Laboratorio di Informatica

Corso di Laurea Magistrale in Ingegneria per l'Ambiente e il Territorio

A.A. 2021/2022

Docente: Lorenzo Putzu

Lezione 01

Il linguaggio Python

Programmazione

- **Algoritmo**: insieme finito e ordinato di passi eseguibili in tempo finito e non ambigui (univocamente definiti), che definiscono un processo che termina.
- **Programma**: sequenza di istruzioni elementari che un computer è in grado di comprendere ed eseguire.
- **Programmare**: risolvere un problema (reale?) mediante un algoritmo.
  - Le capacità di risoluzione dei problemi prescindono dal linguaggio utilizzato.
- Algoritmo (astratto) > Programma (concreto)

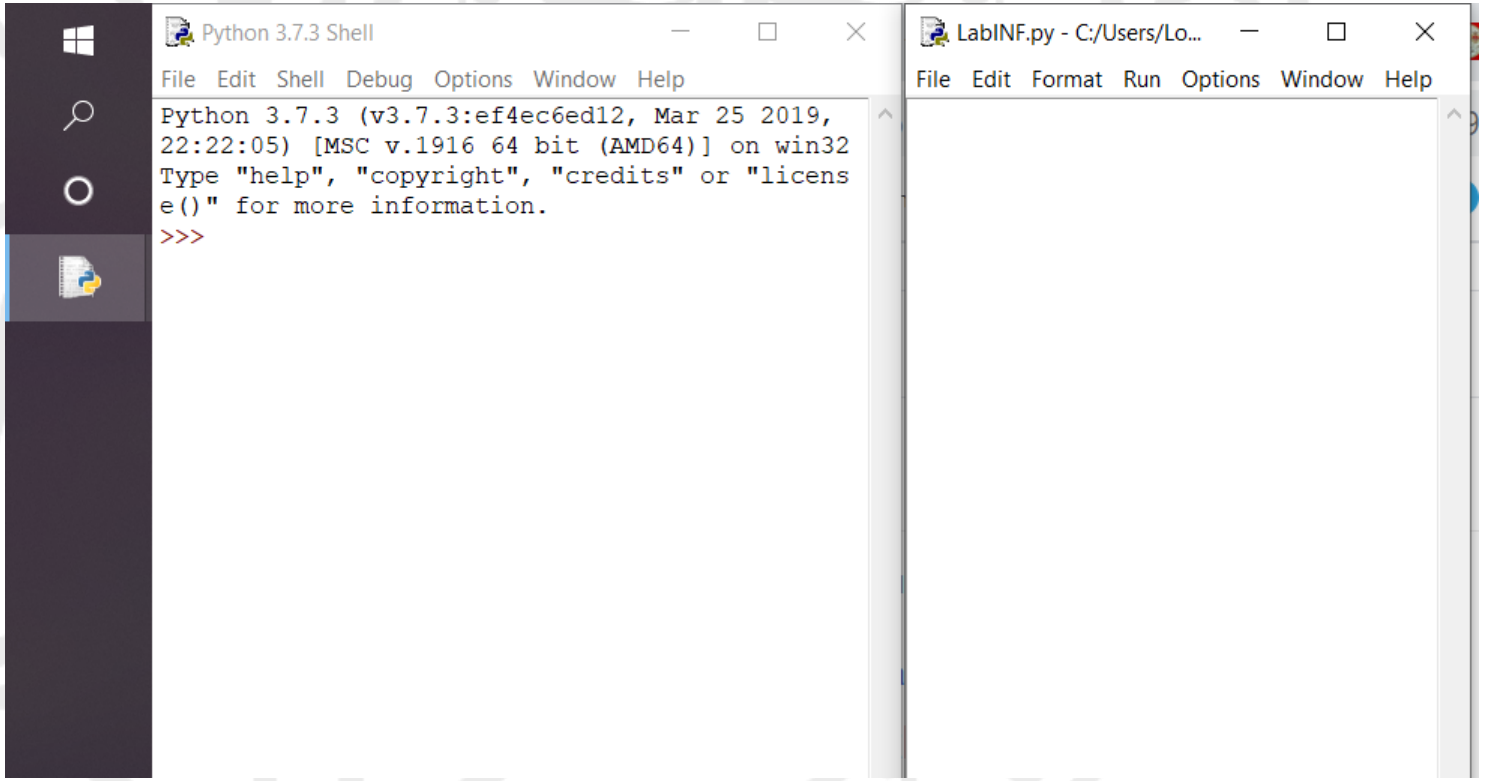
Programmazione

- Aspetti pratici della programmazione:
  - Linguaggio Python
  - Sintassi dei costrutti
  - Stesura del codice
  - Comprensione del codice

- Aspetti più generali della programmazione:
  - Tecniche di problem solving
  - Stile di programmazione
  - Efficienza

Ambiente di sviluppo

- Useremo l'ambiente di programmazione IDLE
  - disponibile per qualsiasi sistema operativo www.python.it
- Si compone di
  - Shell
  - Editor



Variabili

- Le variabili sono i luoghi in cui vengono memorizzati temporaneamente i dati di cui il programma ha bisogno per portare a termine il suo compito.

- Ogni variabile è caratterizzata da:
  - un nome univoco
  - un tipo (tipizzazione dinamica)
  - un valore

- In Python non si distingue una fase di dichiarazione e di assegnamento

C	vs	**Python**
int variabile = espressione;		variabile = espressione

Nomi Variabili

- Possono essere composti da uno o più dei seguenti caratteri:
  - lettere minuscole e maiuscole (anche accentate)
  - cifre
  - il carattere _ (*underscore*)

- **NON** devono iniziare con una cifra;
- **NON** devono coincidere con i nomi predefiniti delle istruzioni e di altri elementi del linguaggio.

Nomi Variabili

catetoUnO

2cateto

QuestaEunaVariabile

nome-casuale

def

print

underscore

variable1

Variabile1

Print

Nomi Variabili

catetoUnO

2cateto (non può iniziare con un numero)

QuestaEunaVariabile

nome-casuale (il trattino viene valutano come simbolo matematico)

def (keyword)

print (keyword)

underscore

variable1

Variabile1

Print

Variabili

- L’istruzione di assegnamento viene **eseguita** in due fasi:
  - l’interprete valuta espressione, cioè ne determina il valore
  - il valore di espressione viene assegnato (o associato) a variabile

- L’effetto concreto di un’istruzione di assegnamento è la memorizzazione del valore di espressione all’interno di una o più celle di memoria.

- Se un’istruzione di assegnamento si riferisce a una varilabile alla quale in precedenza era già stato assegnato un valore, il nuovo valore **sostituirà** quello precedente.

Espressioni e tipi di dato

- Le espressioni Python possono elaborare e produrre valori appartenenti a tre categorie principali, dette **tipi di dato**
  - numeri interi
  - numeri frazionari
  - sequenze (“stringhe”) di caratteri

- La più semplice espressione è un singolo numero

- Python distingue tra due tipi di dato numerici:
  - I numeri interi, sia positivi che negativi;
  - I numeri frazionari (*floating point*), sia positivi che negativi

Espressioni aritmetiche

- Gli operatori disponibili nel linguaggio Python sono i seguenti
  - +, -, *, /, // (quoziente intero), ** (potenza), % (modulo)

- Alcune espressioni aritmetiche possono essere
  - a = 6;
  - b = 8 + 2;
  - c = 7 - 2;
  - d = 6 * 7
  - e = 10/5;
  - f = 10%5
  - g = 10//5

Espressioni aritmetiche

- Espressioni aritmetiche più complesse si ottengono combinando numeri attraverso **operatori**
  - a = b = c
  - a = b * c + 5;
  - a = c - 7 * 5 + a;

- Valgono le regole di precedenza canoniche. Per alterare l'ordine, si utilizzano le parentesi tonde:
  - v = b * c + a * e;
  - v = b * ((c + a) * e);
  - v = ((b * c) + a) * e;

Espressioni che elaborano stringhe

- I programmi Python possono elaborare testi rappresentati come sequenze di caratteri (lettere, cifre, segni di punteggiatura) racchiuse tra **apici singoli** o **doppi**, dette stringhe. Esempi:
  - "Questa è una stringa"
  - ’Questa è una stringa’
  - "" (una stringa vuota)

- È quindi possibile assegnare una stringa a una variabile; esempi:
  - variabile = "Questa è una stringa"

Espressioni che elaborano stringhe

- Il linguaggio Python prevede alcuni operatori anche per il tipo di dato *stringa*. Uno di questi è l'operatore di **concatenazione**, che si rappresenta con il simbolo + e produce come risultato una **nuova** stringa ottenuta concatenando due stringhe qualsiasi
  - variabile = "mappa" + "mondo" → "mappamondo"
  - variabile = "Ciao" + " " + "a" + " " + "tutti" → "Ciao a tutti"

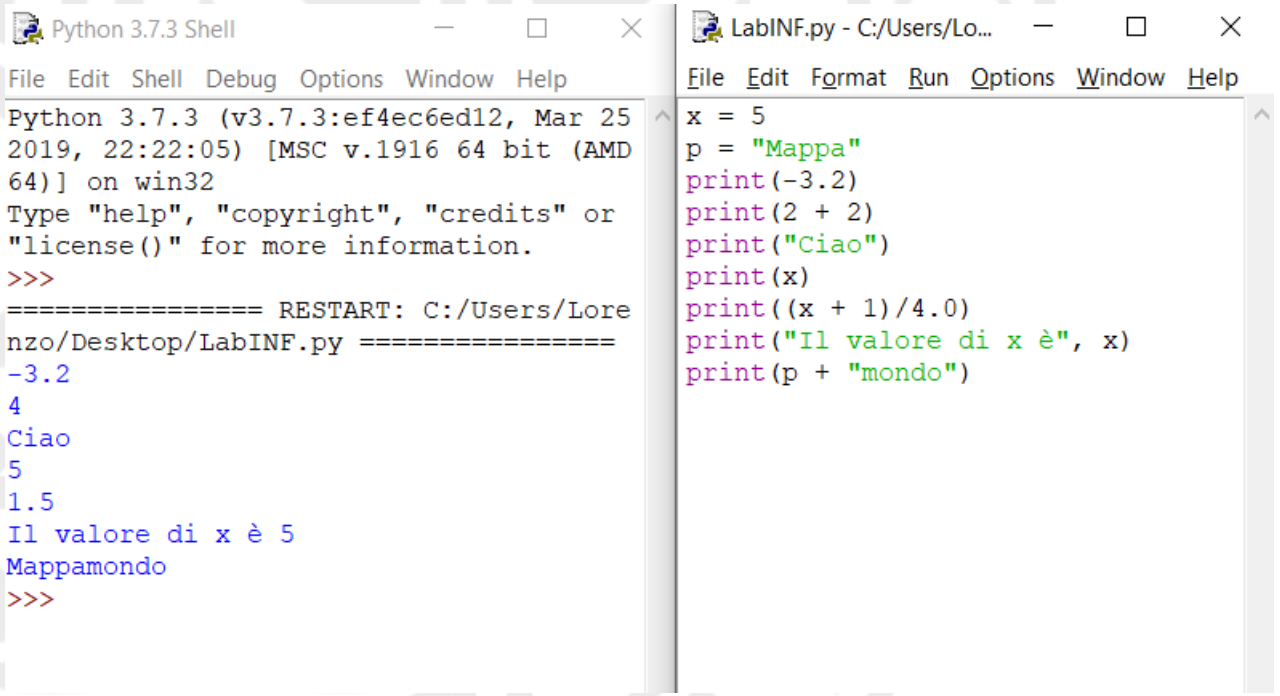
Ingresso/uscita (input/output)

- L'istruzione `print` consente di stampare nella *shell* il **valore** di una qualsiasi **espressione**
  - **Sintassi**: print(espressione)
  - non devono esserci spazi prima della keyword print
  - espressione deve essere una qualsiasi espressione valida del linguaggio Python (quindi anche variabili)

- È anche possibile stampare i valori di un numero qualsiasi di espressioni, con la seguente sintassi:
  - print(espressione1,espressione2,. . .)
  - In questo caso i valori delle espressioni vengono stampati su una stessa riga, separati da un carattere di spaziatura.

Ingresso/uscita (input/output)

- Stampa nella *shell* di una sequenza di valori.
  - x = 5
  - p = "Mappa"
  - print(-3.2)
  - print(2 + 2)
  - print("Ciao")
  - print(x)
  - print((x + 1)/4.0)
  - print("Il valore di x è", x)
  - print(p + "mondo")



Ingresso/uscita (input/output)

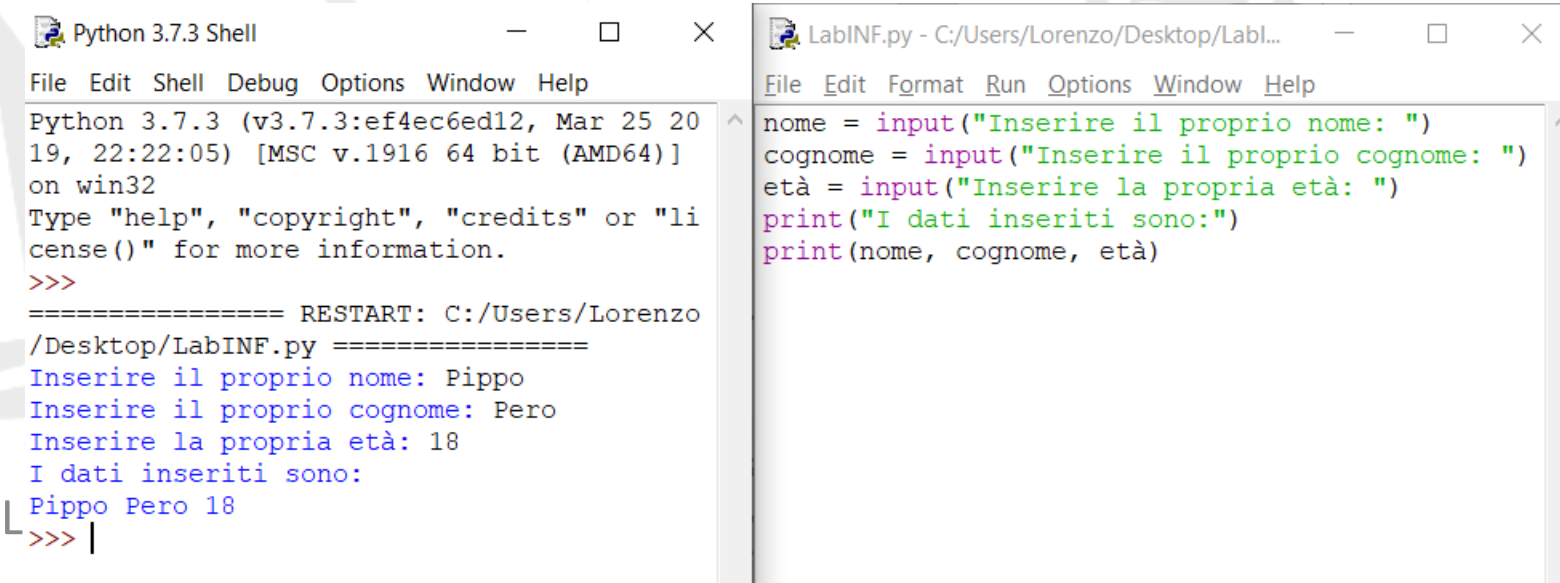
- Per inserire in una stringa un apice singolo o doppio è necessario racchiuderla tra apici dell'altro tipo, oppure far precedere l'apice dal carattere \ (detto *backslash*), senza spazi tra i due.
  - print("L'apostrofo")
  - print('L\'apostrofo')
  - print('Doppi "apici" in una stringa')
  - print("Doppi \"apici\" in una stringa")
- Per rappresentare un'interruzione di riga ("a capo", *newline*) all'interno di una stringa si usa la sequenza di caratteri \n. Esempio:
  - print("Prima riga.\nSeconda riga.")
- Dato il significato particolare del carattere \ nelle stringhe, per stamparlo testualmente si deve scrivere \\
  - print("abc\\def")

Ingresso/uscita (input/output)

- La funzione `input` è necessaria quando i valori dei dati da elaborare non sono noti nel momento della **scrittura** di un programma, ma devono essere acquisiti **durante la sua esecuzione**, attraverso la tastiera
  - **Sintassi**: variabile = input(messaggio)
  - Stampa nella *shell* messaggio (se presente) e consente all'utente, durante l'esecuzione di un programma, di immettere una qualsiasi sequenza di caratteri nella *shell*;
  - durante l'esecuzione della funzione input l'esecuzione del programma resta **sospesa** alla pressione del tasto INVIO
  - la sequenza inserita sarà memorizzata in **variabile** sotto forma di **stringa**.

Ingresso/uscita (input/output)

- Sintatticamente il costrutto input(messaggio) costituisce un tipo particolare di **espressione** Python (detta *chiamata di funzione*, come si vedrà più avanti). Infatti, analogamente alle altre espressioni, esso ha lo scopo di produrre un valore che dovrà poi essere elaborato dal programma
  - nome = input("Inserire il proprio nome: ")
  - cognome = input("Inserire il proprio cognome: ")
  - età = input("Inserire la propria età: ")
  - print("I dati inseriti sono:")
  - print(nome, cognome, età)



La funzione eval

- Se una stringa contiene una qualsiasi espressione **valida** del linguaggio Python, la funzione `eval` consente di ottenere il valore di tale espressione.
- Assumendo che x sia un'espressione di tipo **stringa**, la sintassi della chiamata di eval è la seguente:

```
eval(x)
```

- Per esempio, l'espressione eval("1+1") produce il valore 2.
- La funzione eval può quindi essere usata per "convertire" in valori numerici i dati acquisiti mediante la funzione input.

Esercizio

1. Scrivere un programma che permetta di effettuare la conversione euro dollari, assegnando alle variabili con l'importo in Euro e tasso di conversione un valore a piacere e stampare il risultato in output.