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

Funzioni

- 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.

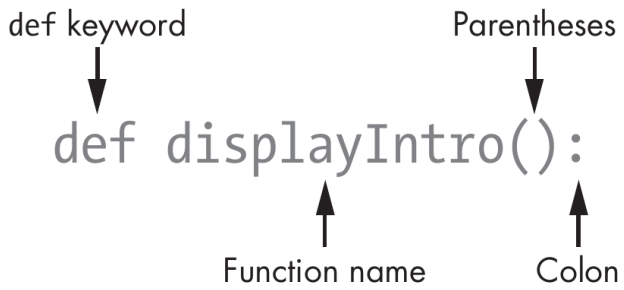
Una funzione

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

! notiamo il print(”’ ”’), stringa multilinea

Funzioni

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

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

```
                                    # O ancora meglio..
def isPositive(number):             def isPositive(number):
    if number > 0 :                     return number > 0
        return True
    else:
        return False
```

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

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
- 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

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'      # Variablile locale
    print('Benvenuta ' + person)
# Anche se non c'e' il return, da qui in poi la
   variable non ha piu' 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'      # Variablile globale
welcomePerson() # Benvenuta Chiara
person = 'Sofia'
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)

			Risultato
True	**and**	True	True
True	**and**	False	False
False	**and**	True	False
False	**and**	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	**or**	True	True
True	**or**	False	True
False	**or**	True	True
False	**or**	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

Costrutto while

while

- Costrutto simile al for, che abbiamo gia' visto
- Il while ripete il ciclo finche' una determinata condizone rimane vera
- Qual e' la differenza con il for?

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

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

Outline

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

Liste

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 cassetti numerati da 0 (primo cassetto) a lungezza 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 gia' definita

```
answers = ['Si', 'No']

answers.append('Forse') # Ora answers avra'
    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 lita stessa!
! Si, la lista e' ordinata !

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

Liste

split()

Un altra funzione utile e' 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'?
```

Liste

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])
```

Liste

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'
```

Liste

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'
```

Liste

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

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 dovra' 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 chiamamta secretWord il cui valore e' il risultato della funzione che ritorna una parola casuale dalla lista di parole

L'impiccato - Parte 2

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 sucessiva

L'impiccato - Parte 3

```
  +---+
  O   |
 /|\  |
 / \  |
      |
     ===

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 sara' 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 e' 'ciao' e ho inserito correttamente le lettere 'i o' il risultato dovra' essere ['_', 'i', '_', 'o']
- Stampa a video la lista appena creata e popolata dalle lettere corrette inserite oppure dai '_'

L'impiccato - Parte 5

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)

L'impiccato - Parte 6

Leggi la parte finale del codice fornito

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