**5 23/11/2020 **

Domande su setting environment con Anaconda 3, Python 3.8, Spyder 4?

Introduzione alla Programmazione in Python 2.pptx

Testi consigliati: Boscaini - imparare a programmare con python (ed. Apogeo)



**Intro Python: variabili e tipi**



Variabili (dai semplici numeri ai più complessi dizionari): dare nomi ad oggetti e poi valori (di inizializzazione o assegnazione). Le Assegnazioni, come i Comandi (es. print – parole riservate per eseguire funzioni), sono esempi di statement.

Operatori numerici (addizione **+** sottrazione **–** divisione **/** moltiplicazione **\*** Potenza **\*\*** mod **%** quante volte un numero sta in un altro e che differenza lascia)

Operatori logici (**> < == != and or**)

Tipi (**Int** integers **4** **-2000**, **Float** floating-point numeri decimali **3.14159**, **Bool** Booleans **True** **False**, **Str** strings **'stringa di testo'** **"0718061** **" "** **"** the empty string **'$p3cial CH4R4CT3r$!!** **'**, the **NoneType**, a special type that represents the absence of a value ovvero il valore **None**)

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo di dato | Nome | Descrizione | Esempi |
| Intero | int | Intero di dimensione arbitraria | -42, 0, 1200, 999999999999999999 |
| Reale | float | Numero a virgola mobile | 3.14, 1.23e-10, 4.0E210 |
| Booleano | bool | Per valori veri o falsi | True, False |
| Complesso | complex | Numeri complessi con parte reale e immaginaria | 3+4j, 5.0+4.1j, 3j |
| Stringhe | str | Usata per rappresentare testo | '', 'stefano', "l'acqua" |
| Bytes | bytes | Usata per rappresentare bytes | b'', b'\x00\x01\x02', b'Python' |
| Liste | list | Una sequenza mutabile di oggetti | [], [1, 2, 3], ['Hello', 'World'] |
| Tuple | tuple | Una sequenza immutabile di oggetti | (), (1, 2, 3), ('Python', 3) |
| Insiemi | set/frozenset | Un’insieme di oggetti unici | {1, 2, 3}, {'World', 'Hello'} |
| Dizionari | dict | Una struttura che associa chiavi a valori | {}, {'nome': 'Ezio', 'cognome': 'Melotti'} |

**Esempio: metri quadri di case**

kitchen = 200

living\_room = 400

bedroom = 300

bathroom = 100

print(type(bedroom))

# True or False (occhio alle maiuscole: contano!)

piùgrande = living\_room > kitchen

print(type(piùgrande))

print(piùgrande)

house = kitchen + living\_room + bedroom + bathroom

print(house)

# casting: conversione tipi (stringhe-interi, interi-float, …)

garage = '50 '

# house = house + garage (darebbe errore)

house = house + int(garage)

print(house)

# False vale 0 True vale 1

piùgrande=int(piùgrande)

print(piùgrande)

# raddoppio le dimensioni di una casa

house = house\*2

print(house)

# calcolo le dimensioni complessive se costruisco 2 case

houses = house\*2

print(id(house))

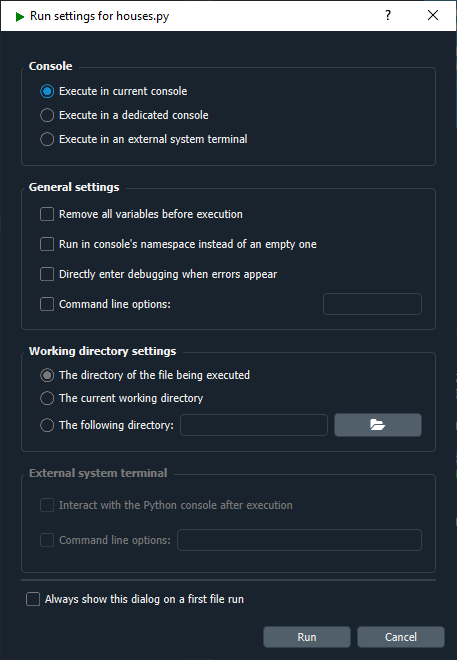
print(id(houses))

print(houses)

houses.py

è differente eseguire codice da righe di comando (IPYTHON) o salvare ed eseguire un file .py

la prima volta che si lancia un programma pyton in SPYDER 4 viene richiesto se eseguirlo nella console corrente (o altrimenti



**Esercitazione: convertire minuti in ore e minuti**

minutes\_to\_convert = 123

# poi provare con i valori 60 e 30

hours = int(minutes\_to\_convert / 60)

minutes = minutes\_to\_convert % 60

print("Hours: ")

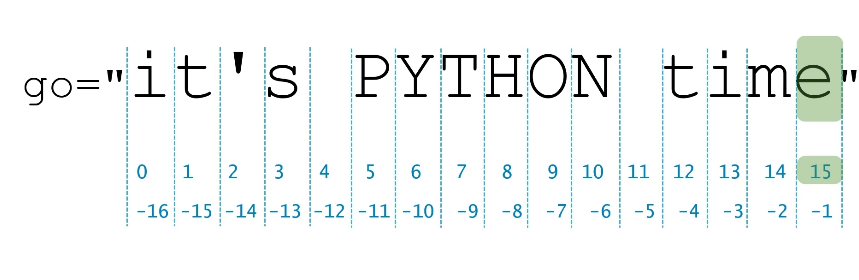
print(hours)

print("Minutes: ")

print(minutes)

è nel file conversione\_min\_ore.py

**Esercitazione stringhe**

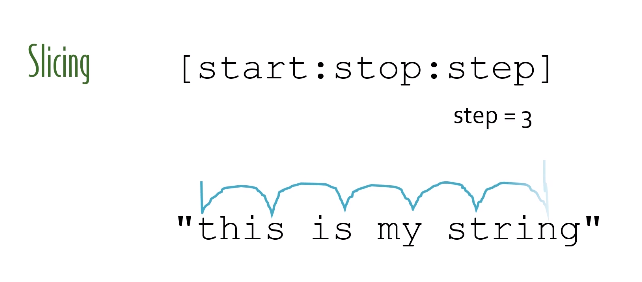


go = "it's PYTHON time"

print(go[10])

print(go[10] == go[-6])

# SLICING (estrarre parti di stringa)



Lo step omesso (predefinito) è 1

# it’s (con spazio finale)

print(go[0:5:1])

# YHNt (fino al 14° carattere, escluso)

print(go[6:14:2])

# it's PYTHON time (tutta la stringa al contrario)

# emit NOHTYP s'ti

print(go[-1:-17:-1])

# (col -1 vado all’indietro) stampo “P s”

print(go[5:2:-1])

# lunghezza stringhe: PYTHON ha 6 caratteri – empty string 0

print(len("PYTHON"))

print(len(''))

# maiusc, minusc, solo prima lettera maiuscola, inverto casi

print(go.upper())

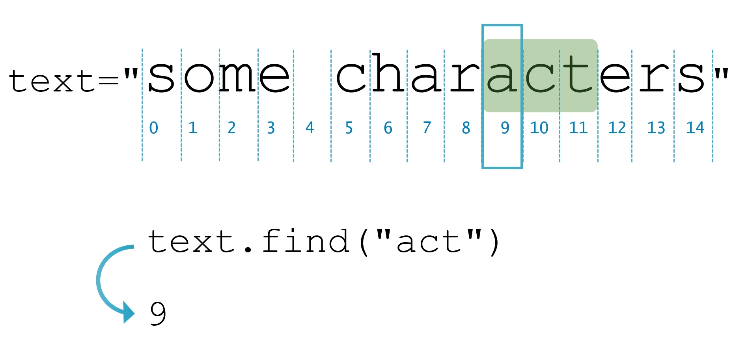
print(go.lower())

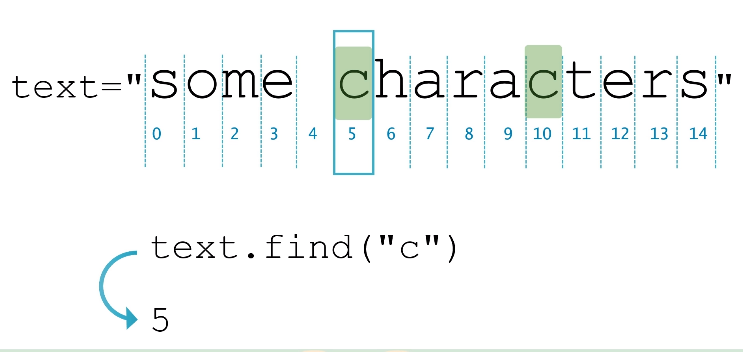
print(go.capitalize())

print(go.swapcase())

è nel file stringa\_go\_PYTHON.py

FIND() – trova la prima occorrenza / ed è anche boolean (trovato vero/falso)





address = "123 Memory Lane"

# falso (-1) (false)

address.find("street")

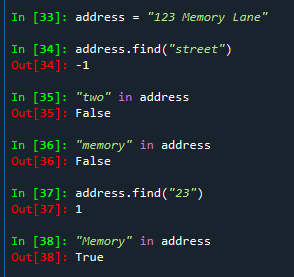
"two" in address

"memory" in address

# vero (1) (true)

address.find("23")

"Memory" in address



COUNT() – quante volte la stringa appare

# e appare 2 volte

address.count ("e")

# car mai - 0

address.count ("car")

REPLACE() – sostituisce testo con altro (con output generato a video)

address=address.replace ("Lane", "Street")

CONCATENAZIONE

address=address+", Coventry (UK)"

print(address)

123 Memory street, Coventry (UK)

num="1"

num\_repeat=num\*4

address=address+" "+num\_repeat

print(address)

123 Memory Street Coventry (UK) 1111

**errori (stoppano esecuzione del codice e dei comandi successivi)**

SyntaxError "me"2 (Python ignora il significato del comando)

NameError year=year+1 (il nome “year” non è mai stato definito)

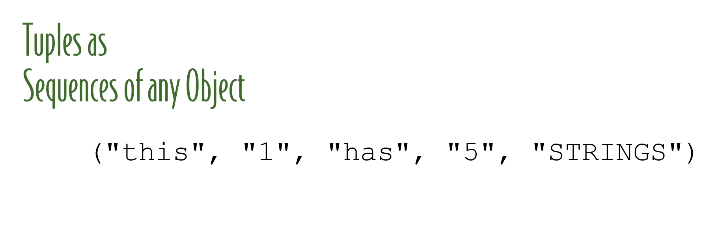
TypeError "one"+1 (somma di una stringa con un intero)

IndexError name = "Ana" print(name[3]) (il 4° carattere non esiste)

IndentationError indentazioni/tabulazioni di parti del codice scorrette

**Tuple**

A differenza delle stringhe possono memorizzare qualsiasi tipo di dato

****

# una tupla con 1 boolean, 1 stringa, 1 intero

(True, "this", 1)

# tupla di tuple (nested tuples)

((True, "this", 1), (8, 0), ("-", "\*"))

# tupla con 1 solo elemento / occhio alla virgola finale

("single",)

Si applicano i comandi LEN e valgono le posizioni in indice

nums=(4,(5,100,3),6,7)

len(nums)

Out: 4

nums[1]

Out: (5, 100, 3)

nums[1][0]

Out: 5

Operazioni matematiche con tuple

(1,2) + (3,4)

Out: (1, 2, 3, 4)

(1,2) \* 3

Out: (1, 2, 1, 2, 1, 2)

Swap di valori tra variabili

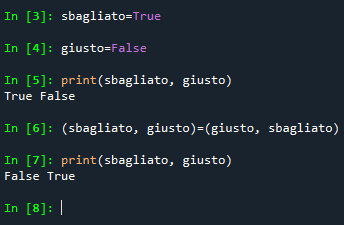
sbagliato=True

giusto=False

print(sbagliato, giusto)

(sbagliato, giusto)=(giusto, sbagliato)

print(sbagliato, giusto)



**interattività con utente: far vedere a schermo (print), catturare input (input)**

baseM=float(input("Base Maggiore? "))

basem=float(input("Base Minore? "))

altezza=float(input("Altezza? "))

areaTrapezio=(baseM+basem)\*altezza/2

print("Formula Area Trapezio:")

print("(Base Maggiore + Base minore) \* altezza / 2")

print("Esecuzione:")

print("(", baseM, "+", basem, ") \*", altezza, "/", 2, "=", areaTrapezio)

Out ( 22.0 + 6.0 ) \* 10.0 / 2 = 140.0

**introdurre decisioni IF/AND/OR/ELIF/ELSE**

age = int(input("Quanti anni hai? "))

if age<0:

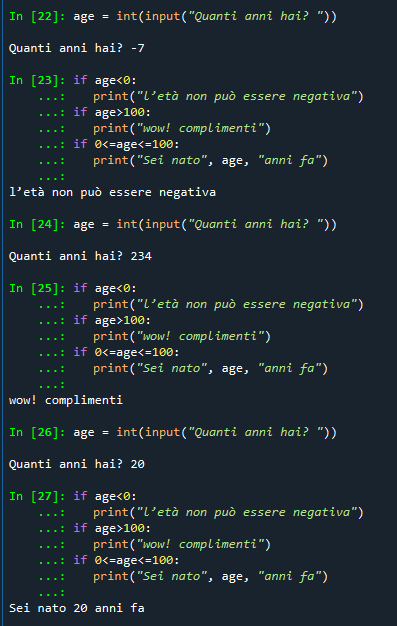
print("l’età non può essere negativa")

if age>100:

print("wow! complimenti")

if 0<=age<=100:

print("Sei nato", age, "anni fa")

****

# tolgo il casting ad int per testare la prima condizione

age = input("Quanti anni hai? ")

inserisco True

if type(age)==str or type(age)==bool:

print("Puoi inserire solo numeri interi")

elif age>=0 and age<=100:

print("Sei nato", age, "anni fa")

elif age>100:

print("wow! complimenti")

else:

print("L’età non può essere negativa")

NB attenzioni alle indentazioni, che non si perdano col copia e incolla

**ripetere task (loops)**

FOR (conosciamo esattamente quante volte ripetere)

per ripetere N volte tutte le istruzioni indentate sotto il for. Per la parola “Hello world”

word = input("What word to spell? ")

N=len(word)

for counter in range(N):

print(counter+1, " :")

print(" ", word[counter])

Range(N) significa incremento un ciclo che va da 0 a N-1 con step 1, ma posso personalizzare con

for counter in range(start, end, step):

*per avere un ciclo che va da start a end-1 con step pari a step*

non solo con contatore numerico: per quanti caratteri ha una stringa

for ch in "Hello world":

if ch != " ":

print(ch)

WHILE (ripetiamo in base al verificarsi di condizioni)

Evitando l’infinite loop / la variabile da testare nel ciclo (a differenza del contatore di FOR) va inizializzata

text = input("Proseguo? SI o NO: ")

while text == "SI":

print("ho fatto un giro")

text = input("Ancora? SI o NO: ")

text2 = input("Puoi ancora interrompere… interrompo? SI o NO: ")

if text2 == "SI":

break

**break** interrompe il ciclo. Un’altra parola riservata **continue** invece forza il ciclo a proseguire



Crea\_password.py

