LINGUAGGIO PYTHON

GENERALITÀ
TIPI DI DATO

SISTEMI INFORMATIVI E DBMS

CORSO DI LAUREA MAGISTRALE IN
MANAGEMENT E MONITORAGGIO DEL TURISMO SOSTENIBILE

PROF. ANDREA PINNA

AA 2019/2020

C.I. PYTHON

Nome: Python
Creatore: Guido van Rossum
Data di nascita: 1991
Sito web: http://www.python.org
Versioni in uso: python 3.x e python 2.7 (in scadenza)

Segni particolari:

- È potente e facile da imparare
- Ha strutture dati efficienti di alto livello (e object oriented)
- Presenta un sintassi elegante e una natura interpretata,
- Ideale per lo *scripting* e lo sviluppo rapido di applicazioni

INTERPRETE PYTHON

L'interprete Python è disponibile a questo indirizzo https://www.python.org/downloads/

In questo corso useremo la versione 3.7

Nota: in molti sistemi Linux è preinstallato.

USARE L'INTERPRETE

Inoltre L'interprete python3 si può usare per:

- interpretare un programma scritto in python,
- eseguire le singole istruzioni (prompt).

Nel terminale eseguiamo `python3` (su linux o mac) o `python` (su windows) per ottenere il prompt Python.
Su windows si può avviare l'applicazione python dal menu start.

Il risultato sarà simile a questo:

```
Python 3.6.8 (default, Aug 20 2019, 17:12:48)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

USARE L'INTERPRETE

Dal prompt possiamo digitare una istruzione in linguaggio Python ed ottenere il risultato.

```
>>> 3+3 #premo invio
6
>>>
```

Il carattere # indica all'interprete di non considerare tutto ciò che sta alla sua destra. Si usa per inserire commenti nel codice

USARE L'INTERPRETE

Se l'interprete non è in grado di elaborare l'istruzione produce un errore.

```
>>> 3b #premo invio
  File "<stdin>", line 1
    3b
     ^
SyntaxError: invalid syntax

>>>quit() #procedura che chiude il terminale
```

ZEN PYTHON

La seguente istruzione è un "easter egg". Mostra 19 concetti della "filosofia" di programmazione in Python.

```
>>> import this
```

Vedremo più avanti l'uso normale della parola chiave `import`

FILE SORGENTE

L'interprete può essere usato per eseguire un programma. Come abbiamo già visto, un programma è una sequenza di istruzioni.

I programmi python sono salvati su file di testo semplice che contiene il codice sorgente,
Per convenzione, i codici sorgente Python hanno estensione .py

Esempio: primoprogramma.py

ESECUZIONE

Sul terminale digito il nome dell'interprete seguito dal nome del programma e premo invio.

```
$python3 primoprogramma.py
```

AMBIENTE DI SVILUPPO

Gli ambienti di sviluppo integrati (IDE) sono software che permettono sia di scrivere programmi informatici che di eseguirli. Inoltre supportano il programmatore nella correzione degli errori.

In questo corso utilizzeremo un ambiente di sviluppo chiamato Pycharm,

Ambienti online: https://repl.it/languages/Python3

TIPI DI DATO

Un dato è un "valore" che può essere un numero, una sequenza di caratteri, un insieme di valori eccetera.

Ogni dato su Python è caratterizzato da un tipo. Si dice che Python è un linguaggio tipizzato.

Ogni tipo si comporta in maniera diversa all'interno del programma.

Per conoscere il tipo di un dato possiamo usare la funzione `type()` presente in python.

TIPI DI DATO SEMPLICE

Python ha tre tipi di dato semplice

- Numeri
- Booleani
- Stringhe

TIPI DI DATO

Esempio:

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

>>> type("ciao")
<class 'str'>
```

il dato sotto esame è 3

3 è di tipo int

TIPI DI DATO: NUMERI

Numeri: rappresentano un valore numerico.

Si distinguono in numeri

- Numeri nteri: int.
- Numeri reali: float

I float hanno una parte decimale

```
>>>type(3.30)
<class 'float'>
```

TIPI DI DATO: NUMERI

I numeri possono far parte di una espressione. Le espressioni in python si costruiscono con operatori.

- Gli operatori sono simboli speciali che rappresentano operazioni (ad esempio funzioni matematiche).
- I valori utilizzati dagli operatori si chiamano operandi.

Operatori matematici:

+, -, /, * (moltiplicazione), ** (esponente).

TIPI DI DATO: NUMERI

Esempi:

```
>>> 3+3.55 #uso l'operatore di somma +
6.55
>>> type(3+3.55)
<class 'float'>
>>> 3/-2 #divide 3 per -2
-1.5
>>> 3**2 #esegue 3 elevato 2
9
```

TIPI DI DATO: NUMERI

Ordine delle operazioni e uso delle parentesi:

```
>>> 3-5/2+1 #Esegue prima le divisioni e le moltiplicazioni
1.5
>>> 3-5/(2+1) #Esegue prima il contenuto della parentesi
1.3333333333333333
>>> (3-5)/(2+1) #Equivale a -2/3
-0.6666666666666666
>>>
```

TIPI DI DATO: BOOLEANI

I booleani rappresentano due valori di verità:
`True` (vero) e `False` (falso).
Rispettano le regole dell'algebra booleana.

```
>>> True
True
>>> True or False #almeno uno è True?
True
>>> True and False #entrambi sono True?
False
>>> not True #con not ribalto il valore
False
```

TIPI DI DATO: BOOLEANI

Sono booleani i risultatati delle operazioni di confronto. Esempi:

```
>>> 10>11 #È 10 maggiore di 11?
False
>>> 10>11-3
True
```

L'interprete valuta prima le espressioni a destra e a sinistra dell'operatore.

TIPI DI DATO: BOOLEANI

Espressioni con operatori di confronto. Esempi:

```
>>> 2<3!=6>1
True
```

L'interprete valuta i diversi confronti a coppie. Il codice rischia di diventare illeggibile.

L'espressione vale `True` se tutti i confronti sono `True`. Infatti corrisponde a:

```
>>> 2<3 and 3!=6 and 6>1
True
```

TIPI DI DATO: BOOLEANI

Operatori di confronto:

>	Maggiore di	>=	Maggiore o uguale di
<	Minore di	<=	Minore o uguale di
==	Valore uguale	!=	Diverso

Esempio:

```
>>>3!=4

True
```

TIPI DI DATO: BOOLEANI

Tutti i valori numerici diversi da zero sono interpretati come True in una espressione booleana.

Esempio:

```
>>> -3 and 1
1
>>> not not 0
False
```

TIPI DI DATO: STRINGHE

Una stringa è una sequenza di caratteri racchiusa tra apici o virgolette. In Python le stringhe possono essere unite con l'operatore +.

```
>>> "Corso"+"Python" #Provare "Corso"*3
'CorsoPython'
>>> type('corso')
<class 'str'>
```

TIPI DI DATO: STRINGHE

Le stringhe sono "oggetti", ovvero dati ai quali sono associate delle operazioni specifiche chiamate metodi.

Esempio

```
>>> "casa".upper()
'CASA'
```

Vedremo in seguito alcuni metodi delle stringhe.

https://docs.python.org/3/library/stdtypes.html#string-methods

CONVERSIONE DI TIPO

Python permette di convertire un dato di un tipo in un dato di un altro tipo, ma soltanto se vi è compatibilità. La conversione di tipo viene detta casting. Per convertire un dato posso usare le specifiche funzioni.

`int(...)`	converte stringhe, float e booleani in interi. La stringa deve rappresentare un numero intero Il float viene convertito restituendo la parte intera.
`float(...)`	converte stringhe, interi e booleani in float. La stringa deve essere un numero valido.
`str(...)`	Converte interi, booleani e float in stringhe.
`bool(...)`	Converte interi, booleani e stringhe in `True` o `False` (solo se il valore da convertire è zero o la stringa vuota).

CONVERSIONE DI TIPO

Esempi:

```
>>> str(1.23)
'1.23'
>>> str(True)
'True'
>>> bool("False")
True
>>> bool(0)
False
>>> int("3")
3
>>> int("-3.2")    #ERRORE!!
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '-3.2'
>>> float("-3.2")
-3.14
```