LINGUAGGIO PYTHON

VARIABILI
FUNZIONI DI BUILT-IN
PRIMI PROGRAMMI

SISTEMI INFORMATIVI E DBMS

CORSO DI LAUREA MAGISTRALE IN
MANAGEMENT E MONITORAGGIO DEL TURISMO SOSTENIBILE

PROF. ANDREA PINNA

AA 2020/2021

MEMORIZZAZIONE DEI DATI

Fino ad ora abbiamo elaborato direttamente i dati.

I linguaggi di programmazione permettono di registrare dati, nella memoria del calcolatore.
Per rendere agevole l'utilizzo dei dati memorizzati si può dare un nome allo spazio di memoria utilizzato dal dato.

Questo nome viene chiamato "variabile".

MEMORIZZAZIONE DEI DATI

Il nome della variabile deve rispettare alcune regole:

- inizia con una lettera (per convenzione minuscola)
- sono vietate le parole chiave

Esempio di nomi errati: `2variabile, and, if`
Esempio di nomi corretti: `variabile2, mioAnd, if2.`

Nella scelta del nome è conveniente favorire la comprensibilità del contenuto della variabile.

LISTA DELLE PAROLE CHIAVE

Le parole chiave (keyword) sono "nomi" di elementi specifici del linguaggio. Queste sono le keywork di Python:

```
False, class, finally, is, return, None,
continue, for, lambda, try, True, def, from,
nonlocal, while, and, del, global, not, with,
as, elif, if, or, yield, assert, else, import,
pass, break, except, in, raise.
```

Durante il corso faremo uso della maggior parte di esse. Le variabili non possono avere un nome tra questi.

MEMORIZZAZIONE DEI DATI

Un nome deve essere definito e inizializzato tramite l'operazione di assegnamento.
L'operatore di assegnamento è il simbolo =

```
miaVariabile = 1 + 1
```

Si legge: "assegna alla casella di memoria di nome miaVariabile il risultato dell'espressione a destra dell'uguale". Dopo questa istruzione, la variabile `miaVariabile` riferirà al valore 2 di tipo `int`.

MEMORIZZAZIONE DEI DATI

```
miaVariabile = 1 + 1
```

- Prima valuta il valore dell'espressione e determina il tipo di dato.
- Successivamente crea una casella di memoria capace di contenere quel tipo di dato e inserisce il valore del dato.
- Poi crea il nome della variabile (parola a sinistra dell'uguale) e lo riferisce alla casella di memoria creata.

CONFRONTO TRA VARIABILI

L'operatore == controlla solo il valore.
L'operatore "is" controlla sia il tipo che il valore.
Esempio:

```
>>> a = 2
>>> b = 2.0
>>> a is b
False
>>> a == b
True
```

SPAZIO DEI NOMI

I nomi delle variabili vengono conservati in una struttura dati dell'interprete chiamata "namespace".
Se si prova ad usare un nome non definito l'interprete genera un errore.
Provare:

```
>>> globals()
```

Vedremo più avanti che i nomi delle variabili esistono nelle porzioni di codice racchiuse in uno "scope".

USO DELLE VARIABILI

Esempio:

```
>>> miaVariabile = 1 #ASSEGNAMENTO
>>> miaVariabile #richiamo la variabile
1
```

Al nome `miaVariabile` è ora associato il valore 1

USO DELLE VARIABILI

Esempi

```
>>> miaVariabile=1
>>> risultato = miaVariabile + 3
>>> risultato += miaVariabile
>>> risultato
5
>>> type(risultato)
<class 'int'>
```

l'operatore `+=` assegna alla variabile la somma del il valore attuale della variabile con il valore a destra dell'operatore

ESPRESSIONI

L'assegnamento è sempre eseguito dopo aver valutato il risultato dell'espressione a destra dell'uguale.
L'espressione deve essere un'istruzione valida.
Può essere composta sia di valori che di funzioni

```
>>> a=-10
>>> miaVariabile=abs(a)+min(1,2)*2
```

FUNZIONE TYPE

Fino ad ora abbiamo usato la funzione `type()` per conoscere il tipo di un dato o del dato contenuto in una variabile. Ciò che scriviamo tra le parentesi si chiama argomento della funzione.

La funzione `type()` prende in ingresso un unico valore.

```
>>>type(3)
```

Questa funzione restituisce (return) un valore.

FUNZIONE TYPE

```
>>> ilTipo = type(3)
```

La funzione `type()` restituisce (`return`) un dato.
Quando una funzione restituisce un dato è possibile raccogliere questo dato assegnandolo ad una variabile.

```
>>> ilTipo
<class 'int'>
```

DI CHE TIPO È?

Possiamo verificare qual è il tipo di dato restituito da `type()`

```
>>> type(type(3))
<class 'type'>
```

Oppure, usando la variabile ilTipo della slide precedente:

```
>>> type(ilTipo)
<class 'type'>
```

FUNZIONE PRINT

La funzione `print()` è una funzione di built-in molto utilizzata nella programmazione. Permette di stampare a video gli argomenti contenuti tra le sue parentesi, sotto forma di testo. È detta funzione di output standard. Esempio:

```
>>> risultato = 2+3
>>> print(risultato)
5
```

Output a schermo

FUNZIONE PRINT

In generale la funzione `print` prende come argomenti un elenco di espressioni separate da virgola. Questa funzione non restituisce nulla. Ciò che stampa è il risultato delle espressioni separato da spazio.

Esempio:

```
>>> print(risultato, 3+4, "ciao"+"Mondo")
5 7 ciaoMondo

>>> a = print(3+4) #Metto in a il risultato della print
7
```

Sembra un dato in return

```
>>> print(a) #Ma se ora stampo il contenuto di a esce None.
None
```

FUNZIONE INPUT

I linguaggi di programmazione sono pensati per accettare dati in ingresso durante l'esecuzione del programma.

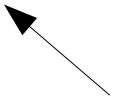
La funzione di built-in `input()` prende come argomento una stringa.
Quando eseguita. la funzione mostra a schermo la stringa e interrompe il programma per aspettare che l'utente inserisca un dato e prema invio. Questo è detto standard input.

La funzione restituisce il dato inserito dall'utente che se assegnato ad una variabile diventa utilizzabile dal programma.

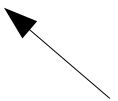
FUNZIONE INPUT

Esempio

```
>>> mioDato = input("Inserisci un valore: ")
Inserisci un valore: 3
>>> print(mioDato)
3
>>>
>>> miaStringa = "Scrivi un valore: "
>>> mioDato = input(miaStringa)
Scrivi un valore: prova
>>> print(mioDato)
prova
```



Scrivo 3 e premo invio



Scrivo prova e premo invio

ALTRE FUNZIONI DI BUILT-IN

Funzioni che operano su dati numerici:

`abs` restituisce il valore assoluto di un numero
`min` restituisce il minore tra gli argomenti
`max` restituisce il maggiore tra gli argomenti

Esempio:

```
>>> risultato = max(min(1,7),9)-abs(1-10)
>>> print(risultato)
0
```

https://docs.python.org/3/library/functions.html

PRIMO PROGRAMMA

Un programma python è una sequenza di istruzioni scritta in linguaggio python.

Per scrivere un programma è sufficiente aprire un editor di testo, scrivere le istruzioni rispettando la sintassi del linguaggio e salvare il file con estensione .py

Vi sono diversi editor di testo alternativi pensati per scrivere programmi in diversi linguaggi. Uno di questi è geany (www.geany.org)

PRIMO PROGRAMMA

In un editor di testo creiamo un nuovo file e scriviamo le seguenti due istruzioni:

```
a = "Hello, World!"
print(a)
```

Salviamo il file nel desktop (o in un'altra cartella) con nome `primo.py` e torniamo al prompt dei comandi

PRIMO PROGRAMMA

Eseguiamo il programma.
Esempio da terminale.

Cartella che contiene il file sorgente

```
~/Scrivania$ python3 primo.py

Hello, World!
```

Output del programma

Nome del file sorgente

SECONDO PROGRAMMA

In un editor di testo creiamo un nuovo file e scriviamo le seguenti tre istruzioni:

```
saluto = "Ciao"
nome = input("Come ti chiami? ")
print(saluto + " " + nome)
```

Salviamo il file nel desktop (o in un'altra cartella) con nome secondo.py e torniamo al prompt dei comandi

SECONDO PROGRAMMA

Eseguiamo il programma.

Da terminale

Nome del file sorgente

```
~/Scrivania$ python3 secondo.py
```

```
Come ti chiami? Andrea
Ciao Andrea
```

Input al programma che deve inserire l'utente.

Output del programma