Laboratorio di programmazione

Python

A.A. 2020-2021

Lezione 4

Formattazione delle stringhe

Oltre a calcolare quantità e svolgere operazioni, spesso e volentieri vogliamo visualizzare quanto abbiamo calcolato e stamparlo a video o su di un file (vedremo come a tempo debito...)

Ad esempio, assumiamo di avere le seguenti variabili:

```
nome = 'Giangeppo'
anni = 42
altezza = 1.75
```

E di voler stampare a video la seguente frase:

```
"mi chiamo Giangeppo, ho 42 anni e sono alto 1.75. Tra 145 anni avrò 187 anni"
```

Dobbiamo costruire la stringa tenendo conto del **tipo** dei diversi oggetti che stiamo concatenando. In pratica, dobbiamo ragionare sulla **formattazione** della stringa che vogliamo visualizzare

Primo modo: operatori e logica

Prendiamo le tre variabili:

Primo modo: operatori e logica

Prendiamo le tre variabili:

In [2]:
```
nome = 'Giangeppo'
anni = 42
altezza = 1.75
```

Primo modo: operatori e logica

Prendiamo le tre variabili:

In [2]:
```
nome = 'Giangeppo'
anni = 42
altezza = 1.75
```

E proviamo a costruire la stringa di output con quanto abbiamo visto finora: operatori e casting di variabili

Primo modo: operatori e logica

Prendiamo le tre variabili:

In [2]:
```
nome = 'Giangeppo'
anni = 42
altezza = 1.75
```

E proviamo a costruire la stringa di output con quanto abbiamo visto finora: operatori e casting di variabili

In [3]:
```
out = "Mi chiamo " + nome + ", ho " + str(anni) + ' anni e sono alto ' + str(altezza) + \
'. Tra 145 anni avrò ' + str (anni+145) +' anni'
print(out)
```

```
Mi chiamo Giangeppo, ho 42 anni e sono alto 1.75. Tra 145 anni avrò 187 anni
```

Primo modo: operatori e logica

Prendiamo le tre variabili:

In [2]:
```
nome = 'Giangeppo'
anni = 42
altezza = 1.75
```

E proviamo a costruire la stringa di output con quanto abbiamo visto finora: operatori e casting di variabili

In [3]:
```
out = "Mi chiamo " + nome + ", ho " + str(anni) + ' anni e sono alto ' + str(altezza) + \
'. Tra 145 anni avrò ' + str (anni+145) +' anni'
print(out)
```

```
Mi chiamo Giangeppo, ho 42 anni e sono alto 1.75. Tra 145 anni avrò 187 anni
```

Potrei anche "semplificare" il tutto passando diversi argomenti alla funzione `print()`, ma impazzirei a gestire gli spazi:

Primo modo: operatori e logica

Prendiamo le tre variabili:

In [2]:
```
nome = 'Giangeppo'
anni = 42
altezza = 1.75
```

E proviamo a costruire la stringa di output con quanto abbiamo visto finora: operatori e casting di variabili

In [3]:
```
out = "Mi chiamo " + nome + ", ho " + str(anni) + ' anni e sono alto ' + str(altezza) + \
'. Tra 145 anni avrò ' + str (anni+145) +' anni'
print(out)
```

```
Mi chiamo Giangeppo, ho 42 anni e sono alto 1.75. Tra 145 anni avrò 187 anni
```

Potrei anche "semplificare" il tutto passando diversi argomenti alla funzione `print()`, ma impazzirei a gestire gli spazi:

In [4]:
```
print("Mi chiamo ", nome, ", ho ", anni, ' anni e sono alto ', altezza, '. Tra 145 anni avrò ', anni
```

```
Mi chiamo  Giangeppo , ho  42  anni e sono alto  1.75 . Tra 145 anni avrò  187  a
nni
```

Secondo modo: formattazione *old style*

Python ci consente di formattare le stringhe tramite l'utilizzo dell'operatore `%`.

Si tratta di una formattazaione di tipo posizionale:

Secondo modo: formattazione *old style*

Python ci consente di formattare le stringhe tramite l'utilizzo dell'operatore `%`.

Si tratta di una formattazaione di tipo posizionale:

In [5]:
```
out = "Mi chiamo %s, ho %i anni e sono alto %f. Tra 145 anni avrò %d anni"%(nome, anni, altezza, ann
print(out)
```

```
Mi chiamo Giangeppo, ho 42 anni e sono alto 1.750000. Tra 145 anni avrò 187 anni
```

Secondo modo: formattazione *old style*

Python ci consente di formattare le stringhe tramite l'utilizzo dell'operatore `%`.

Si tratta di una formattazaione di tipo posizionale:

In [5]:
```
out = "Mi chiamo %s, ho %i anni e sono alto %f. Tra 145 anni avrò %d anni"%(nome, anni, altezza, ann
print(out)
```

```
Mi chiamo Giangeppo, ho 42 anni e sono alto 1.750000. Tra 145 anni avrò 187 anni
```

Questo tipo di formattazione ci semplifica la vita, ma ci obbliga a ricordare una serie di *conversion types* per formattare correttamente la stringa. I più utilizzati sono:

- `%s`: `str`
- `%i` o `%d`: `int`
- `%f`: `float`
- `%e`: `float` in notazione esponenziale

Trovate la lista completa qui: https://docs.python.org/3/library/stdtypes.html#old-string-formatting

Inoltre dobbiamo fare attenzione a come formattiamo i `float`.

Dobbiamo familiarizzare con la sintassi che ci porta da questo:

Inoltre dobbiamo fare attenzione a come formattiamo i `float`.

Dobbiamo familiarizzare con la sintassi che ci porta da questo:

In [6]:
```
stringa_float = "%f"%(altezza)
print(stringa_float)
```

```
1.750000
```

Inoltre dobbiamo fare attenzione a come formattiamo i `float`.

Dobbiamo familiarizzare con la sintassi che ci porta da questo:

In [6]:
```
stringa_float = "%f"%(altezza)
print(stringa_float)
```

```
1.750000
```

A questo:

Inoltre dobbiamo fare attenzione a come formattiamo i `float`.

Dobbiamo familiarizzare con la sintassi che ci porta da questo:

In [6]:
```
stringa_float = "%f"%(altezza)
print(stringa_float)
```

```
1.750000
```

A questo:

In [7]:
```
stringa_float = "%3.2f"%(altezza)
print(stringa_float)
```

```
1.75
```

Terzo metodo: formattazione *new style*

Python 3 introdice un nuovo metodo di formattazione tramite un metodo del tipo `str`: il metodo `format()`.

La formattazione *new style* non prevede l'uso dell'operatore `%` e rende la sintassi più regolare.

Possiamo applicare una formattazione di tipo pozionale, come nel caso precedente:

Terzo metodo: formattazione *new style*

Python 3 introdice un nuovo metodo di formattazione tramite un metodo del tipo `str`: il metodo `format()`.

La formattazione *new style* non prevede l'uso dell'operatore `%` e rende la sintassi più regolare.

Possiamo applicare una formattazione di tipo pozionale, come nel caso precedente:

In [6]:
```
out = "Mi chiamo {}, ho {} anni e sono alto {}. Tra 145 anni avrò {} anni".format(nome, anni, altezz
print(out)
```

```
Mi chiamo Giangeppo, ho 42 anni e sono alto 1.75. Tra 145 anni avrò 187 anni
```

Terzo metodo: formattazione *new style*

Python 3 introdice un nuovo metodo di formattazione tramite un metodo del tipo `str`: il metodo `format()`.

La formattazione *new style* non prevede l'uso dell'operatore `%` e rende la sintassi più regolare.

Possiamo applicare una formattazione di tipo pozionale, come nel caso precedente:

In [6]:
```
out = "Mi chiamo {}, ho {} anni e sono alto {}. Tra 145 anni avrò {} anni".format(nome, anni, altezz
print(out)
```

```
Mi chiamo Giangeppo, ho 42 anni e sono alto 1.75. Tra 145 anni avrò 187 anni
```

Oppure possiamo fare riferimento al nome delle variabili che abbiamo già assegnato. In questo modo possiamo cambiare l'ordine delle sostituzioni, senza cambiare gli argomenti che passiamo al metodo `format()`:

Terzo metodo: formattazione *new style*

Python 3 introdice un nuovo metodo di formattazione tramite un metodo del tipo `str`: il metodo `format()`.

La formattazione *new style* non prevede l'uso dell'operatore `%` e rende la sintassi più regolare.

Possiamo applicare una formattazione di tipo pozionale, come nel caso precedente:

In [6]:
```
out = "Mi chiamo {}, ho {} anni e sono alto {}. Tra 145 anni avrò {} anni".format(nome, anni, altezz
print(out)
```

```
Mi chiamo Giangeppo, ho 42 anni e sono alto 1.75. Tra 145 anni avrò 187 anni
```

Oppure possiamo fare riferimento al nome delle variabili che abbiamo già assegnato. In questo modo possiamo cambiare l'ordine delle sostituzioni, senza cambiare gli argomenti che passiamo al metodo `format()`:

In [7]:
```
out_1 = "Il signor {name} ha {age} anni".format(name = nome, age = anni)
print(out_1)
out_2 = "Il signore di {age} anni è {name}".format(name = nome, age = anni)
print(out_2)
```

```
Il signor Giangeppo ha 42 anni
Il signore di 42 anni è Giangeppo
```

Quarto metodo: *f-strings*

Da Python 3.6 in poi, abbiamo a disposizione un approccio nuovo al problema della formattazione delle stringhe: le *formatted string literals*, per gli amici *f-strings*.

Il nostro esempio diventa:

Quarto metodo: *f-strings*

Da Python 3.6 in poi, abbiamo a disposizione un approccio nuovo al problema della formattazione delle stringhe: le *formatted string literals*, per gli amici *f-strings*.

Il nostro esempio diventa:

In [8]:
```
out = f"Mi chiamo {nome}, ho {anni} anni e sono alto {altezza}. Tra 145 anni avrò {anni+145} anni"
print(out)
```

```
Mi chiamo Giangeppo, ho 42 anni e sono alto 1.75. Tra 145 anni avrò 187 anni
```

Quarto metodo: *f-strings*

Da Python 3.6 in poi, abbiamo a disposizione un approccio nuovo al problema della formattazione delle stringhe: le *formatted string literals*, per gli amici *f-strings*.

Il nostro esempio diventa:

In [8]:
```
out = f"Mi chiamo {nome}, ho {anni} anni e sono alto {altezza}. Tra 145 anni avrò {anni+145} anni"
print(out)
```

```
Mi chiamo Giangeppo, ho 42 anni e sono alto 1.75. Tra 145 anni avrò 187 anni
```

Il grosso vantaggio di questo metodo è che possiamo utilizzare espressioni Python all'interno della stringa stessa:

Quarto metodo: *f-strings*

Da Python 3.6 in poi, abbiamo a disposizione un approccio nuovo al problema della formattazione delle stringhe: le *formatted string literals*, per gli amici *f-strings*.

Il nostro esempio diventa:

In [8]:
```
out = f"Mi chiamo {nome}, ho {anni} anni e sono alto {altezza}. Tra 145 anni avrò {anni+145} anni"
print(out)
```

```
Mi chiamo Giangeppo, ho 42 anni e sono alto 1.75. Tra 145 anni avrò 187 anni
```

Il grosso vantaggio di questo metodo è che possiamo utilizzare espressioni Python all'interno della stringa stessa:

In [10]:
```
a = 5
b = 6
frase = f"La somma di {a} e {b} è {a+b}, non {2*(a+b)}"
print(frase)
```

```
La somma di 5 e 6 è 11, non 22
```

Esercizi

Prendiamo il testo seguente:

```
'''Nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura,
ché la diritta via era smarrita.

Ahi quanto a dir qual era è cosa dura,
esta selva selvaggia e aspra e forte,
che nel pensier rinova la paura!'''
```

e lavoriamoci un po' su.

Esercizi

1. Contate le righe (di effettivo testo) che compongono l'estratto
2. Contate le parole che compongono l'estratto
3. Scrivete al contrario il terzo verso
4. Create un dizionario che mappi ogni carattere (chiave) con la sua occorrenza nel testo (valore)
5. Create un dizionario come il precedente per le sole lettere (no caratteri speciali), ignorando maiuscole e minuscole
6. Individuate l'indice del verso che contiene la stringa `"selva"`
7. Individuate la posizione della stringa `"selva"` all'interno del verso
8. Inserite, dopo la stringa `"selva"` la stringa:

   ```
   "(la selva è un bosco)"
   ```

Esercizi

Prendiamo la seguente rubrica (costruita con dizionari annidati):

```
{
  'Paolino Paperino': {'giorno': 9,
                       'mese': 'agosto',
                       'anno': 1934,
                       'età': 87,
                       'sesso': 'M',
                       'mail': 'paolino.paperin0@disney.org'},
 'Ron Weasley': {'giorno': 1, 'mese': 'maggio', 'anno': 1980, 'età': 41, 'sesso': 'M',
'mail': 'ron_weasley80@hogwards.uk'},
 'Ramona Flowers': {'giorno': 18, 'mese': 'ottobre', 'anno': 2004, 'età': 42, 'sesso':
'F', 'mail': 'ramona.fls@gmail.com'},
 'Madoka Ayukawa': {'giorno': 25, 'mese': 'luglio', 'anno': 1969, 'età': 52, 'sesso':
'F', 'mail': 'madoka_sax@asahi_net.jp'}
}
```

Esercizi

1. Utilizzare la rubrica costruita in precedenza per scrivere, ad ogni membro della rubrica, il seguente messaggio:

```
'''Car[o/a] [Nome],
sei nat[o/a] il [giorno] di [mese] del [anno] e quindi a breve compirai [età]
anni.
Ti manderemo gli auguri a [mail]'''
```

2. A partire dalla rubrica, costruire la lista delle età, ordinata in ordine crescente
3. Invertire la lista precedentemente costruita
4. Riorganizzare la rubrica in ordine crescente di età dei personaggi