Variabili e tipi di dato

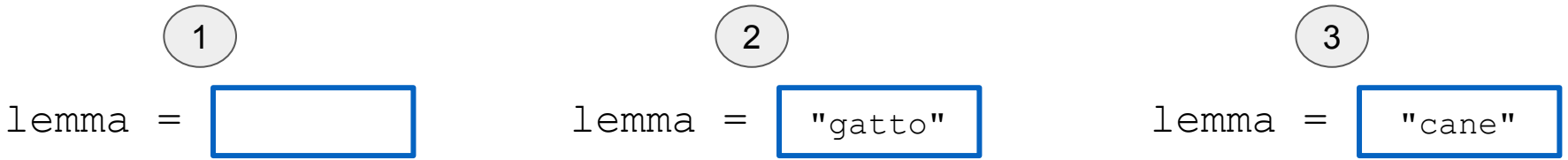
Rachele Sprugnoli

rachele.sprugnoli@unipr.it

UNIVERSITÀ
DI PARMA

Variabili

- Servono a memorizzare dei valori da riutilizzare
- ASSEGNAZIONE = dare un nome a una variabile
  - Una variabile è come un contenitore: ha un'etichetta all'esterno (il nome) e contiene alcune informazioni (il valore)
- 3 fasi:
  1) la variabile viene creata alla prima assegnazione (INIZIALIZZAZIONE)
  2) la variabile viene inizializzata dando un valore
  3) una nuova assegnazione sovrascrive il valore precedente



Assegnazione

- Enunciato di assegnazione
  - `anni = 20`
  - `lemma = "gatto"`

→ ATTENZIONE: il simbolo = è operatore di assegnazione, da non confondere con l'uguaglianza matematica che in Python si rende con il simbolo ==

In un'assegnazione si possono svolgere calcoli aritmetici:

  - `anni = anni + 1`
  - `anni = anni * 2`

Nominare una variabile

- Devono iniziare con una lettera o un underscore _
  - `lemma` `_lemma`
- Usare SOLO lettere, numeri o underscore per i caratteri successivi
  - `lemma1` `lemma_1`
- Usare maiuscole per distinguere le parole: *camel case*
  - `lemmaIt` `lemmaEn`
- Maiuscole e minuscole sono diverse: *case sensitive*
  - `lemma` versus `Lemma`
- Non si possono usare le parole riservate, come *if* o *for*
  - https://docs.python.org/3/reference/lexical_analysis.html#keywords

N.B. Usare nomi descrittivi: `lemmaIt` versus `cosa`

Esercizio

- Quali nomi sono validi e perché?
  a. `1_PartOfSpeech`
  b. `PartOfSpeech_1`
  c. `Part+Speech`
  d. `part_of_speech`
  e. `Part-Of-Speech`
  f. `Part of Speech`
  g. `PartOfSpeech`

Costanti

- Variabili il cui valore non deve essere modificato dopo l'assegnazione iniziale

  → ATTENZIONE: Python NON segnala errore se si assegna un nuovo valore a una costante

- Convenzione: il nome delle variabili va TUTTO IN MAIUSCOLO

```
VOLUME_BOTTIGLIA = 2

bottiglie = 6

volumeTotale = bottiglie * VOLUME_BOTTIGLIA
```

Tipi di dati

- Ciascun valore è di uno specifico tipo
- Il tipo di dato determina come il valore viene rappresentato e memorizzato nel computer ma anche le operazioni che si possono eseguire sul valore
- Esempi:
  - intero, es. 6: `int`
  - in virgola mobile (decimale), es. 6.5: `float` (attenzione al punto!)
  - stringa, es. “Hello”: `string`
  - booleano, true/false: `bool`
  - lista, dizionario e tupla

Esercizio

- Indicare il tipo di dato di ciascun esempio:

  a. -6
  b. 0.5
  c. 0,5
  d. 3 ½
  e. 100.000

- Per capire di che tipo si tratta: `type(dato)`
  a. `type(6)` → int
  b. `type("Ciao!")` → string

Operatori aritmetici

- NON si possono usare tra stringhe
- + (addizione), - (sottrazione), * (moltiplicazione), / (divisione), ** (elevamento a potenza)

`3+3` → 6

`3-3` → 0

`3*3` → 9

`3/3` → 1

`3**2` → 9

N.B. Cosa succede con: `3+"3"`?

Operatori relazionali

- > (maggiore), < (minore), >= (maggiore o uguale), <= (minore o uguale), **== (uguale), != (diverso) → questi ultimi si possono usare anche con le stringhe**

  `3 == 4` → falso

  `3 <= 4` → vero

  `4 >= 4` → vero

  `3 > 4` → falso

  `3 == 5 - 2` → vero

  `3 != 5 - 1` → vero

  `"ciao" != "arrivederci"` → vero

Operatori booleani

- Da George Boole, matematico e logico inglese del XIX sec.:
  - `and` (congiunzione): ritorna True se entrambi gli operandi sono veri, altrimenti False
  - `or` (disgiunzione): ritorna True se almeno uno degli operandi è vero, altrimenti False
  - `not` (negazione): ritorna False se l'operando è vero, True se l'operando è falso

```
21 > 1 and 3 < 5 → True
2 < 1 and "asd" == "asd" → False
"io" == "io" or "io" == "robot" → True
4 == 5 or 5 == 6 → False
not "io" == "robot" → True
not 3 == 3 → False
```

Operazioni con numeri

- Oltre a usare gli operatori aritmetici visti precedentemente, ci sono delle apposite **funzioni** come:
  - `round(45.621)` → 46 → arrotondamento all'intero più vicino, se i due numeri interi su entrambi i lati sono equidistanti, viene scelto il numero pari:
    - `round(4.5)` → 4
  - `round(45.621,1)` → 45.6 → arrotondamento a una cifra decimale
  - `max(3,1,10,15)` → 15 → elemento con valore più alto
  - `min(3,1,10,15)` → 1 → elemento con valore più basso

Stringhe

- Sequenza di caratteri tra virgolette corrispondenti (semplici o doppie)
- Carattere di ***escape*** (barra rovesciata, *backslash* \)

  You're "Welcome" → `"You're \"Welcome\""`

  Ciao!
Come va? → `print("Ciao!\nCome va?")`

  - \n --> per andare a capo
  - \t --> tabulazione
  - \' --> apostrofo
  - \" --> doppie virgolette

Stringhe e caratteri

- Il numero di caratteri di una stringa è la sua **lunghezza**

  - funzione `len(stringa)`

    `len("Hello!")` → `6`

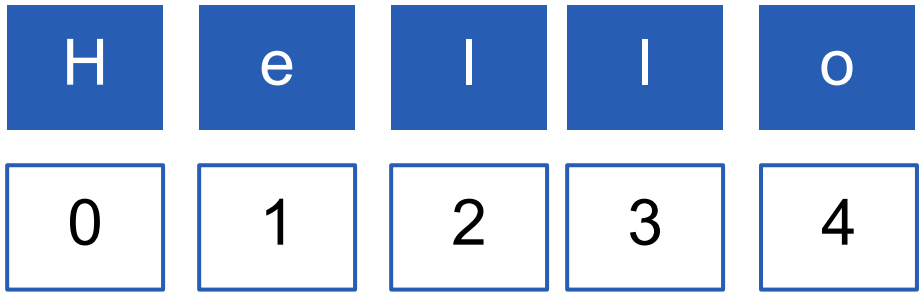
    `len("Hello, world!")` → `13`

N.B. La funzione `len()` si applica anche ad altri tipi di dato, ad es. alle liste

```
listA=[1,2,3]
len(listA) → 3
```

Stringhe e caratteri

- È possibile accedere ai singoli caratteri di una stringa riferendosi alla loro posizione al suo interno → INDICE, parte da 0




```
saluto = "Hello"
```

`first = saluto[0]` → H

`last = saluto[4]` → o

`first_second = saluto[0:2]` → He

Stringhe e caratteri: slicing

- Ottenere un intervallo di caratteri:

```
istituzione = "Università di Parma"
```

`istituzione[0:9]` --> Università

`istituzione[0:11]` --> Università d

`istituzione[0:-1]` --> Università di Parm

`istituzione[:1]` --> U

`istituzione[:]` --> Università di Parma

`istituzione[2:]` --> iversità di Parma

Funzioni di conversione

- Da valore numerico a stringa

  `anni = 43` → intero

  `str(43)` → diventa stringa (funziona anche per convertire un `float`)

- Da stringa a valore numerico

  `anni = "43"`

  `int("43")` → diventa intero

  `litri = "2.5"`

  `float("2.5")` → diventa numero decimale

Conversioni e concatenazione

- Non è possibile concatenare stringhe e numeri insieme, quindi è necessario convertire per stampare

```
anno = 2016
evento = “Referendum”
print("Risultati del " + evento + " " + str(anno) +
":")
```

Conversioni: f-string

- Stringhe letterali formattate: consentono di includere il valore delle espressioni Python all'interno di una stringa anteponendo alla stringa f o F come prefisso e scrivendo le espressioni tra parentesi graffe
  - introdotto da Python 3.6

```
anno = 2016

evento = "Referendum"

print(f"Risultati del {evento} {anno}:")
```

Concatenazioni e ripetizioni

- Come visto nella slide 18, le stringhe si concatenano con il +
- Una stringa di qualsiasi lunghezza si può ripetere usando *
  - + e * si chiamano **operatori**

ESEMPIO:

```
trattino = "-" * 50
```

`print (trattino)` → ripete _ 50 volte

ESEMPIO:

```
messaggio = "Ciao…"
print(messaggio * 5)
```

→ il fattore moltiplicativo deve essere un numero intero (NO 5.5)

Metodi per stringhe

- `.upper()` → trasforma in maiuscolo una stringa
- `.lower()` → trasforma in minuscolo una stringa
- `.replace(old,new)` → sostituisce una stringa (o sottostringa) con un'altra
- `.split()` → suddivide una stringa in un elenco di sottostringhe utilizzando un separatore scelto (separatore di default è lo spazio)

nome_dato.nome_metodo(eventuali_parametri)

Metodi per stringhe

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

→ ATTENZIONE: `len()` è una funzione e non un metodo perché non è specifica delle stringhe: si può applicare, tra gli altri, anche a liste e dizionari

```
lista=[1,2,3,4]

len(lista) → 4
```

Metodi per stringhe

```
personaggio = "Gatto Silvestro"

personaggio.upper() → GATTO SILVESTRO

personaggio.lower() → gatto silvestro

personaggio.replace("Silvestro","con gli stivali") → Gatto
con gli stivali

personaggio.split() → ["Gatto", "Silvestro"]

personaggio.split("i") → ["Gatto S", "lvestro"]
```

Metodi per stringhe

Per stampare:

```
personaggio = "Gatto Silvestro"
print(personaggio.upper())
print(personaggio.lower())
print(personaggio.replace("Silvestro","con gli stivali"))
print(personaggio.split())
```

Esercizio 1

- Variabili e stampa di stringhe
  - Crea due variabili (nome ed età)
  - Stampa un’intera frase che includa le due variabili usando le funzioni di conversione
  - Stampa un’intera frase che includa le due variabili usando la f-string

Esercizio 2

- Maiuscolo
  - Trasforma in maiuscolo la frase che hai stampato nell'esercizio precedente
  - Stampa la stessa frase sostituendo il nome
  - Stampa in maiuscolo la frase con il nome sostituito

Esercizio 3

- Stampa iniziali
  - Crea un programma che stampi una coppia di iniziali di due nomi propri divise dalla “e” commerciale : esempio, T&T per Taylor e Travis

Esercizio 4

- Cicli e concatenazioni
  - Scrivi uno script che stampi tutti i numeri da 0 a 48
    - Se il numero è inferiore a 25, stampa "X è minore di 25", dove X è il numero
    - Se il numero è maggiore di 25, stampa "X è maggiore di 25", dove X è il numero

```
0 è minore di 25
1 è minore di 25
2 è minore di 25
```

Esercizio 5

- Operatori relazionali
  - Scegliere due numeri e stampare tutti i possibili confronti tra di essi (maggiore, minore, uguale…)

Soluzioni esercizi

- https://colab.research.google.com/drive/1COPkr0KtaRpFQxrnrsJ_G4h1VL48ZSa5?usp=sharing