RAGAZZE DIGITALI ER

Digital Girl Power

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

Funzioni

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

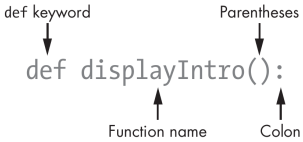
Una funzione

- 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 **:**




```
1 def displayIntro():
2     print('''Ti trovi in
          una terra piena
          di draghi...''')
3     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

```
1 def isPositive(number):
2     if number > 0:
3         return True
4     else:
5         return False
```

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

```
1 # Usiamo la funzione, ad esempio, in questo modo
2 print(isPositive(5))     # True
3 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**

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

```
1 # Usiamo la funzione, ad esempio, in questo modo
2 result = sum(5, 4)     # result = 9
3 result = sum(-10, 5)   # result = -5
```

Outline

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 o la fine (e quindi ha finito le sue elaborazioni).

```
1 def welcomePerson():
2     person = 'Chiara'      # Variablile locale
3     print('Benvenuta ' + person)
4 # Anche senza return, qui la variabile non esiste
```

```
1 welcomePerson()
2 person = 'Sofia'
3 print(person) # Sofia
4 welcomePerson() # Benvenuta Chiara
```

Variabili globali

Variabili globali

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

```
1 def welcomePerson():
2     print('Benvenuta ' + person)
3
4 person = 'Chiara' # Variablile globale
5 welcomePerson() # Benvenuta Chiara
6 person = 'Sofia'
7 welcomePerson() # Benvenuta Sofia
```

Outline

Operatore and (e)

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)

Quando è vero l' **and** tra due booleani?

Quando **entrambi** sono veri!

and		Risultato
True	True	True
True	False	False
False	True	False
False	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

```
1 city = 'Bologna'
2 result = 10 < 20 or city == 'Bologna'
3 # quanto vale result?
4 result = 10 > 20 or city == 'Bologna'
5 # quanto vale result?
```

Operatore or (o)

Quando è vero l' **or** tra due booleani?

Quando **almeno uno** è vero!

or		Risultato
True	True	True
True	False	True
False	True	True
False	False	False

Operatore not

```
1 city = 'Bologna'
2 result = not (10 < 20 or city == 'Bologna')
3 # quanto vale result?
4 result = not (10 > 20 or city == 'Bologna')
5 # quanto vale result?
```

Operatore not

Quando è vero il **not** di un booleano?

Quando quel booleano è falso (perchè il **not** "inverte" il valore di verità)!

not	Risultato
True	False
False	True

Outline

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?

```
1 while month == 'giugno' and year == '2022' :
2     # codice del ciclo while al posto di questi commenti
3     # svegliati
4     # vai a lezione
5     # ...
6     # vai a dormire (incrementa il giorno/mese/anno)
```

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

Outline

Serviranno le funzioni di alcuni moduli

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

Modulo `random`

Utilizzeremo la funzione `randint(min, max)` che abbiamo già visto

Modulo `time`

Utilizzeremo la funzione `sleep(seconds)`

Il regno dei draghi - Codice

```
1 import random
2 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
È 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

È 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

È 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 il numero stesso della grotta scelta.
Aiuto:

```
answer = ''
while answer != '1' and 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

È 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)
  - 'È 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`

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

Il regno dei draghi - Parte 3

È il vostro turno!

Nella funzione della pagina precendete, 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!'

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

Il regno dei draghi - Parte 4

È 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())

Soluzione

Outline

Variabili Costanti

Variabili costanti

Capiterà 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 più parole, queste sono separate da _(underscore).

```
1 MAX_SIZE = 10
2 MIN_SIZE = 1
3 LENGHT = 10
4 POSITIVE_ANSWER = 'Si'
5 NEGATIVE_ANSWER = 'No'
```

Outline

Liste

Liste

Le liste sono delle variabili che invece che contenere dei singoli valori ne contengono molteplici. Le liste sono delimitate dalle parentesi quadre e ogni elemento è separato da una virgola.

```
1 answers = ['Si', 'No']
2 vocals = ['a', 'e', 'i', 'o', 'u']
3 heterogeneous = [10, '42', True]
4 nested = [vocals, ['testo']]
```

Liste

Come leggiamo i valori di una lista

Pensiamo ad una lista come ad un mobile con dei cassetti numerati da 0 (primo cassetto) a lungezza della lista -1 (ultimo cassetto).
! Attenzione, stiamo partendo da 0 !

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

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

```
1 vocals[0] # Contiene 'a'
2 vocals[1] # Contiene 'e'
3 vocals[2] # ...
4 vocals[3]
5 vocals[4]
```

Liste

Come scriviamo i valori di una lista

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

```
1 answers = ['Si', 'No']
2 # In posizione 0 della lista scrivo 'Yes', al posto di 'Si'
3 answers[0] = 'Yes'
```

E per aggiungere nuovi valori?

La funzione `append()` aggiunge valori in fondo ad una lista già definita allungandola.

```
1 answers = ['Si', 'No']
2 answers.append('Forse')
3 # Ora answers contiene i valori ['Si', 'No', 'Forse']
```

Liste

Come possiamo invertire l'ordine di una lista?

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

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

Liste

Lista vuota

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

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

Lista piena

Utilizzando la funzione remove() possiamo rimuovere dei valori

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

Liste

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.

```
1 sentence = 'Oggi a Bologna nevica e fa un gran caldo'
2 sentence.split()
3 # ['Oggi', 'a', 'Bologna', 'nevica', 'e', 'fa', 'un', 'gran',
     'caldo']
4 splittedSentence = sentence.split()
5 splittedSentence[3] # Quanto vale?
```

Liste

Cosa succede se...

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

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

```
IndexError: list index out of range
```

```
1 vocals[10] = 'Ragazze Digitali'
```

```
IndexError: list assignment index out of range
```

Ricordate la funzione `range()`?

La funzione `range()` restituisce una lista di numeri interi compresi nell'intervallo specificato come parametro, escluso l'estremo superiore.

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

Liste

`list()`

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

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

Controllare se un elemento è presente nella lista

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

```
1 vocals = ['a', 'e', 'i', 'o', 'u']
2 'e' in vocals # ???
3 'g' in vocals # ???
```

Liste

Come possiamo sapere la lunghezza di una lista?

La funzione `len()` consente di ottenere la lunghezza di ciò che gli viene passato come parametro..

```
1 vocals = ['a', 'e', 'i', 'o', 'u']
2 len(vocals) # ???
```

Outline

L'impiccato - Parte 0

Per realizzare il gioco dell'impiccato dovrete completare il codice che trovate nel link qui sotto seguendo le indicazioni riportate nelle pagine successive.

Download

L'impiccato - Parte 1

È 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 è il risultato della funzione che ritorna una parola casuale dalla lista di parole

È 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 in **posizione** random.
  (ad esempio se ho la lista ['ragazze', 'digitali', 'bologna', 'giugno'] mi deve restituire un elemento random di questa lista).

L'impiccato - Parte 3

È il vostro turno!

Crea una funzione, dandole il nome che preferisci, che ci servirà per mostrare a video il "campo di gioco" In particolare la funzione dovrà:

- 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 stamperà l'immagine alla posizione 0 della lista `HANGMAN_PICS`, se ha commesso 1 errore stamperà l'immagine alla posizione 1 e cosi' via...
- Stampare a video tutte le lettere sbagliate inserite dal giocatore

Vedi esempio alla pagina sucessiva

L'impiccato - Parte 3

```
1
2
3
4
5
6
7 Lettere sbagliate: abcdfh
```

L'impiccato - Parte 4

È 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' il risultato dovrà essere ['_', 'i', '_', 'o']
- Stampa a video la lista appena creata e popolata dalle lettere corrette inserite oppure dai '_'

L'impiccato - Parte 5

È 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)

L'impiccato - Parte 6

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()`?

Soluzione