Variabili

Introduzione

Una variabile è una "scatoletta" che racchiude un valore.

Possiamo assegnare alla scatoletta un **nome** qualunque e un **valore**, l'interprete andrà quindi a salvare questa scatoletta e il suo contenuto nel suo spazio di memoria.

Una buona pratica di programmazione è dare dei nomi **parlanti** alle variabili, cioè cercare di chiamarle in modo tale da capire subito cosa c'è dentro.

In [11]:
```
# Assegno alla variabile pippo il valore 3
pippo = 3
pluto = 5
```

L'operatore **=** in tutti i linguaggi di programmazione si chiama **operatore di assegnazione**. Infatti il suo compito è quello di assegnare a una variabile un determinato valore.

In [14]:
```
# Comando built-in di IPython che mostra tutte le variabili definite nel notebook
# con anche delle informazioni relative ad essi
%whos
```

```
Variable               Type        Data/Info
--------------------------------------------
dataframe_columns      function    <function dataframe_columns at 0x12b476e60>
dataframe_hash         function    <function dataframe_hash at 0x1039c8430>
dtypes_str             function    <function dtypes_str at 0x1039c8940>
get_dataframes         function    <function get_dataframes at 0x12b477400>
getpass                module      <module 'getpass' from '/<...>b/python3.10/getpass.py'>
hashlib                module      <module 'hashlib' from '/<...>b/python3.10/hashlib.py'>
import_pandas_safely   function    <function import_pandas_safely at 0x1039c89d0>
is_data_frame          function    <function is_data_frame at 0x12b476ef0>
json                   module      <module 'json' from '/opt<...>on3.10/json/__init__.py'>
pippo                  int         3
pluto                  int         5
```

Funzione print()

print() è una funzione! Parleremo più avanti, più nel dettaglio delle funzioni; per questa basta sapere che è una funzione *built-in* di Python, cioè che il linguaggio mette a disposizione del programmatore. La funzione **print()** semplicemente stampa il valore che gli viene passato tra parentesi.

In [15]:
```
print(pippo)
print(pluto)
```

```
3
5
```

In [17]:
```
# Questo è un errore, RUN per vedere cosa ci dice l'interprete
# print(pip)
# print(Pippo) # Il Python è case-sensitive, A != a
```

In [18]:
```
print(f"La variabile \"pippo\" ha valore: {pippo}")
print("La variabile \"pippo\" ha valore:", pippo)
```

```
La variabile "pippo" ha valore: 3
La variabile "pippo" ha valore: 3
```

Se non ci si ricorda cosa fa una funzione basta scrivere:

In [19]:
```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Esercizio 1

Assegnare alla varabile **pippo** il valore della variabile **pluto**. Deve essere una soluzione generale, deve valere per ogni valore assegnato a **pluto**!

In [22]:
```
pippo = pluto
print(pippo)
print(pluto)
```

```
5
5
```

Esercizio 2

Dato un trapezio con base maggiore, base minore e altezza rispettivamente: 100, 50, 25. Calcolarne l'area e stampare a schermo il risultato.

In [24]:
```
base_maggiore = 100
base_minore = 50
altezza = 25
area_trapezio = (base_maggiore + base_minore)*altezza/2
print("Il trapezio ha area:", area_trapezio)
```

```
Il trapezio ha area: 1875.0
```

Esercizio 3

Assegnare ad **a** un valore, alla variabile **b** un altro valore e scambiarli. Attenzione: ciò vuol dire che l'algoritmo deve funzionare per ogni valore di **a** e **b**

In [26]:
```
a = 7
b = 9
print("a =", a)
print("b =", b)
c = b
b = a
a = c
print("a =",a)
print("b =", b)
```

```
a = 7
b = 9
a = 9
b = 7
```

Tipi di una variabile

Una variabile ha sempre un tipo, può immagazzinare valori con tipi diversi. Il **tipo** denota la natura di una variabile. Python è molto intelligente (per nostra fortuna) e in base al valore che assegnamo a una variabile cambia il tipo in modo dinamico, si die che è **tipato dinamicamente** (concetto leggermente avanzato, i linguaggi di programmazione qui sono diversi tra loro, non tutti sono tipati dinamicamente).

In [28]:
```
a = 10 # int -> questa variabile è di tipo intero
b = 5.5 # float -> questa variabile è di tipo float
c = "Ciao a tutti" # str -> questa variabile è una stringa
d = True # bool -> questa variabile è un booleano
```

Per verificare il tipo di una variabile si può utilizzare una funzione built-in di Python.

In [30]:
```
type(a)
```

Out[30]:
```
int
```

In [31]:
```
type(b)
```

Out[31]:
```
float
```

In [32]:
```
type(c)
```

Out[32]:
```
str
```

In [33]:
```
type(d)
```

Out[33]:
```
bool
```

Int e float

Provare a indovinare i tipi delle nuove variabili.

In [35]:
```
a = 10
b = 5.5
```

In [36]:
```
print(a+a)
type(a+a)
```

```
20
```

Out[36]:
```
int
```

In [37]:
```
print(a+b)
type(a+b)
```

```
15.5
```

Out[37]:
```
float
```

In [38]:
```
c = 5.5
print(b+c)
type(b+c)
```

```
11.0
```

Out[38]:
```
float
```

Stringhe

Esistono molti modi per interagire con delle stringhe.

In [40]:
```
testo = "Questo corso di informatica è proprio bello, programmare mi sta divertendo un sacco!"
type(testo)
```

Out[40]:
```
str
```

In [41]:
```
frase = 'Che bella giornata'
type(frase)
```

Out[41]:
```
str
```

In [42]:
```
intero_o_stringa = '345'
type(intero_o_stringa)
```

Out[42]:
```
str
```

Si può utilizzare la funzione built-in per estrarre la lunghezza della stringa.

In [51]:
```
len(testo)
```

Out[51]:
```
84
```

Metodi

Metodo	Descrizione
capitalize()	Converte il primo carattere in maiuscolo
count()	Restituisce il numero di volte in cui un valore specifico appare in una stringa
index()	Cerca nella stringa un valore specifico e restituisce la posizione in cui è stato trovato
isalnum()	Restituisce True se tutti i caratteri nella stringa sono alfanumerici
isalpha()	Restituisce True se tutti i caratteri nella stringa sono lettere
isascii()	Restituisce True se tutti i caratteri nella stringa sono caratteri ASCII
isdecimal()	Restituisce True se tutti i caratteri nella stringa sono numeri decimali
isdigit()	Restituisce True se tutti i caratteri nella stringa sono numeri
islower()	Restituisce True se tutti i caratteri nella stringa sono minuscoli
lower()	Converte una stringa in minuscolo
swapcase()	Inverte maiuscole e minuscole: le maiuscole diventano minuscole e viceversa
title()	Converte il primo carattere di ogni parola in maiuscolo
upper()	Converte una stringa in maiuscolo

Esempi:

In [6]:
```
# capitalize()
frase = "ciao a tutti!"
print("Capitalize:", frase.capitalize())  # Converte il primo carattere in maiuscolo

# isdigit()
testo = "12345"
print("È composto solo da numeri?:", testo.isdigit())  # True, perché tutti i caratteri sono numeri
testo2 = "123a45"
print("È composto solo da numeri?:", testo2.isdigit())  # False, c'è una lettera

# lower()
parola = "HELLO WORLD"
print("In minuscolo:", parola.lower())  # Converte tutta la stringa in minuscolo

# swapcase()
parola = "Python è Fantastico!"
print("Swapcase:", parola.swapcase())  # Converte maiuscole in minuscole e viceversa

# isalnum()
stringa = "Python3"
print("È alfanumerico?:", stringa.isalnum())  # True, contiene solo lettere e numeri
stringa2 = "Python 3!"
print("È alfanumerico?:", stringa2.isalnum())  # False, ci sono spazi e caratteri speciali
```

```
Capitalize: Ciao a tutti!
È composto solo da numeri?: True
È composto solo da numeri?: False
In minuscolo: hello world
Swapcase: pYTHON È fANTASTICO!
È alfanumerico?: True
È alfanumerico?: False
```

Booleani

Concetto molto importante in programmazione, utilizzato per valutare se un'espressione qualsiasi sia Vera (**True**) o falsa (**False**). Concetto molto legato al **bit**, l'unità standard utilizzata dai computer (**0**, **1**)

In [58]:
```
a = 5<7
type(a)
```

Out[58]:
```
bool
```

In [59]:
```
10<9
```

Out[59]:
```
False
```

In [60]:
```
9<10
```

Out[60]:
```
True
```

Bisogna fare attenzione a una questione: in Python, ma in generale in tutti i linguaggi di programmazione, tutti i valori sono **True**, pochi sono i valori **False**; utilizziamo una funzione built-in per capire meglio il concetto.

In [62]:
```
bool(False)
bool(None)
bool(0)
bool("")
bool(())
bool([])
bool({})
```

Out[62]:
```
False
```

In [63]:
```
bool(10)
bool("ciao")
```

Out[63]:
```
True
```

Funzione input()

Una importantissima funzione built-in di Python è la funzione **input** che permette all'utente di dare degli input testuali al programma e salvarli in una variabile.

In [66]:
```
input()
```

Out[66]:
```
'Ciao'
```

In [67]:
```
user = input("Inserire nome utente")
```

In [68]:
```
pin = input("Inserire PIN\n")
```

In [69]:
```
print(type(user))
print(type(pin))
```

```
<class 'str'>
<class 'str'>
```

Esercizio 4a

Dato il bilancio del vostro portafoglio, inserire una spesa appena eseguita e aggiornare il bilancio.

In [71]:
```
bilancio = 120
spesa = input("Inserire l'importo della nuova spesa")

# Errore, Python non riesce a fare la differenza tra un intero e una stringa
# risultato = bilancio - spesa
# print("Il nuovo bilancio è pari a:")
# print(risultato)
```

Casting

Il casting è l'azione usata per specificare il tipo di una variabile. Ci sono casi, ad esempio l'esercizio precedente, in cui si vuole specificare il tipo di nua variabile per poter eseguire determinate operazioni successive. Data una variabile x, possiamo eseguire 3 diversi tipi di casting:

- casting a stringa -> str(x)
- casting a intero -> int(x)
- casting a float -> float(x)

In [73]:
```
x = 10
type(x)
```

Out[73]:
```
int
```

In [74]:
```
str(x)
print(x)
print(type(x)) # Notare che qui x è sempre un intero, il casting non sostituisce in generale il tipo di una variabile
```

```
10
<class 'int'>
```

In [75]:
```
x = str(x)
type(x)
```

Out[75]:
```
str
```

Esercizio 4b

Riprovare con le nuove conoscenze acquisite. Dato il bilancio del vostro portafoglio, inserire una spesa appena eseguita e aggiornare il bilancio.

In [77]:
```
bilancio = 120
spesa = input("Inserire l'importo della nuova spesa")

risultato = bilancio - int(spesa)
print("Il nuovo bilancio è pari a:")
print(risultato)
```

```
Il nuovo bilancio è pari a:
95
```

In [78]:
```
# Oppure
bilancio = 120
spesa = input("Inserire l'importo della nuova spesa")

spesa = int(spesa) # ricordarsi di sovrascrivere la variabile con il nuovo tipo
risultato = bilancio - spesa
print("Il nuovo bilancio è pari a:")
print(risultato)
```

```
Il nuovo bilancio è pari a:
95
```

Esercizio 5

Calcolo area e perimetro di un rettangolo

- Dare il benvenuto all'utente
- Chiedere all'utente di inserire i valori di base e altezza di un rettangolo
- Calcolare area
- Calcolare perimetro
- Stampare area e perimetro della figura

In [80]:
```
# Presentazione e benvenuto all'utente
print("Benvenuto, posso aiutarti a calcolare l'area e il perimetro di un rettangolo.")
print("Forniscimi base e altezza.")

# Salvo i valori di base e altezza
b = float(input("Base:\n"))
h = float(input("Altezza:\n"))

# Calcolo l'area
area = b*h

# Calcolo il perimetro
perimetro = b+b+h+h

# Stampo a schermo i risultati
print("Il perimetro del rettangolo vale: " + str(perimetro))
print("L'area del rettangolo vale: " + str(area))
```

```
Benvenuto, posso aiutarti a calcolare l'area e il perimetro di un rettangolo.
Forniscimi base e altezza.
Il perimetro del rettangolo vale: 40.0
L'area del rettangolo vale: 96.0
```