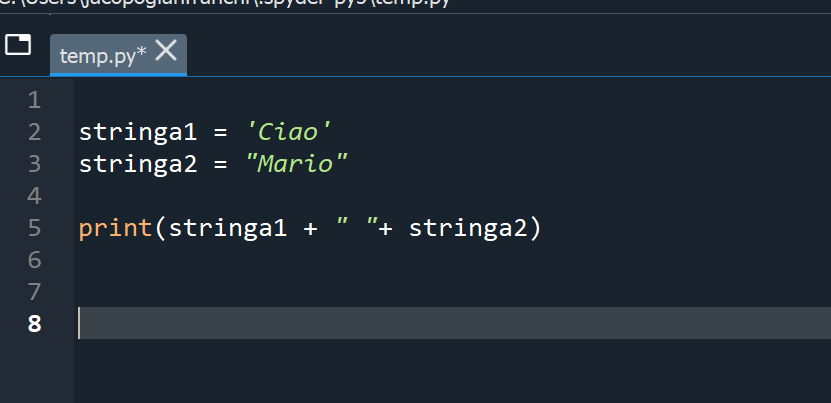
**INTRODUZIONE A PYTHON**

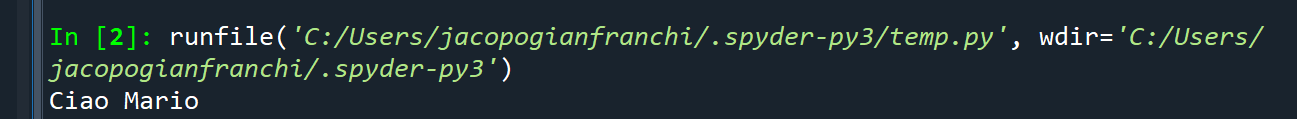
In Python le stringhe vengono inserite in singoli oppure in doppi apici. Per stampare qualsiasi cosa in Python si utilizza la funzione print()

Esempio

Codice Python:



Console:



In generale, si utilizzando i singoli apici, come nel caso di stringa1.

In generale, le funzioni possono essere dichiarate,richiamate o invocate. Si utilizzando entrambi i termini indistintamente.

Dichiarazione funzione:

Immagine che contiene testo

Descrizione generata automaticamente

Richiamo o invocazione della funzione:

Immagine che contiene testo

Descrizione generata automaticamente

In generale, quindi, una funzione viene prima dichiarata e successivamente richiamata o invocata (richiamare o invocare sono sinonimi).

Al momento della dichiarazione, x e y sono definiti parametri. Al momento del richiamo, i valori passati al posto di x e y (nel nostro caso 10 e 5) sono definiti argomenti, o input.

In altre parole, al momento del richiamo, i valori passati alla funzione sono argomenti, ossia i valori assegnati ai rispettivi parametri x e y.

In una prima fase del nostro studio, le funzioni sono già presenti in python. Non abbiamo quindi bisogno di dichiararle. Si passerà direttamente alla fase del richiamo. La funzione vista ora, print, è stata quindi direttamente richiamata (o invocata).

**PRINT() FUNCTION**

Ripassiamo il richiamo della funzione print. Il suo funzionamento lo abbiamo già visto nel paragrafo precedente.

Supponiamo di commettere un errore di sintassi. Invece di print, richiamiamo la funzione praaant

Codice Python:

stringa = "ciao a tutti"

praaant(stringa)

Console:

Immagine che contiene testo

Descrizione generata automaticamente

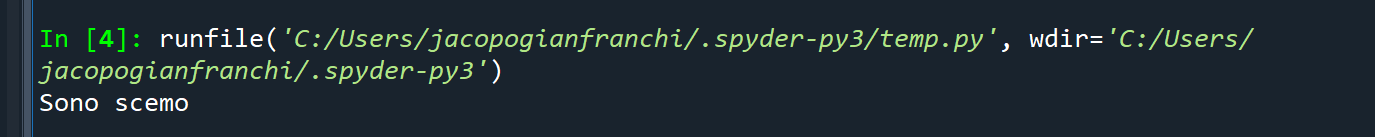
Quindi, nel caso in cui si richiama una funzione non esistente, otterremo un errore del tipo “name ‘funzione inesistente’ is not defined.

Un’altra cosa che si nota è che, nonostante le stringhe vadano scritte circondate da apici, tali apici non compaiono nella stampa.

Codice Python:

print('Sono scemo')

Console:



Gli apici non compaiono quindi nell’output.

Come facciamo a farli comparire?

Supponiamo di voler stampare un apostrofo. Se proviamo a inserirlo a buffo nella stringa, otterremo un errore:

Codice Python:

frase = 'Nell'orario scolastico non si fa casino'

print(frase)

Console:

Immagine che contiene testo

Descrizione generata automaticamente

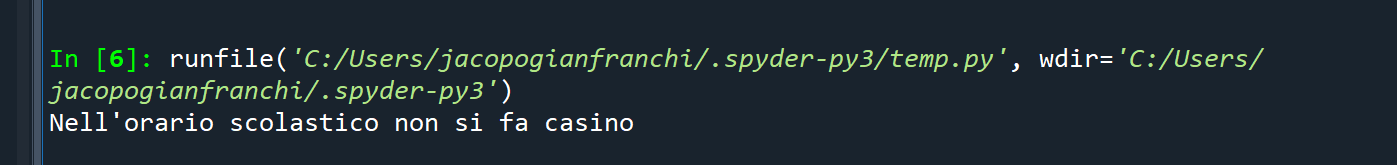
Non va bene. Bisogna utilizzare un simbolo chiamato “carattere di escape”. Tale simbolo è il seguente: \

Codice Python:

frase = 'Nell\'orario scolastico non si fa casino'

print(frase)

Console:



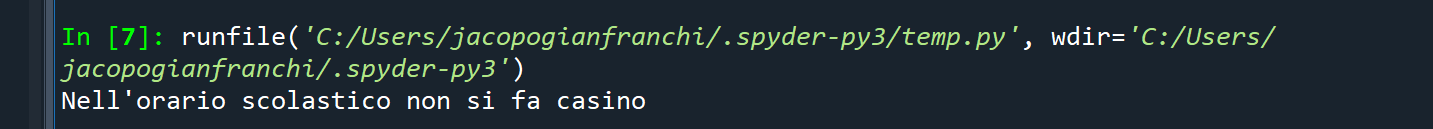
Un altro modo, anche se meno utilizzato, è quello di circondare il testo dai doppi apici. In questo modo possiamo utilizzare tutti i singoli apici che vogliamo all’interno della frase.

Codice Python:

frase = "Nell\'orario scolastico non si fa casino"

print(frase)

Console:



Proviamo ora a inserire dure o più richiami della funzione print sulla stessa linea:

Codice Python:

frase1 = "Nell\'orario scolastico non si fa casino"

frase2 = 'Ieri l\'albero contro cui sono andato a sbattere mi ha fatto malissimo all\'uccello'

frase3 = 'L\'oculista mi ha detto che sto diventando cieco'

print(frase1) print(frase2) print(frase3)

Console:

Immagine che contiene testo

Descrizione generata automaticamente

Per risolvere, si mettere un solo print() per ogni riga:

Codice Python:

frase1 = "Nell\'orario scolastico non si fa casino"

frase2 = 'Ieri l\'albero contro cui sono andato a sbattere mi ha fatto malissimo all\'uccello'

frase3 = 'L\'oculista mi ha detto che sto diventando cieco'

print(frase1)

print(frase2)

print(frase3)

Console:

Immagine che contiene testo

Descrizione generata automaticamente

Supponiamo di voler inserire una riga vuota tra una stampa e l’altra. Per farlo, si utilizza la funzione print() senza passare alcun argomento al suo interno.

Codice Python:

frase1 = 'Nell\'orario scolastico non si fa casino"'

frase2 = 'Ieri l\'albero contro cui sono andato a sbattere mi ha fatto malissimo all\'uccello'

frase3 = 'L\'oculista mi ha detto che sto diventando cieco'

print(frase1)

print()

print(frase2)

print()

print(frase3)

Console:

Immagine che contiene testo

Descrizione generata automaticamente

Ripasso. Consideriamo la seguente stringa e la seguente stampa.

frase = 'La mia tv è samsung'

print(frase)

frase è una variabile (vediamo bene dopo le variabili). Rispatto alla funzione print, “frase” è un parametro o un argomento?

La risposta è argomento. Ricordiamo che le funzioni vengono prima dichiarate e poi richiamate/invocate. Noi stiamo lavorando con il richiamo della funzione print 🡺 all’interno del richiamo si inseriscono gli argomenti. La funzione print è già dichiarata in python. Noi lavoriamo solamente richiamandola.

Come detto, frase1, frase2 e frase3 sono”variabili”. Al loro interno vengono inseriti degli oggetti. Nel nostro caso, gli oggetti sono di tipo stringa. Ovviamente vedremo in seguito le variabili in dettaglio, per ora è sufficiente immaginarle come scatole al cui interno ci sono degli oggetti. In gergo informatico, si dice che una variabile punta ad un oggetto. Nel nostro esempio, quindi, la variabile frase1 punta ad un oggetto avente valore Nell’orario scolastico non si fa casino. Il tipo dell’oggetto al quale frase1 punta è stringa.

Quindi, la nostra funzione print ha lo scopo di stampare il valore dell’oggetto al quale una variabile punta (punta è voce del verbo “puntare”).

Infatti, nonostante si utilizza il simbolo di uguale per assegnare un oggetto ad una variabile, in realtà ha piu senso assegnare una freccia.

ESEMPIO

frase 🡺 ‘Ho 30 anni’

frase è una scatola che contiene un oggetto di tipo string avente valore ‘ho 30 anni’.

In gergo informatico, frase è una variabile che punta ad un oggetto di tipo string e avente valore ‘ho 30 anni’.

Come facciamo a vedere il valore dell’oggetto al quale la variabile frase1 punta? Si richiama la funzione print e si passa il nome della variabile come argomento.

Come si fa a capire qual è il tipo dell’oggetto al quale il nostra variabile punta? Si richiama una funzione chiamata type(), anch’essa già dichiarata in Python, e si passa il nome della variabile puntante all’oggetto ‘Ho 30 anni’ come argomento. Avremo così type(frase). Per visualizzare il tipo di questo oggetto, si passa la funzione type(frase) come argomento della funzione print(). Anzi, per l’esattezza, si passa la funzione type(frase) come argomento al richiamo della funzione print():

Vediamo tutto in python:

Codice Python:

# Dichiarazione variaile frase e suo puntamento all'oggetto avente valore 'Ho 30 anni'

# di tipo string:

frase = 'Ho 30 anni'

# Stampa del valore dell'oggetto al quale la variabile frase punta:

print(frase)

# Assegnazione ad un'altra variabile del tipo dell'oggetto al quale la

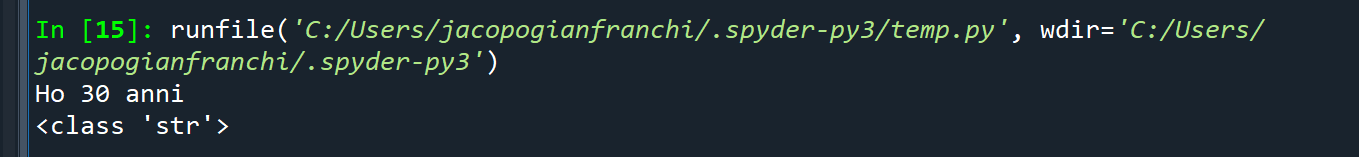
# variabile frase punta:

tipo\_frase = type(frase)

# Stampa del tipo dell'oggetto al quale la variabile frase punta:

print(tipo\_frase)

Console:



Per una questione di comodità, da ora in poi diremo “stampa della variabile”. Ricordiamo però che questo tipo di linguaggio è privo di senso in realtà. Noi non stampiamo una variabile, ma stampiamo il valore dell’oggetto al quale la variabile punta, il che è completamente diverso!

Possiamo stampare le nostre tre variabili frase1, frase2 e frase3 utilizzando una sola riga? Certo!

Vediamo qui sotto come si fa

Codice Python:

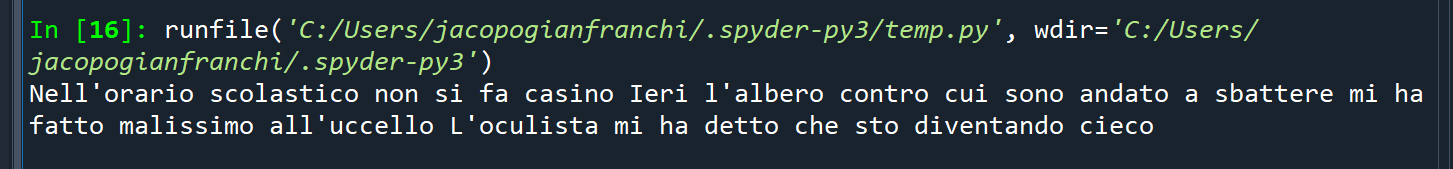
frase1 = "Nell\'orario scolastico non si fa casino"

frase2 = 'Ieri l\'albero contro cui sono andato a sbattere mi ha fatto malissimo all\'uccello'

frase3 = 'L\'oculista mi ha detto che sto diventando cieco'

print(frase1,frase2,frase3)

Console:

.

L’output fa cagare. Conviene mettere un po di punteggiatura e un po di spazi nelle nostre variabili iniziali!

Codice Python:

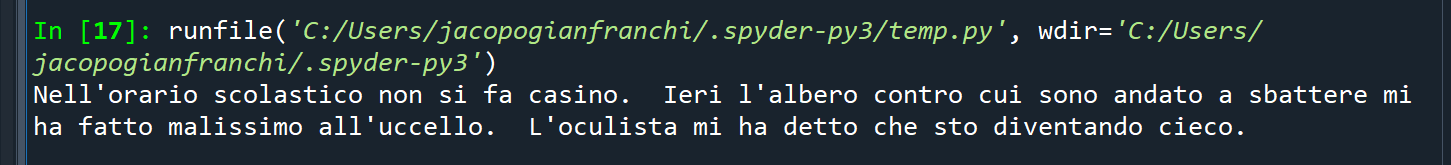
frase1 = "Nell\'orario scolastico non si fa casino. "

frase2 = 'Ieri l\'albero contro cui sono andato a sbattere mi ha fatto malissimo all\'uccello. '

frase3 = 'L\'oculista mi ha detto che sto diventando cieco.'

print(frase1,frase2,frase3)

Console:



Un po meglio. Supponiamo ora di voler rendere il nostro output ancora più leggibile, e di voler mettere una frase su ogni riga. Come si fa?

Abbiamo due strade.

La prima l’abbiamo vista a pagina 6: Possiamo stampare il valore delle variabili utilizzando tre funzioni print() differenti, ognuna su una riga a se stante.

Il secondo modo è quello di utilizzare un carattere di escape. Ne abbiamo già visto uno, che serve ad inserire gli apici all’interno delle stringhe.  
Ora vedremo un carattere di escape che ci permette di mandare a capo tutto ciò che viene dopo l’inserimento di tale carattere. Il carattere di escape è \n

Codice Python

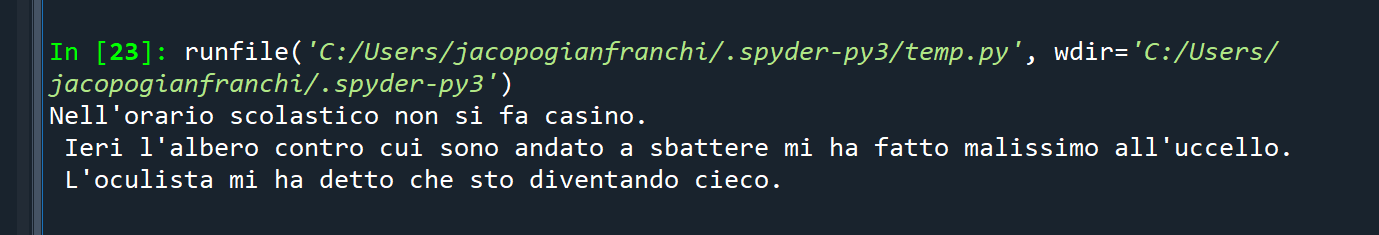
frase1 = 'Nell\'orario scolastico non si fa casino.\n'

frase2 = 'Ieri l\'albero contro cui sono andato a sbattere mi ha fatto malissimo all\'uccello.\n'

frase3 = 'L\'oculista mi ha detto che sto diventando cieco.\n'

print(frase1,frase2,frase3)

Console:



Molto meglio.

Si noti che, rispetto alla prima frase, la seconda e la terza hanno uno spazio all’inizio. Perché? Perché se utilizziamo la virgola all’interno della funzione print, tale funzione aggiunge in automatico uno spazio tra le varie variabili che si stampano.

Consideriamo ora la seguente variabile:

frase = 'Sono tre i colori preferiti da Giuda: 1.Nero 2.Rosso 3.Ocra Gialla'

Supponiamo di voler mandare a capo la lista. Vediamo come farlo con i caratteri di escape. Vediamo così anche il carattere di escape TAB, che sposta il testo verso destra

Codice Python:

frase = "Sono tre i colori preferiti da Giuda:\n\t1.Nero\n\t2.Rosso\n\t3.Ocra Gialla"

print(frase)

Console:

Immagine che contiene testo

Descrizione generata automaticamente

**ESERCIZIO**

**Immagine che contiene testo

Descrizione generata automaticamente**

**VARIABILI**

Come detto in precedenza, le variabili puntano ad un oggetto. Per ora abbiamo visto solo oggetti di tipo string. Presto vedremo anche altri tipi.

Ogni oggetto, come sappiamo, ha un valore, un tipo, e si trova in un determinato indirizzo di memoria. Per capire qual è l’indirizzo di memoria dell’oggetto al quale la variabile punta, si utilizza la funzione id sull’oggetto in questione.

Costruiamo quindi un oggetto e vediamone valore, tipo e il corrispettivo indirizzo di memoria nel quale si trova.

Codice Python:

frase1 = 'Nell\'orario scolastico non si fa casino.\n'

# Stampa valore dell'oggetto al quale frase1 punta:

print(frase1)

print()

# Stampa tipo dell'oggetto al quale frase1 punta:

print(type(frase1))

print()

# Stampa indirizzo di memoria al quale frase1 punta:

print(id(frase1))

Console:

Immagine che contiene testo

Descrizione generata automaticamente

Quindi, la variabile frase1 punta ad un oggetto avente valore pari a ‘Nell'orario scolastico non si fa casino’, è di tipo string e si trova nell’indirizzo di memoria 2672715257968.

Si noti che una variabile va prima creata, poi la si fa puntare ad un oggetto e poi può essere utilizzata.

Vediamo un esempio.

Consideriamo due stringhe, e concateniamole tra loro tramite l’operatore +. Vedremo che, se una variabile non è stata creata, non può essere utilizzata.

Per prima cosa vediamo l’esempio corretto.

Codice Python:

frutto1 = 'Mela,\t'

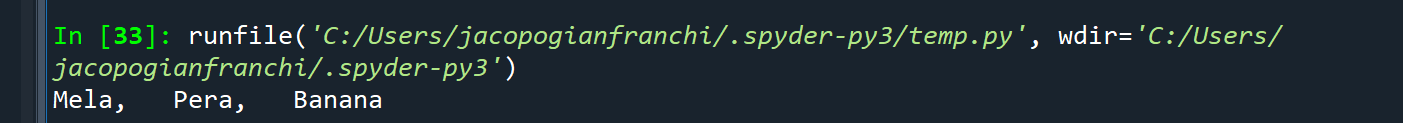
frutto2 = 'Pera,\t'

frutto3 = 'Banana'

frutta = frutto1 + frutto2 + frutto3

print(frutta)

Console:



Proviamo ora a cambiare l’ordine delle cose. Concateniamo le nostre variabili, ma definiamo frutto3 dopo la concatenazione. Cosa succederà? Avremo ovviamente un errore!

Codice Python:

frutto1 = 'Mela,\t'

frutto2 = 'Pera,\t'

frutta = frutto1 + frutto2 + frutto3

frutto3 = 'Banana'

print(frutta)

Console:

Immagine che contiene testo

Descrizione generata automaticamente

**VARIABLES NAME CONVENSION**

La creazione delle variabili ha delle regole. Esse non possono iniziare per numero e non possono essere delle keyword, come get, from etc. Vedremo le keyword più avanti.

**DATA TYPES**

Finora abbiamo visto variabili che puntano a stringhe. E’ arrivato il momento di vedere altri data types.

Codice Python:

# DATA TYPES

# STRINGHE

stringa = 'O sole miooo'

print(type(stringa))

print()

# INTEGER

numero\_intero = 70

print(type(numero\_intero))

print()

# FLOATING

numero\_float1 = 10.5

print(type(numero\_float1))

print()

numero\_float2 = 5.

print(type(numero\_float2))

print()

# BOOLEAN

isVerde = True

print(type(isVerde))

# Il Booleano può avere solo due valori: True e False. Più avanti vedremo che True corrisponde a 1, mentre False corrisponde a 0

print()

# Si noti che inserendo tra apici i diversi tipi di variabili esse si trasformeranno in stringhe!!!

a = 'Ciao!'

b = '30'

c = '14.23'

d = 'True'

print(type(a),type(b),type(c),type(d))

Console:

Immagine che contiene testo

Descrizione generata automaticamente

**COMMENTI**

I commenti si inseriscono tramite cancelletto #. Essi sono solo esplicativi: non fanno parte del codice.

Codice Python:

# Questo è un commento. Tutto ciò che si trova qui non fa parte del codice. Facciamo una prova. Creiamo una variabile in questo commento.   x = 4

# Vedremo che x non esisterà, in quanto scritto all'interno di un commento!!

print(x)

Console:

Immagine che contiene testo

Descrizione generata automaticamente

**NOTAZIONI SCIENTIFICHE**

Codice Python:

# NOTAZIONI SCIENTIFICHE

# Consideriamo innanzitutto due interi. Consideriamo un numero >= 1000. E' possibile

# Utilizzare una notazione che permette la separazione degli zeri, in modo

# Da migliorarne la leggibilità.

x = 12000000

x\_zeri\_separati = 12\_000\_000

# Stampiamo il valore di x e di x\_zeri\_separati

print(x,x\_zeri\_separati)   # 12000000 12000000

print()

# Stampiamo il tipo di x e di x\_zeri\_separati

print(type(x),type(x\_zeri\_separati)) # <class 'int'> <class 'int'>

print()

# Stampiamo l'area di memoria di x e x\_zeri\_separati

print(id(x),id(x\_zeri\_separati)) # 2672715269136 2672715269136

print()

# Perfetto! x ed x\_zeri\_separati hanno stesso valore, stesso id e stessa area di

# memoria ==> si dice che x ed x\_zeri\_separati sono identici, ossia entrambe

# le variabili puntano alla stessa area di memoria.

# Ne approfitto per ricordare che il linguaggio utilizzato finora è improprio.

# Noi non abbiamo stampato il valore di x e di x\_zeri\_separati. Abbiamo stampato

# il valore dell'oggetto al quale le variabili x ed x\_zeri\_separati puntano.

# Stesso discorso vale per type e per id.

# Vediamo ora un modo più comodo per stampare variabili aventi vari zeri.

# Consideriamo la seguante variabile di tipo int. Come al solito si ricorda che il

# linguaggio utilizzato è impreciso. Non è la variabile ad essere di tipo int, ma è

# l'oggetto al quale la variabile punta ad essere di tipo int.

numero\_intero = 3000000

print(numero\_intero)   #3000000

print()

# Possiamo scrivere questo numero in modo più "contratto". Per prima cosa si conta

# il numero di zeri. In questo caso sono sei. Possiamo allora riscrivere la

# variabile numero\_intero nel seguente modo:

numero\_intero\_con\_e = 3e6

print(numero\_intero\_con\_e) #3000000.0

print()

# oppure:

numero\_intero\_con\_E = 3E6

print(numero\_intero\_con\_E) #3000000.0

print()

# Si noti che l'utilizzo di e o di E restituisce un float e non un int.

# Supponiamo ora di voler stampare un numero decimale. Prendiamo ad esempio

# 0.00002. Abbiamo quattro zeri dopo la virgola, e uno prima della virgola.

# Nel nostro esempio abbiamo quindi in tutto cinque zeri. 0.00002 equivale quindi

# a 2E-5, oppure 2e-5

numero\_decimale = 0.00002

print(numero\_decimale) # 2e-05

print()

numero\_decimale\_con\_E = float(2E-5)

print(numero\_decimale\_con\_E) #2e-05

print()

numero\_decimale\_con\_e = float(2e-5)

print(numero\_decimale\_con\_e) # 2e-05

print()

# Nella stampa vediamo che i numeri decimali li stampa con la notazione e-,

# Compreso il nostro numero decimale di partenza (riga 81)

# Consideriamo ora delle misure delle unità di misura diverse per i numeri.

# Ovviamente, si lavora sempre con numeri in base decimale. Tuttavia, raramente

# si utilizza l'unità di misura ottale e esadecimale.

# Unità di misura ottale.

# In python, un numero in base ottale inizia per 0o oppure per 0O (E' indifferente)

# Il numero in base ottale verrà automaticamente convertito in base decimale.

# Qui sotto nel word vedremo come avvengono le trasformazioni da ottale a decimale e

# viceversa, ossia da decimale a ottale. Per ora vediamo solo un esempio.

numero\_ottale = 0o13

print(numero\_ottale) # 11

print()

# Unità di misura esadecimale.

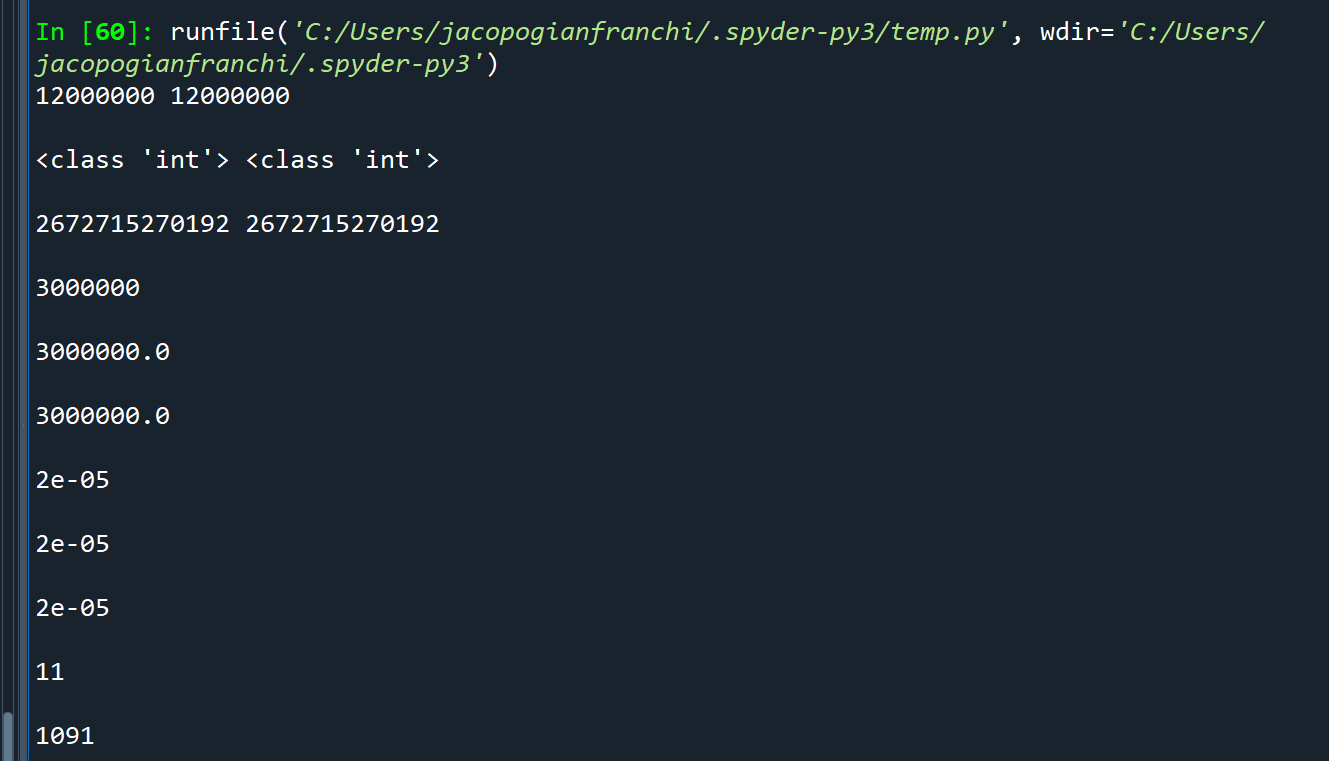
# Anche qui vedremo nel word come si trasforma un numero esadecimale in decimale e

# viceversa.  Un numero esadecimale inizia con 0x o, indistintamente, con 0X

numero\_esadecimale = 0X443

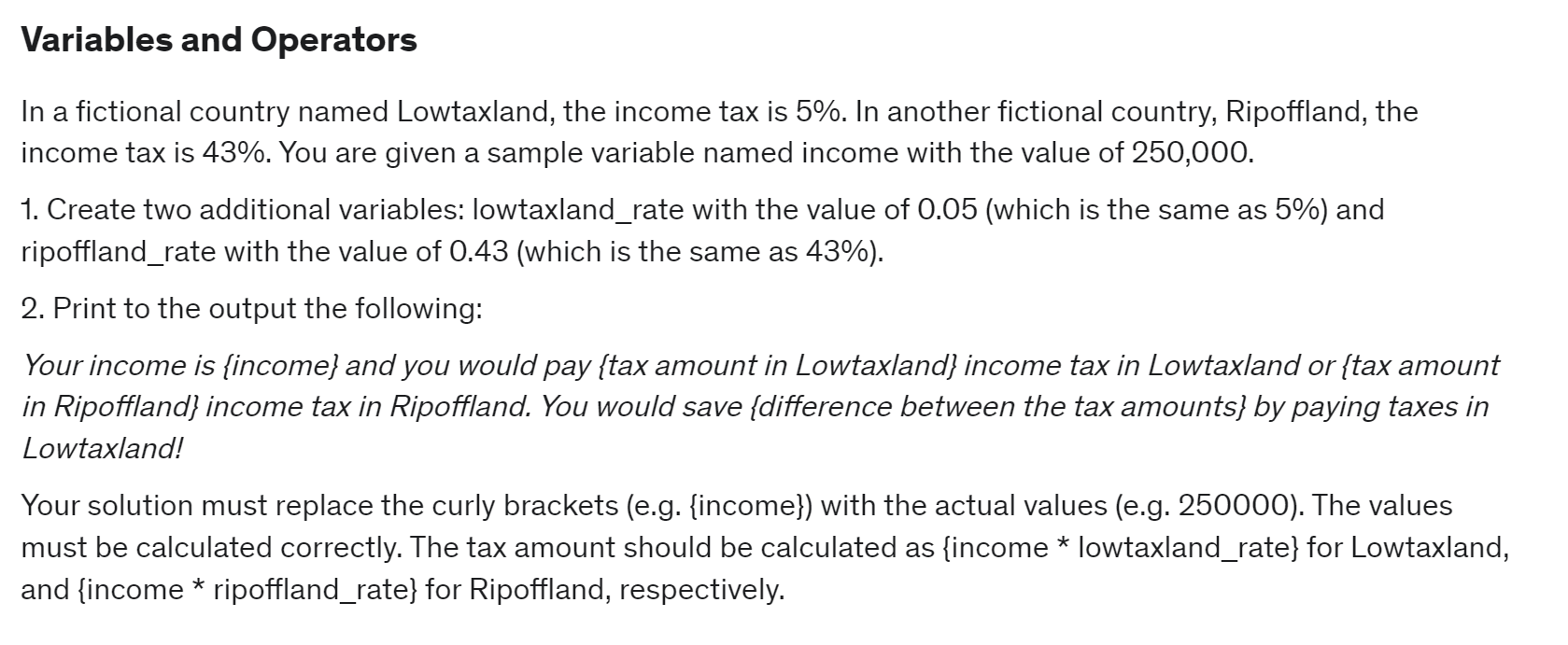
print(numero\_esadecimale) #1091

Console:



Console:

**ESERCIZIO**

****

**Soluzione esercizio**

income = 250\_000

lowtaxland\_rate = 0.05

ripoffland\_rate = 0.43

print('Your income is ' + str(income) + ' and you would pay ' + str(lowtaxland\_rate\*income) + ' income tax in Lowtaxland or ' + str(ripoffland\_rate \* income) + ' tax in Ripoffland. You would save ' +  str(-lowtaxland\_rate\*income + ripoffland\_rate \* income)   + ' by paying taxes in Lowtaxland!')

**PYTHON RIASSEGNAZIONE VARIABILI**

Per ora abbiamo visto int, float e string. Prendiamo ad esempio una variabile di tipo int, senza ledere la generalità.

Codice Python:

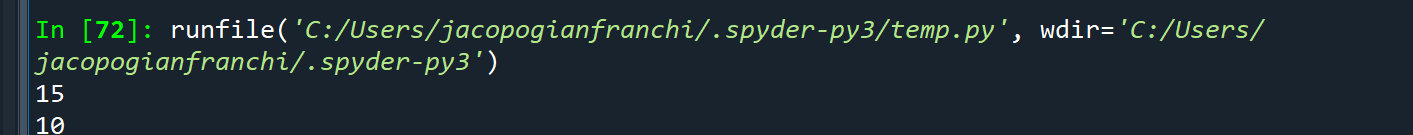
x = 15

print(x)

x = 10

print(x)

Console:



Abbiamo quindi riassegnato, o sovrascritto, il valore alla variabile x.

**MODO CONTRATTO PER RIASSEGNARE LE VARIABILI**

Vediamo ora dei modi per riassegnare le variabili incrementandole, moltiplicandole, dividendole o sottraendole per loro stesse.

Codice Python:

print("Forma Estesa\n")

x = 15

print(x)  # 15

x = x + 4

print(x) # 19

x = x - 10

print(x) # 9

x = x / 4

print(x) # 2.25

x = x \* 6

print(x) # 13.5

print()

print("Contratta\n")

x = 15

print(x)  # 15

x += 4

print(x) # 18ì9

x -= 10

print(x) # 9

x /= 4

print(x) # 2.25

x \*= 6

print(x) # 13.5

Console:

Immagine che contiene testo

Descrizione generata automaticamente

**CONCATENAZIONE STRINGHE**

Focalizziamoci sui numeri e sulle stringhe.

Abbiamo fino ad ora visto che:

3 + 5 = 8 (int)

‘ciao’ + ‘sono’ + ‘io’ = ciaosonoio (str)

Vediamo ora alcuni casi particolari.

**Concatenazione stringa con int**

Codice Python:

x = 'ciao'

y = 10

print(x + y)

Console:

Immagine che contiene testo

Descrizione generata automaticamente

Non è quindi possibile concatenare direttamente una stringa con un int. Per farlo, dobbiamo effettuare un cast, ossia trasformare l’int in string.

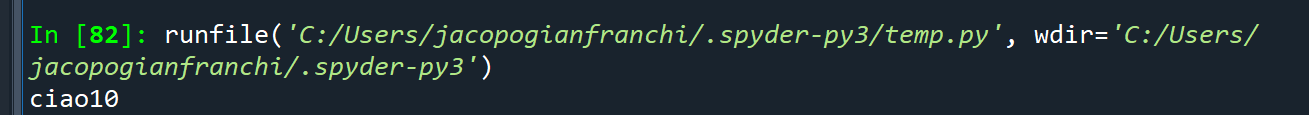
Codice Python:

x = 'ciao'

y = 10

print(x + str(y))

Console:



**Concatenazione int con string**

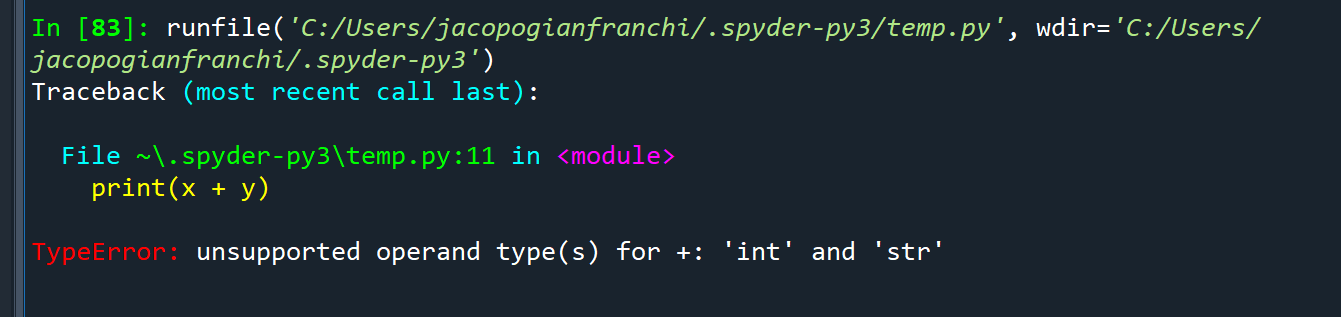
Codice Python:

x = 10

y = 'ciao'

print(x + y)

Console:



Anche in questo caso, si può effettuare il casting:

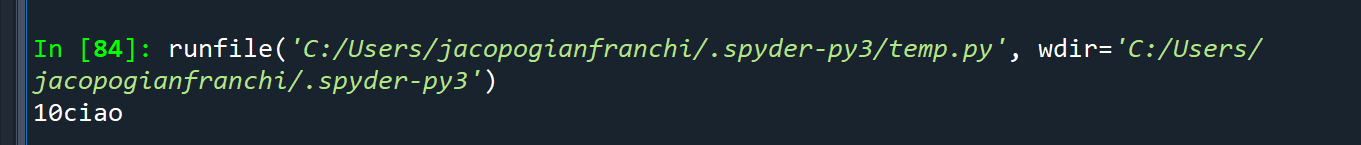
Codice Python:

x = 10

y = 'ciao'

print(str(x) + y)

Console:



C’è un modo per evitare il casting. E’ sufficiente utilizzare la funzione print()!

Codice Python:

x = 10

y = 'ciao'

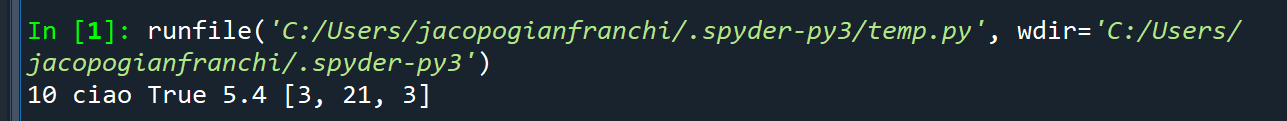
z = True

h = 5.4

j = [3,21,3]

print(x,y,z,h,j)

Console:



**FUNZIONE INPUT**

La funzione input permette di immagazzinare una stringa all’interno di una variabile.

La funzione input sembra non funzionare con spider, per questo utilizziamo jupiter notebook.

Esempio con Jupiter Notebook:

Immagine che contiene testo

Descrizione generata automaticamente

Ovviamente c’è stata la concatenazione, perché qualsiasi cosa venga inserita dall’utente all’interno della funzione input viene inserita nella variabile corrispondente sottoforma di stringa.

Possiamo quindi castare la stringa in int



**ESERCIZIO**

**Immagine che contiene testo

Descrizione generata automaticamente**