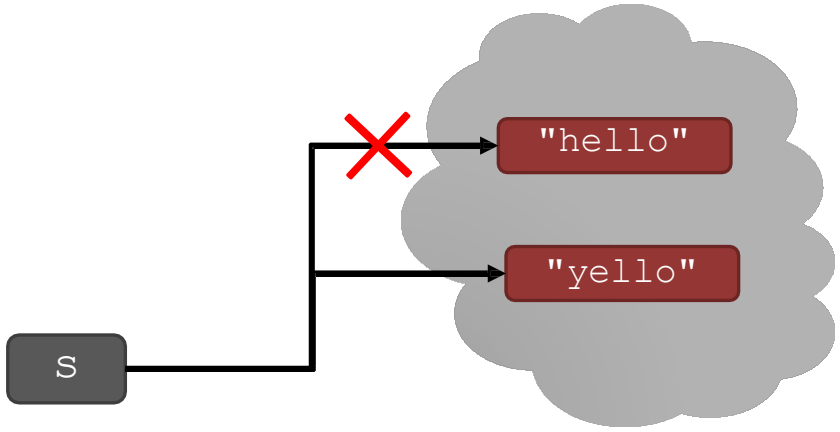
Stringhe

- Le stringhe sono "**immutabili**" – non possono essere modificate

```
s = "hello"
s[0] = 'y'
s = 'y'+s[1:len(s)]
```

→ gives an error

→ is allowed,
s bound to new object



INPUT/OUTPUT: `print`

- Utilizzato per effettuare **output** di elementi sulla console (stdout)
- La parola chiave è `print`

```
x = 1
print(x)
x_str = str(x)
print("my fav num is", x, ".", "x =", x)
print("my fav num is " + x_str + ". " + "x = " + x_str)
```

INPUT/OUTPUT: `input("")`

- La funzione **`input`** effettua la stampa del primo argomento messo tra doppi apici e attende che l'utente inserisca qualcosa da tastiera (stdin)
- Una volta che si digita INVIO, il *bind* avviene tra il valore inserito da tastiera e la variabile che viene assegnata tramite **`input`**

```
text = input("Type anything... ")
print(5*text)
```

- **`input`** **restituisce una stringa**, quindi è necessario il cast se dobbiamo operare per esempio con numeri

```
num = int(input("Type a number... "))
print(5*num)
```

Operatori di comparazione tra `int`, `float`, `string`

- Assumiamo `i` e `j` come nomi di variabile
- Le comparazioni di seguito restituiscono un booleano (True, False)

`i > j`

`i >= j`

`i < j`

`i <= j`

`i == j` → **equality** test, `True` se il valore di `i` è lo stesso di `j`

`i != j` → **inequality** test, `True` se il valore di `i` non è lo stesso di `j`

Operatori logici tra booleani

- Assumiamo `a` e `b` nomi di variabili booleane

`not a` → `True` se `a` è `False`
`False` se `a` è `True`

`a and b` → `True` se entrambi sono `True`

`a or b` → `True` se una delle due è `True`

A	B	A and B	A or B
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False