

A stylized pink lightbulb icon with radiating lines, positioned to the left of the word 'ragazze'.

# ragazze DIGITALI

IDEE PER UN FUTURO SMART

## Cosa faremo oggi

Ti trovi in una terra piena di draghi.  
Di fronte a te ci sono due grotte  
In una grotta si trova un drago simpatico e socievole che  
condividerà con te il suo tesoro.  
Nell'altra grotta c'è invece un drago affamato e vorace.  
Dentro quale grotta vuoi entrare? (1 o 2)

1

Entri nella caverna 1  
E' buia e spaventosa  
Un enorme drago compare all'improvviso davanti a te! Apre  
le sue fauci e... ti inghiottisce in un batter d'occhio!  
Vuoi giocare ancora?  
No

# Outline

- 1 Funzioni
- 2 Variabili locali e variabili globali
- 3 Operatori booleani
- 4 Costrutto while
- 5 Il regno dei draghi
- 6 Variabili Costanti
- 7 Liste
- 8 L'impiccato

- Abbiamo già visto e usato alcune funzioni, ad esempio: `print('ciao')`, `input()`, `randint(1, 20)`, ecc..
- Possiamo anche crearne noi!
- Con le funzioni possiamo incapsulare del codice e riutilizzarlo più volte nel nostro programma.

## Una funzione

- Deve essere dichiarata **prima** di essere chiamata
- Può avere dei parametri
- Restituisce un risultato

# Funzioni

## Come si dichiara una funzione

- Utilizziamo la keyword **def**
- seguita dal **nome** della funzione
- Seguita dai **parametri** racchiusi tra parentesi (o le sole parentesi aperta e chiusa nel caso non ci siano parametri)
- Seguiti da :

def keyword  
↓  
def displayIntro():  
↑                    ↑  
Function name      Colon

```
def displayIntro():  
    print('''Ti trovi  
           in una terra  
           piena di  
           draghi...''')  
print()
```

! notiamo il `print(''' ''')`, stringa multilinea

## Alcuni consigli..

- Ciò che abbiamo detto per i nomi delle variabili vale anche per i nomi delle funzioni
- Meglio se le nostre funzioni sono corte, con poco codice
- Con 1 funzione facciamo 1 cosa soltanto!
- Evitiamo di usare più di 1 o 2 parametri in una funzione  
Se abbiamo bisogno di più parametri dividiamo quello che vogliamo fare in più funzioni

## Funzioni con parametri

- Possiamo "dare in pasto" ad una funzione dei dati, delle informazioni, che chiamiamo **parametri**
- Queste informazioni ci serviranno per elaborare il risultato della funzione

```
def isPositive(number):  
    if number > 0 :  
        return True  
    else:  
        return False
```

# 0 ancora meglio..  
def isPositive(number):  
 return number > 0

```
# Usiamo la funzione, ad esempio, in questo modo  
print(isPositive(5))      # True  
print(isPositive(-5))    # False
```

## Funzioni con parametri

- Possiamo "dare in pasto" ad una funzione dei dati, delle informazioni, che chiamiamo **parametri**
- Queste informazioni ci serviranno per elaborare il risultato della funzione
- Il risultato verrà restituito utilizzando la keyword **return**

```
def sum(firstNumber, secondNumber):  
    return firstNumber + secondNumber
```

```
# Usiamo la funzione, ad esempio, in questo modo  
result = sum(5, 4)      # result avra' valore 9  
result = sum(-10, 5)   # result avra' valore -5
```



# Outline

- 1 Funzioni
- 2 Variabili locali e variabili globali
- 3 Operatori booleani
- 4 Costrutto while
- 5 Il regno dei draghi
- 6 Variabili Costanti
- 7 Liste
- 8 L'impiccato

# Variabili locali e variabili globali

## Variabili locali

Chiamiamo **Variabile locale** qualunque variabile dichiarata all'interno di una funzione, questa esiste solo all'interno della funzione stessa. Le variabili locali vengono "dimenticate" dopo che la funzione ha raggiunto il return (e quindi ha finito le sue elaborazioni).

```
def welcomePerson():  
    person = 'Chiara'      # Variabile locale  
    print('Benvenuta ' + person)  
# Anche se non c'è il return, da qui in poi la  
# variabile non ha più effetto
```

```
welcomePerson()  
person = 'Sofia'  
print(person) # Sofia  
welcomePerson() # Benvenuta Chiara
```

# Variabili globali

## Variabili globali

Nell'esempio precedente `person = 'Sofia'` e' una variabile globale, ovvero una variabile che ha effetto in tutto il nostro programma. Modifichiamo leggermente l'esempio:

```
def welcomePerson():  
    print('Benvenuta ' + person)
```

```
person = 'Chiara'          # Variabile globale  
welcomePerson() # Benvenuta Chiara  
person = 'Sofia'  
welcomePerson() # Benvenuta Sofia
```

# Outline

- 1 Funzioni
- 2 Variabili locali e variabili globali
- 3 Operatori booleani**
- 4 Costrutto while
- 5 Il regno dei draghi
- 6 Variabili Costanti
- 7 Liste
- 8 L'impiccato

Quale frase è complessivamente vera e quale falsa?

- I gatti hanno i baffi E i cani hanno la coda
- I gatti hanno i baffi E i cani hanno le ali
- I gatti abbaiano E i cani hanno le ali

## Operatore and (e)

			Risultato
True	<b>and</b>	True	True
True	<b>and</b>	False	False
False	<b>and</b>	True	False
False	<b>and</b>	False	False

# Operatore or (o)

Quale frase è complessivamente vera e quale falsa?

- I gatti hanno i baffi O i cani hanno la coda
- I gatti hanno i baffi O i cani hanno le ali
- I gatti abbaiano O i cani hanno le ali

```
city = 'Cesena'  
result = 10 < 20 or city == 'Cesena'  
# result avra' valore?  
result = 10 > 20 or city == 'Cesena'  
# result avra' valore?
```

## Operatore or (o)

			Risultato
True	<b>or</b>	True	True
True	<b>or</b>	False	True
False	<b>or</b>	True	True
False	<b>or</b>	False	False



# Operatore not

```
city = 'Cesena'
result = not (10 < 20 or city == 'Cesena')
# result avra' valore?
result = not (10 > 20 or city == 'Cesena')
# result avra' valore?
```

# Operatore not

	Risultato
not True	False
not False	True

# Outline

- 1 Funzioni
- 2 Variabili locali e variabili globali
- 3 Operatori booleani
- 4 Costrutto while**
- 5 Il regno dei draghi
- 6 Variabili Costanti
- 7 Liste
- 8 L'impiccato

# Costrutto while

## while

- Costrutto simile al for, che abbiamo già visto
- Il while ripete il ciclo finché una determinata condizione rimane vera
- Qual è la differenza con il for?

```
while month == 'giugno' and year == '2019' :  
    codice del ciclo while
```

! come nel for, nell'if e nelle funzioni occhio all'indentazione del codice del ciclo!

# Outline

- 1 Funzioni
- 2 Variabili locali e variabili globali
- 3 Operatori booleani
- 4 Costrutto while
- 5 Il regno dei draghi**
- 6 Variabili Costanti
- 7 Liste
- 8 L'impiccato

## Serviranno le funzioni di alcuni moduli

Cosa dobbiamo fare per poter richiamare funzioni di un modulo/libreria?

`random`

Utilizzeremo la funzione `randint()` che abbiamo già visto

`time`

Utilizzeremo la funzione `sleep(forSeconds)`

# Il regno dei draghi - Codice

```
import random  
import time
```

## Ritorniamo al gioco di oggi

Ti trovi in una terra piena di draghi.  
Di fronte a te ci sono due grotte  
In una grotta si trova un drago simpatico e socievole che  
condividerà con te il suo tesoro.  
Nell'altra grotta c'è invece un drago affamato e vorace.  
Dentro quale grotta vuoi entrare? (1 o 2)

1

Entri dentro la caverna 1  
E' buia e spaventosa  
Un enorme drago compare all'improvviso davanti a te! Apre  
le sue fauci e... ti inghiottisce in un batter d'occhio!  
Vuoi giocare ancora?  
No



# Il regno dei draghi - Parte 1

E' il vostro turno!

Scrivi una funzione, chiamandola come preferisci, che stampi a video il testo introduttivo del gioco:

'Ti trovi in una terra piena di draghi.

Di fronte a te ci sono due grotte

In una grotta si trova un drago simpatico e socievole che condividerà con te il suo tesoro.

Nell'altra grotta c'è invece un drago affamato e vorace.

Dentro quale grotta vuoi entrare? (1 o 2)'

## Il regno dei draghi - Parte 2

E' il vostro turno!

Scrivi una funzione, chiamandola come preferisci, che chieda al giocatore di inserire il numero della grotta nella quale vuole entrare (1 o 2) e restituisca/ritorni il numero stesso della grotta scelta

Aiuto:

```
answer = ''  
while answer != '1' or answer != '2' :  
    codice del ciclo while  
    "ripensa al primo giorno quando chiedevamo  
    all'utente il nome e stampavamo 'ciao  
    nome'"
```

## Il regno dei draghi - Parte 3

E' il vostro turno!

Scrivi una funzione, chiamandola come preferisci, che

- Prenda come parametro il numero della grotta scelto dal giocatore
- Stampi a video
  - 'Ti avvicini alla grotta numero ...' (numero della grotta inserito dal giocatore)
  - 'E' buia e spaventosa..'
  - 'Un enorme drago compare all'improvviso davanti a te! Apre le sue fauci e...'
- Dopo la stampa a video di ogni frase fai attendere 2 secondi al giocatore per aggiungere suspense utilizzando la funzione sleep del modulo time

```
time.sleep(2) # Interrompe l'esecuzione del  
              programma per i secondi specificati come  
              parametro
```

## Il regno dei draghi - Parte 3

### E' il vostro turno!

Nella funzione della pagina precedente, dopo le stampe e i `time.sleep(2)`

- Aggiungi una variabile `friendlyCave` dandole un valore random tra 1 e 2 (utilizzando la funzione `randint` del modulo `random` come mostrato nel codice qui sotto)
- Se la variabile `friendlyCave` ha lo stesso valore del numero della grotta scelta dal giocatore allora stampa  
'Hai vinto il tesoro!'  
altrimenti stampa  
'Il drago ti mangia in un boccone!'

```
friendlyCave = random.randint(1, 2)
```

## Il regno dei draghi - Parte 4

### E' il vostro turno!

Dopo aver inizializzato una variabile chiamata 'playAgain' con valore 'si', utilizza un ciclo while che si ripeta finche' il valore di playAgain e' la stringa 'si' e al suo interno

- Richiami la prima funzione che avete creato per stampare il testo introduttivo del gioco
- Crei una variabile locale con il valore della grotta scelta dal giocatore (richiamando la seconda funzione che avete creato)
- Controlli se il valore della grotta scelta sia lo stesso della friendlyCave (richiamando terza funzione che avete creato)
- Infine stampi a video la stringa 'Vuoi giocare ancora?' e riassegni/modifichi il valore della variabile playAgain con la nuova scelta del giocatore (usando la funzione input())

# Outline

- 1 Funzioni
- 2 Variabili locali e variabili globali
- 3 Operatori booleani
- 4 Costrutto while
- 5 Il regno dei draghi
- 6 Variabili Costanti**
- 7 Liste
- 8 L'impiccato

# Variabili Costanti

## Variabili costanti

Capitera' di usare variabili il cui valore rimane costante nel tempo e che non verranno modificate dopo la loro prima dichiarazione. Secondo la convenzione il nome di queste variabili va scritto tutto in maiuscolo e, se composto da piu parole, queste sono separate da `_`(underscore).

```
MAX_SIZE = 10
MIN_SIZE = 1
LENGHT = 10
POSITIVE_ANSWER = 'Si' #
NEGATIVE_ANSWER = 'No' # Tra poco vedremo come
                       gestire in modo piu' elegante i valori si/no
```

# Outline

- 1 Funzioni
- 2 Variabili locali e variabili globali
- 3 Operatori booleani
- 4 Costrutto while
- 5 Il regno dei draghi
- 6 Variabili Costanti
- 7 Liste**
- 8 L'impiccato



## Liste

Le liste sono delle variabili che, invece che contenere dei singoli valori ne contengono molteplici.

```
teachers = ['Chiara', 'Enrico', 'Sofia']  
answers = ['Si', 'No']  
vocals = ['a', 'e', 'i', 'o', 'u']
```

# Liste

## Come leggiamo i valori di una lista

Pensiamo ad una lista come ad un mobile con dei cassettei numerati da 0 (primo cassetto) a lunghezza della lista -1 (ultimo cassetto).

! Attenzione, stiamo partendo da 0 !

```
vocals = ['a', 'e', 'i', 'o', 'u']
```

Per aprire i "cassetti" e leggere i singoli valori della lista quindi faremo

```
vocals[0] # Avra' valore 'a'  
vocals[1] # Avra' valore 'e'  
vocals[2] # ...  
vocals[3]  
vocals[4]
```

# Liste

## Come scriviamo i valori di una lista

Similmente a come li leggiamo e a come assegnamo un valore ad una variabile normale.

```
answers = ['Si', 'No']  
  
answers[0] = 'Yes' # Nella posizione 0 della  
                    lista scrivo 'Yes', sostituendolo al 'Si'
```

## append()

La funzione append() aggiunge valori ad una lista già definita

```
answers = ['Si', 'No']  
  
answers.append('Forse') # Ora answers avrà  
                        valori ['Si', 'No', 'Forse']
```

# Liste

## Lista vuota

Adesso che conosciamo `append()` possiamo anche crearci delle lista vuote e all'occorrenza aggiungere valori

```
answers = []  
answers.append('Si')  
answers.append('No')
```

## reverse()

La funzione `reverse()` inverte l'ordine degli elementi di una lista, modificando la lista stessa!

! Si, la lista e' ordinata !

```
vocals = ['a', 'e', 'i', 'o', 'u']  
vocals.reverse() # ['u', 'o', 'i', 'e', 'a']
```

# Liste

## Lista piena

Utilizzando la funzione `remove()` possiamo rimuovere dei valori

```
days = [10, 11, 12, 13, 14, 15]
days.remove(12)
print(days) # [10, 11, 13, 14, 15]
```

## reverse()

La funzione `reverse()` inverte l'ordine degli elementi di una lista, modificando la lista stessa!

! Sì, la lista è ordinata !

```
vocals = ['a', 'e', 'i', 'o', 'u']
vocals.reverse() # ['u', 'o', 'i', 'e', 'a']
```

## split()

Un'altra funzione utile è `split()`. Data una frase, un insieme di parole separate da spazi, `split()` restituisce una lista fatta dalle parole della frase stessa

```
sentence = 'Oggi a Cesena nevica e fa un gran  
caldo'  
sentence.split()  
# ['Oggi', 'a', 'Cesena', 'nevica', 'e', 'fa',  
  'un', 'gran', 'caldo']  
splittedSentence = sentence.split()  
splittedSentence[3] # Quale valore avra'?
```

Cosa succede se...

provo a leggere in una posizione maggiore della lunghezza della lista?

```
vocals = ['a', 'e', 'i', 'o', 'u']  
print(vocals[999])  
print(vocals[10])
```

## Cosa succede se...

provo a leggere o scrivere in una posizione maggiore della lunghezza della lista?

```
vocals = ['a', 'e', 'i', 'o', 'u']  
print(vocals[999])  
'vocals[9999]  
IndexError: list index out of range'  
  
vocals[10] = 'abcdefghi...'  
'IndexError: list assignment index out of range'
```



## range()

La funzione range restituisce una lista di numeri interi compresi nell'intervallo specificato come parametro

```
range(5) # [0, 1, 2, 3, 4]
# Se non specifico il minimo parto da 0
range(0, 10) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
range(5, 10) # [5, 6, 7, 8, 9] ! Il 10 non c'e'
```

`list()`

La funzione `list` prende come parametro un valore e restituisce una lista fatta degli elementi che compongono il parametro

```
list('Ciao') # ['C', 'i', 'a', 'o']
```

## Operatore in

L'operatore **in** ci dice se un determinato elemento sia contenuto in una lista o meno.

```
vocals = ['a', 'e', 'i', 'o', 'u']  
'e' in vocals # True  
'g' in vocals # False
```

# Outline

- 1 Funzioni
- 2 Variabili locali e variabili globali
- 3 Operatori booleani
- 4 Costrutto while
- 5 Il regno dei draghi
- 6 Variabili Costanti
- 7 Liste
- 8 L'impiccato

# L'impiccato - Parte 0

Per realizzare il gioco dell'impiccato dovrete completare il codice che trovate a questo link seguendo le indicazioni riportate nelle pagine successive

Download

# L'impiccato - Parte 1

## E' il vostro turno!

- Importa il modulo random, come visto in precedenza
- Crea una variabile globale chiamata words (nello specifico una stringa) che contenga le parole che il giocatore dovrà indovinare separate da spazi. Trasforma quindi la stringa in una lista che abbia le parole della variabile creata come elementi [aiuto: utilizza la funzione split()]
- Crea una variabile globale chiamata missedLetters (una stringa vuota nello specifico) con la quale memorizzeremo le lettere sbagliate inserite dall'utente
- Crea una variabile globale chiamata correctLetters (una stringa vuota nello specifico) on la quale memorizzeremo le lettere sbagliate inserite dall'utente
- Crea una variabile globale chiamata secretWord il cui valore e' il risultato della funzione che ritorna una parola casuale dalla lista di parole

### E' il vostro turno!

- Crea una funzione chiamata `getRandomWord(wordList)` che, data come parametro una lista di parole (la quale in precedenza abbiamo chiamato `words`), restituisca un suo elemento random (ad esempio se ho la lista `['Ragazze', 'Digitali', 'Cesena', 'Giugno']` mi deve restituire un elemento random di questa lista).

## L'impiccato - Parte 3

### E' il vostro turno!

Crea una funzione, dandole il nome che preferisci, che ci servira' per mostrare a video il 'campo di gioco' In particolare la funzione dovra':

- Prendere come parametri la stringa contenente le lettere sbagliate inserite dal giocatore, la stringa contenente le lettere corrette inserite dal giocatore e la parola da indovinare
- Stampare a video l'immagine dell'impiccato corrispondente al numero di errori commessi, presa dalla lista `HANGMAN_PICS` creata in precedenza,  
Ad esempio: se il giocatore ha commesso 0 errori stampera' l'immagine alla posizione 0 della lista `HANGMAN_PICS`, se ha commesso 1 errore stampera' l'immagine alla posizione 1 e cosi' via..
- Stampare a video tutte le lettere sbagliate inserite dal giocatore

Vedi esempio alla pagina successiva



# L'impiccato - Parte 3

```
+---+
0   |
/|\ |
/ \ |
    ===
```

Lettere sbagliate: a b c d f h

## L'impiccato - Parte 4

E' il vostro turno!

All'interno della funzione creata nella Parte 3 aggiungi

- una lista inizialmente vuota, dandole il nome che preferisci. Questa sarà una variabile locale della funzione nella quale scriveremo o le lettere inserite correttamente dal giocatore oppure dei '\_'
- un ciclo for che, per un numero di volte pari alla lunghezza della parola segreta da indovinare, aggiunga alla lista creata nel punto precedente o la lettera della parola segreta (se presente tra le lettere indovinate) oppure il carattere '\_' [aiuto: utilizza la funzione `append('a')`]

Esempio: Se la parola da indovinare è 'ciao' e ho inserito correttamente le lettere 'i' o 'o' il risultato dovrà essere ['\_', 'i', '\_', 'o']

- Stampa a video la lista appena creata e popolata dalle lettere corrette inserite oppure dai '\_'

E' il vostro turno!

Crea una funzione, chiamandola `playAgain()`, che chieda all'utente se vuole giocare di nuovo e restituisca, con `return`, la risposta (similmente a come abbiamo visto in precedenza con l'esempio per chiedere il nome)

Leggi la parte finale del codice fornito

- Che cosa fa il while?
- A che cosa ci serve la variabile play?
- Che differenza c'è con il while dentro alla funzione getGuess()?

Materiale rilasciato con licenza  
**Creative Commons - Attributions, Share-alike 4.0**

