

Es. 1 (Le strutture dati in Python) PDF LaTeX

April 30, 2024

1 LE STRUTTURE E TIPI DI DATI IN PYTHON

In questa esercitazione vengono mostrati le principali strutture e i tipi di dati in Python.

1.1 I TIPI DI DATI

I tipi di dati servono per definire dei valori che una variabile può assumere e per definire le operazioni che quella variabile può fare. I tipi di dati possono essere semplici oppure non semplici, cioè le sequenze.

1.1.1 I TIPI DI DATI SEMPLICI

I tipi di dati semplici sono i tipi di dati fondamentali per lo sviluppo di Python, come per qualsiasi linguaggio di programmazione.

I tipi di dati semplici sono: - Gli integer (int) - I float (float) - I boolean (bool) - I complex (complex) - I long (long) - I char (chr)

GLI INTEGER Gli integer (int) definiscono i valori di variabili corrispondenti a dei numeri interi (solo a numeri interi, non a numeri decimali). Con gli integer si possono eseguire operazioni aritmetiche di base come la somma, la differenza, il prodotto, la divisione, la potenza

```
[ ]: a = 10 # Valore integer
      print(a)
      print("Il tipo di dato infatti è:")
      print(type(a)) # attraverso il comando type è possibile sapere il tipo di dato
                      ↪ in Python
```

10

Il tipo di dato infatti è:

<class 'int'>

ESEMPIO: SOMMA Questo codice è un esempio di somma di due numeri interi (due variabili con valori integer)

```
[ ]: a = 10 # il primo valore è integer
      b = 8 # il secondo valore è integer
      c = a + b
      print(f"La somma di {a} e {b} è:")
```

```
print(c)
print("Il tipo di dato infatti è:")
print(type(c))
```

La somma di 10 e 8 è:

18

Il tipo di dato infatti è:

<class 'int'>

ESEMPIO: DIFFERENZA Questo codice è un esempio di differenza di due numeri interi (due variabili con valori integer)

```
[ ]: a = 10 # il primo valore è integer
      b = 8 # il secondo valore è integer
      c = a - b
      print(f"La differenza di {a} e {b} è:")
      print(c)
      print("Il tipo di dato infatti è:")
      print(type(c))
```

La differenza di 10 e 8 è:

2

Il tipo di dato infatti è:

<class 'int'>

ESEMPIO: DIVISIONE Questo codice è un esempio di divisione di due numeri interi (due variabili con valori integer)

```
[ ]: a = 10 # il primo valore è integer
      b = 2 # il secondo valore è integer
      c = a / b # Esegue la divisione normale (ritorna sempre un float)
      d = a // b # Esegue la divisione ritornando solo la parte intera (ritorna
      ↪ sempre un int), si usano le due barre anziché una barra sola
      print(f"La divisione normale di {a} e {b} è:")
      print(c)
      print("Il tipo di dato della divisione normale infatti è:")
      print(type(c))
      print(f"La divisione con solo la parte intera di {a} e {b} è:")
      print(d)
      print("Il tipo di dato della divisione con solo la parte intera infatti è:")
      print(type(d))
```

La divisione normale di 10 e 2 è:

5.0

Il tipo di dato della divisione normale infatti è:

<class 'float'>

La divisione con solo la parte intera di 10 e 2 è:

5

Il tipo di dato della divisione con solo la parte intera infatti è:
<class 'int'>

ESEMPIO: MOLTIPLICAZIONE Questo codice è un esempio di prodotto di due numeri interi (due variabili con valori integer)

```
[ ]: a = 8 # il primo valore è integer
      b = 2 # il secondo valore è integer
      c = a * b # Esegue la moltiplicazione (si usa sempre l'asterisco)

      print(f"La moltiplicazione di {a} e {b} è:")
      print(c)
      print("Il tipo di dato della moltiplicazione infatti è:")
      print(type(c))
```

La moltiplicazione di 8 e 2 è:
16
Il tipo di dato della moltiplicazione infatti è:
<class 'int'>

ESEMPIO: POTENZA Questo codice è un esempio di calcolo della potenza di un numero (variabile integer a) calcolato alla seconda (perchè è il valore della variabile integer b)

```
[ ]: a = 8 # il primo valore è integer
      b = 2 # il secondo valore è integer
      c = a ** b # Esegue la potenza (si usa sempre il doppio asterisco)

      print(f"La potenza di {a} e {b} è:")
      print(c)
      print("Il tipo di dato della potenza infatti è:")
      print(type(c))
```

La potenza di 8 e 2 è:
64
Il tipo di dato della potenza infatti è:
<class 'int'>

I FLOAT I float (float) definiscono i valori di variabili corrispondenti a dei numeri con la virgola. Con i float si possono eseguire operazioni aritmetiche di base come la somma, la differenza, il prodotto, la divisione, la potenza ma con i numeri decimali

```
[ ]: a = 12.3 # Valore float
      print(a)
      print("Il tipo di dato infatti è:")
      print(type(a))
```

12.3
Il tipo di dato infatti è:
<class 'float'>

ESEMPIO: SOMMA Questo codice è un esempio di somma di due numeri reali (due variabili con valori float)

```
[ ]: a = 10.2 # il primo valore è un float
      b = 8.2 # il secondo valore è un float
      d = 3 # il terzo valore è un integer

      c = a + b # Si sommano i due float
      e = a + d # Si sommano un float con un int, il risultato da SEMPRE un float
      print(f"La somma di {a} e {b} è:")
      print(c)
      print("Il tipo di dato infatti è:")
      print(type(c))
      print(f"La somma di {a} e {d} è:")
      print(e)
      print("Il tipo di dato infatti è:")
      print(type(e))
```

La somma di 10.2 e 8.2 è:

18.4

Il tipo di dato infatti è:

<class 'float'>

La somma di 10.2 e 3 è:

13.2

Il tipo di dato infatti è:

<class 'float'>

ESEMPIO: DIFFERENZA Questo codice è un esempio di differenza di due numeri reali (due variabili con valori float)

```
[ ]: a = 10.2 # il primo valore è un float
      b = 8.2 # il secondo valore è un float
      d = 3 # il terzo valore è un integer

      c = a - b # Si sottraggono due float
      e = a - d # Si sottraggono un float con un int, il risultato da SEMPRE un float
      print(f"La differenza di {a} e {b} è:")
      print(c)
      print("Il tipo di dato infatti è:")
      print(type(c))
      print(f"La differenza di {a} e {d} è:")
      print(e)
      print("Il tipo di dato infatti è:")
      print(type(e))
```

La differenza di 10.2 e 8.2 è:

2.0

Il tipo di dato infatti è:

<class 'float'>

La differenza di 10.2 e 3 è:
7.1999999999999999
Il tipo di dato infatti è:
<class 'float'>

ESEMPIO: DIVISIONE Questo codice è un esempio di divisione di due numeri reali (due variabili con valori float)

```
[ ]: a = 10.2 # il primo valore è un float
     b = 8.2 # il secondo valore è un float
     d = 3 # il terzo valore è un integer

     c = a / b # Si esegue la divisione normale
     c2 = a // b
     e = a / d # Si esegue comunque la divisione normale, con la divisione con i
        ↪ float la doppia barra non vale, ma approssima per difetto il risultato
     e2 = a // d

     print(f"La divisione con una barra di {a} e {b} è:")
     print(c)
     print("Il tipo di dato infatti è")
     print(type(c))
     print(f"La divisione con due barre di {a} e {b} è:")
     print(c2)
     print("Il tipo di dato infatti è")
     print(type(c2))
     print(f"La divisione con una barra di {a} e {d} è:")
     print(e)
     print("Il tipo di dato infatti è:")
     print(type(e))
     print(f"La divisione con due barre di {a} e {d} è:")
     print(e2)
     print("Il tipo di dato infatti è:")
     print(type(e2))
```

La divisione con una barra di 10.2 e 8.2 è:
1.2439024390243902
Il tipo di dato infatti è
<class 'float'>
La divisione con due barre di 10.2 e 8.2 è:
1.0
Il tipo di dato infatti è
<class 'float'>
La divisione con una barra di 10.2 e 3 è:
3.4
Il tipo di dato infatti è:
<class 'float'>
La divisione con due barre di 10.2 e 3 è:

3.0

Il tipo di dato infatti è:

<class 'float'>

ESEMPIO: MOLTIPLICAZIONE Questo codice è un esempio di prodotto di due numeri reali (due variabili con valori float)

```
[ ]: a = 10.2 # il primo valore è un float
      b = 8.2 # il secondo valore è un float
      d = 5 # il terzo valore è un integer

      c = a * b # Si esegue la moltiplicazione
      e = a * d # Si esegue la moltiplicazione (non cambia se c'è un int, il
        ↳ risultato sarà sempre un float)

      print(f"La moltiplicazione di {a} e {b} è:")
      print(c)
      print("Il tipo di dato infatti è:")
      print(type(c))
      print(f"La moltiplicazione di {a} e {d} è:")
      print(e)
      print("Il tipo di dato infatti è:")
      print(type(e))
```

La moltiplicazione di 10.2 e 8.2 è:

83.63999999999999

Il tipo di dato infatti è:

<class 'float'>

La moltiplicazione di 10.2 e 5 è:

51.0

Il tipo di dato infatti è:

<class 'float'>

ESEMPIO: POTENZA Questo codice è un esempio di calcolo della potenza di un numero (variabile float a) calcolato alla seconda (perchè è il valore della variabile float b)

```
[ ]: a = 9.1 # il primo valore è un float
      b = 2.3 # il secondo valore è un float
      d = 3 # il terzo valore è un integer

      c = a**b # Si esegue la potenza
      e = a**d # Si esegue la potenza (non cambia se c'è un int, il risultato sarà
        ↳ sempre un float)

      print(f"La potenza di {a} e {b} è:")
      print(c)
      print("Il tipo di dato infatti è:")
      print(type(c))
```

```
print(f"La differenza di {a} e {d} è:")
print(e)
print("Il tipo di dato infatti è:")
print(type(e))
```

La potenza di 9.1 e 2.3 è:

160.61836559414905

Il tipo di dato infatti è:

<class 'float'>

La differenza di 9.1 e 3 è:

753.5709999999999

Il tipo di dato infatti è:

<class 'float'>

I BOOLEANI I booleani (bool) definiscono i valori di variabili corrispondenti a dei booleani, cioè a True o False (Vero o Falso). Con i booleani possiamo eseguire operazioni logiche di base come l'and, l'or e il not.

```
[ ]: a = True # Valore booleano positivo
      b = False # Valore booleano negativo
      print(a)
      print("Il tipo di dato infatti è:")
      print(type(a))
      print(b)
      print("Il tipo di dato infatti è:")
      print(type(b))
```

True

Il tipo di dato infatti è:

<class 'bool'>

False

Il tipo di dato infatti è:

<class 'bool'>

L'AND LOGICO (E LOGICA) COSTRUIENDO LA TABELLA DELLA VERITÀ

```
[ ]: a = True
      b = False

      # Quando c'è False prevale sempre quest'ultimo come output finale

      print(f"L'AND di {a} e {b} è:")
      c = a and b
      print(c)
      print(f"L'AND di {a} e {a} è:")
      c = a and a
      print(c)
      print(f"L'AND di {b} e {a} è:")
```

```

c = a and b
print(c)
print(f"L'AND di {b} e {b} è:")
c = b and b
print(c)

```

L'AND di True e False è:
 False
 L'AND di True e True è:
 True
 L'AND di False e True è:
 False
 L'AND di False e False è:
 False

L'OR LOGICO (O LOGICO) COSTRUIENDO LA TABELLA DELLA VERITÀ

```

[ ]: a = True
      b = False

print(f"L'OR di {a} e {b} è:")
c = a or b
print(c)
print(f"L'OR di {a} e {a} è:")
c = a or a
print(c)
print(f"L'OR di {b} e {a} è:")
c = b or a
print(c)
print(f"L'OR di {b} e {b} è:")
c = b or b
print(c)

```

L'OR di True e False è:
 True
 L'OR di True e True è:
 True
 L'OR di False e True è:
 True
 L'OR di False e False è:
 False

LA NEGAZIONE (NOT)

```

[ ]: a = True
      b = False

print(f"Il not di {a} è:")
c = not a

```



```
print(c)
print(f"Il not di {b} è:")
c = not b
print(c)
```

Il not di True è:
False
Il not di False è:
True

I COMPLESSI (COMPLEX) I complessi (complex) definiscono i valori di variabili corrispondenti a dei numeri complessi. Con i complessi possiamo eseguire operazioni aritmetiche di base per numeri complessi come la somma, la differenza, il prodotto e la divisione.

```
[ ]: # Il numero a sinistra del più (3) si chiama parte reale
      # Il numero a destra del più (1j*3) si chiama parte immaginaria

      # I numeri complessi vengono definiti qui sotto
a = 3 + 1j*3 # Per creare una variabile complessa bisogna sommare un numero_
      ↪ (parte reale) ad una parte immaginaria (valore di j)

print("Il risultato di 3 + 1j*3 è:")
print(a)
print(f"Il tipo di dato infatti è: ")
print(type(a))
```

Il risultato di 3 + 1j*3 è:
(3+3j)
Il tipo di dato infatti è:
<class 'complex'>

ESEMPIO: SOMMA

```
[ ]: a = 3 + 1j*3 # prima variabile di tipo complex
      b = 2 + 1j*2 # seconda variabile di tipo complex
      c = 3.0 # terza variabile di tipo float

      d = a + b # Si sommano due complessi (e il risultato è sempre un complesso)
      e = a + c # Si sommano un float con un complesso (e il risultato è sempre un_
      ↪ complesso)

print(f"La somma di {a} e {b} è:")
print(d)
print("Il tipo di dato infatti è:")
print(type(d))
print(f"La somma di {a} e {c} è:")
print(e)
print("Il tipo di dato infatti è:")
print(type(e))
```

La somma di $(3+3j)$ e $(2+2j)$ è:

$(5+5j)$

Il tipo di dato infatti è:

`<class 'complex'>`

La somma di $(3+3j)$ e 3.0 è:

$(6+3j)$

Il tipo di dato infatti è:

`<class 'complex'>`

ESEMPIO: DIFFERENZA

```
[ ]: a = 3 - 1j*3 # prima variabile di tipo complex
      b = 2 - 1j*2 # seconda variabile di tipo complex
      c = 3.0 # terza variabile di tipo float

      d = a - b # Si sottraggono due complessi (e il risultato è sempre un complesso)
      e = a - c # Si sottraggono un float con un complesso (e il risultato è solo la
      ↪ parte immaginaria)

      print(f"La differenza di {a} e {b} è:")
      print(d)
      print("Il tipo di dato infatti è:")
      print(type(d))
      print(f"La differenza di {a} e {c} è:")
      print(e)
      print("Il tipo di dato infatti è:")
      print(type(e))
```

La differenza di $(3-3j)$ e $(2-2j)$ è:

$(1-1j)$

Il tipo di dato infatti è:

`<class 'complex'>`

La differenza di $(3-3j)$ e 3.0 è:

$-3j$

Il tipo di dato infatti è:

`<class 'complex'>`

ESEMPIO: DIVISIONE

```
[ ]: a = 3 + 1j*3 # prima variabile complessa
      b = 2 + 1j*2 # seconda variabile complessa
      c = 3.0 # terza variabile float

      d = a / b # Si dividono due complessi (e il risultato è sempre un complesso)
      e = a / c # Si dividono un float con un complesso

      print(f"La divisione di {a} e {b} è:")
      print(d)
      print("Il tipo di dato infatti è:")
      print(type(d))
```

```
print(f"La divisione di {a} e {c} è:")
print(e)
print("Il tipo di dato infatti è:")
print(type(e))
```

La divisione di $(3+3j)$ e $(2+2j)$ è:

$(1.5+0j)$

Il tipo di dato infatti è:

<class 'complex'>

La divisione di $(3+3j)$ e 3.0 è:

$(1+1j)$

Il tipo di dato infatti è:

<class 'complex'>

ESEMPIO: MOLTIPLICAZIONE

```
[ ]: a = 3 - 1j*3 # prima variabile di tipo complex
      b = 2 - 1j*2 # seconda variabile di tipo complex
      c = 3.0 # terza variabile di tipo float

      d = a * b # Si moltiplicano due complessi (e il risultato è solo la parte
      ↪immaginaria)
      e = a * c # Si moltiplicano un float con un complesso (e il risultato è sempre
      ↪un complesso)

print(f"La moltiplicazione di {a} e {b} è:")
print(d)
print("Il tipo di dato infatti è:")
print(type(d))
print(f"La moltiplicazione di {a} e {c} è:")
print(e)
print("Il tipo di dato infatti è:")
print(type(e))
```

La moltiplicazione di $(3-3j)$ e $(2-2j)$ è:

$-12j$

Il tipo di dato infatti è:

<class 'complex'>

La moltiplicazione di $(3-3j)$ e 3.0 è:

$(9-9j)$

Il tipo di dato infatti è:

<class 'complex'>

ESEMPIO: POTENZA

```
[ ]: a = 3 - 1j*3 # prima variabile di tipo complex
      b = 2 - 1j*2 # seconda variabile di tipo complex
      c = 3.0 # terza variabile di tipo float
```

```

d = a ** b # Si moltiplicano due complessi (e il risultato è solo la parte
↳immaginaria)
e = a ** c # Si moltiplicano un float con un complesso (e il risultato è sempre
↳un complesso)

print(f"La moltiplicazione di {a} e {b} è:")
print(d)
print("Il tipo di dato infatti è:")
print(type(d))
print(f"La moltiplicazione di {a} e {c} è:")
print(e)
print("Il tipo di dato infatti è:")
print(type(e))

```

La moltiplicazione di (3-3j) e (2-2j) è:
 (-0.9301698201642012+3.6243749286074007j)
 Il tipo di dato infatti è:
 <class 'complex'>
 La moltiplicazione di (3-3j) e 3.0 è:
 (-54-54j)
 Il tipo di dato infatti è:
 <class 'complex'>

I LONG In passato, in Python 2 (la scorsa edizione di Python), c'era una distinzione tra due tipi di numeri interi: "int" e "long". Il tipo "int" aveva un limite preciso e il tipo "long" veniva utilizzato per rappresentare numeri più grandi rispetto a quel limite. Tuttavia, con l'avvento di Python 3 (l'edizione di Python che si usa comunemente oggi), questa distinzione è stata eliminata. In Python 3, il tipo "int" è stato quindi potenziato per gestire anche numeri interi di dimensioni arbitrarie. Quindi ormai qualsiasi numero intero di una qualsiasi lunghezza numerica è considerato come integer (int). Nell'immagine qui sotto viene usato un simulatore di Python 2 online per vedere com'era l'output una volta, dove infatti il tipo di dato veniva riconosciuto come long

```

[ ]: from IPython.display import Image

# Specifica il percorso della tua immagine locale
percorso_screenshot = r'C:\Users\matte\OneDrive - Scuola Paritaria S. Freud
↳SRL\Desktop\FREUD\2°D\QUADERNI E ALTRO\INFORMATICA\PYTHON\ESERCIZI IN
↳CLASSE\Es. 1 (Le strutture dati in Python)\Screenshot Python 2.png'

# Visualizza l'immagine nel notebook
Image(filename=percorso_screenshot)

```



Oggi infatti lo stesso identico codice ha un output ben diverso...

```
[ ]: numero_lungo = 12**33
      print(numero_lungo)
      print("Il tipo di dato infatti e':")
      print(type(numero_lungo))
```

```
410186270246002225336426103593500672
Il tipo di dato infatti e':
<class 'int'>
```

I CHAR

1.2 I TIPI DI DATI NON SEMPLICI (LE SEQUENZE)

Le sequenze sono i tipi di dati più complessi, cioè più strutturati rispetto a quelli semplici, e contengono tipi di dati semplici mantenendo un ordine (ad eccezione della stringa, poichè definisce una parola o frase con una sequenza di caratteri).

Le sequenze si dividono in mutabili e immutabili. La differenza nel fatto che le mutabili sono sequenze che possono cambiare dopo essere state create, invece le sequenze immutabili sono sequenze che possono cambiare dopo essere state create.

Le sequenze mutabili sono: - Le liste (list) - I dizionari (dict) - Gli insiemi (set) - I file (file)

1.2.1 LE SEQUENZE MUTABILI

LE LISTE

```
[ ]: lista_numerica = [7, 12, 18, 33, 5, 44, 88, 99, 112]
      print(lista_numerica)
```

```
[7, 12, 18, 33, 5, 44, 88, 99, 112]
```

```
[ ]: lista_con_caratteri = ["a","c","q","k","ciao","Matteo","Vittorio","Mattia","come va?"]
      print(lista_con_caratteri)
```

```
['a', 'c', 'q', 'k', 'ciao', 'Matteo', 'Vittorio', 'Mattia', 'come va?']
```

```
[ ]: lista_mista = ["12","3","5", 12, 18, 33, 5,
                  ↪44,"ciao","Matteo","Vittorio","Mattia","come va?"]
      print(lista_mista)
```

```
['12', '3', '5', 12, 18, 33, 5, 44, 'ciao', 'Matteo', 'Vittorio', 'Mattia',
'come va?']
```

```
[ ]: lista_di_una_lista = ["12",["3"],["5", [12, 18], 33], 5, 44]]
      print(lista_di_una_lista)
```

```
['12', ['3', ['5', [12, 18], 33], 5, 44]]
```

```
[ ]: lista_numerica.append(1012)
print(lista_numerica)
```

[7, 12, 18, 33, 5, 44, 88, 99, 112, 1012]

```
[ ]: lista_con_caratteri.append("Python")
print(lista_con_caratteri)
```

['a', 'c', 'q', 'k', 'ciao', 'Matteo', 'Vittorio', 'Mattia', 'come va?', 'Python']

```
[ ]: lista_numerica.remove(112) # Rimuove il singolo numero all'interno della lista
    ↪ (solo se è veramente presente)
print(lista_numerica)
```

[7, 12, 18, 33, 5, 44, 88, 99, 1012]

```
[ ]: lista_numerica.pop(3) # Rimuove il singolo numero alla posizione indicata (in
    ↪ questo caso alla posizione n°3, da ricordare il fatto che su Python la
    ↪ numerazione parte da 0) all'interno della lista (solo se è veramente
    ↪ presente)
print(lista_numerica)
```

[7, 12, 18, 5, 44, 88, 99, 1012]

```
[ ]: lista_sommata=lista_numerica + lista_mista
print(lista_sommata)
```

[7, 12, 18, 5, 44, 88, 99, 1012, '12', '3', '5', 12, 18, 33, 5, 44, 'ciao', 'Matteo', 'Vittorio', 'Mattia', 'come va?']

I DIZIONARI

```
[ ]: dizionario_cibo_calorie = {
    "pizza": 285, # La virgola è fondamentale perchè serve ad indicare che c'è
    ↪ un'altra key
    "hamburger": 250,
    "insalata": 100,
    "pasta": 158,
    "pollo arrosto": 195,
    "riso": 130,
    "sushi": 374,
    "lasagna": 336,
    "fragole": 32,
    "gelato": 207,
    "panino": 320,
    "patatine fritte": 365,
    "uva": 69,
    "cioccolato": 546,
    "yogurt": 61,
```

```

    "spaghetti": 131,
    "pesce al vapore": 95,
    "muffin": 444,
    "cereali": 363,
    "torta al cioccolato": 237
}

```

```

[ ]: dizionario_cibo_calorie["frittata"] = "non importa le calorie, è buona"
dizionario_cibo_calorie["ciambellone"] = "non importa le calorie, è buono, non_
    ↳mi toccate il ciambellone della nonna"
dizionario_cibo_calorie # Per avere un output migliore lo stampo direttamente_
    ↳indicando il nome della variabile

```

```

[ ]: {'pizza': 285,
      'hamburger': 250,
      'insalata': 100,
      'pasta': 158,
      'pollo arrosto': 195,
      'riso': 130,
      'sushi': 374,
      'lasagna': 336,
      'fragole': 32,
      'gelato': 207,
      'panino': 320,
      'patatine fritte': 365,
      'uva': 69,
      'cioccolato': 546,
      'yogurt': 61,
      'spaghetti': 131,
      'pesce al vapore': 95,
      'muffin': 444,
      'cereali': 363,
      'torta al cioccolato': 237,
      'frittata': 'non importa le calorie, è buona',
      'ciambellone': 'non importa le calorie, è buono, non mi toccate il ciambellone
della nonna'}

```

```

[ ]: print(dizionario_cibo_calorie) # Come output questo a mio parere è più_
    ↳difficile da capire, meglio quello sopra

```

```

{'pizza': 285, 'hamburger': 250, 'insalata': 100, 'pasta': 158, 'pollo arrosto':
195, 'riso': 130, 'sushi': 374, 'lasagna': 336, 'fragole': 32, 'gelato': 207,
'panino': 320, 'patatine fritte': 365, 'uva': 69, 'cioccolato': 546, 'yogurt':
61, 'spaghetti': 131, 'pesce al vapore': 95, 'muffin': 444, 'cereali': 363,
'torta al cioccolato': 237, 'frittata': 'non importa le calorie, è buona',
'ciambellone': 'non importa le calorie, è buono, non mi toccate il ciambellone
della nonna'}

```

```
[ ]: print(dizionario_cibo_calorie['muffin'])
print("Mhhhh... buono il muffin!")
```

444

Mhhhh... buono il muffin!

```
[ ]: if "pesce al vapore" in dizionario_cibo_calorie:
    print(f"La key è presente nel dizionario, quindi puoi mangiare quel cibo")
else:
    print("No non c'è quella key, adesso te tocca sta' digiuno")
```

La key è presente nel dizionario, quindi puoi mangiare quel cibo

```
[ ]: if "zuppa di miso" in dizionario_cibo_calorie:
    print(f"La key è presente nel dizionario, quindi puoi mangiare quel cibo")
else:
    print("No non c'è quella key, adesso te tocca sta' digiuno")
```

No non c'è quella key, adesso te tocca sta' digiuno

GLI INSIEMI

```
[ ]:
```

I FILE

```
[ ]:
```

1.2.2 LE SEQUENZE IMMUTABILI

Le sequenze immutabili sono: - Le stringhe (str) - Le tuple (tuple) - Gli insiemi immutabili (frozenset)

LE STRINGHE

```
[ ]: stringa = "Ciao mi chiamo Matteo"
```

```
[ ]: stringa1 = "Ciao mi chiamo Matteo"
stringa2 = "e ho 15 anni"
stringa_concatenata = stringa1 + " " + stringa2 # Ho inserito anche uno spazio
↳ come stringa per essere precisi con l'output
print(stringa_concatenata)
```

Ciao mi chiamo Matteo e ho 15 anni

```
[ ]: sottostringa = stringa_concatenata[1:12] # Ottiene i caratteri dalla posizione
↳ 1 alla 12
print(sottostringa)
```

iao mi chia

LE TUPLE


```
[ ]: tupla = (1, 2, 3)
      print(tupla)
```

(1, 2, 3)

```
[ ]: tupla = (1, 2, 3)
      primo_elemento = tupla[0]
      secondo_elemento = tupla[1]
      print(tupla)
```

(1, 2, 3)

```
[ ]: tupla = (1, 2, 3, 4, 5)
      lunghezza = len(tupla)
      print(tupla)
```

(1, 2, 3, 4, 5)

GLI INSIEMI IMMUTABILI

```
[ ]:
```