

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

Costruiremo un semplice gioco che chiameremo **Guess the number**. Il computer penserà un numero da 1 a 20 e ci darà sei possibilità avvertendoci, per ognuna di esse, se il numero da noi tentato è più grande o più piccolo di quello che ha pensato.

# Guess the number

Hello! What is your name?

Sofia

Well, Sofia, I am thinking of a number between 1 and 20.

Take a guess.

10

Your guess is too high.

Take a guess.

2

Your guess is too low.

Take a guess.

4

Good job, Sofia! You guessed my number in 3 guesses!

# Guess the number

## Per iniziare

- Apriamo Spyder, l'IDE che useremo per programmare in Python
- Troviamo già un file vuoto che ci aspetta..
- ..Salviamo il file tramite **File** ▷ **Save as**
- Rinominiamo il file *guess.py*, selezioniamo la cartella di destinazione (DA CHIEDERE A ENRI: QUALE CARTELLA DI DESTINAZIONE?) e iniziamo a riempirlo di codice!

# Outline

- 1 Moduli, import e funzione per generare numeri casuali
- 2 Istruzioni di controllo di flusso e organizzazione in blocchi
- 3 Conversione di valori e Boolean data type
- 4 Operatori di comparazione e if statement
- 5 Carattere di escape e uso di apici e virgolette

```
# This is a Guess the Number game
import random
guessesTaken = 0
print('Hello! What is your name?')
myName = input()
number = random.randint(1, 20)
```

# Moduli, import e funzione per generare numeri casuali

## Istruzione di `import` e funzione `randint()`

- Il comando **import** serve per importare un *modulo*, ovvero un programma separato nel quale sono presenti altre funzioni
- Importiamo il modulo *random* dal quale chiamiamo la funzione `randint()` che ci permette di generare un numero casuale all'interno del range formato dai due numeri passati in input dalla funzione. Salviamo questo numero nella variabile **number**
- Catturiamo l'input dell'utente tramite la funzione `input()` e mettiamolo nella variabile `myname`

# Moduli, import e funzione per generare numeri casuali

## Ora prova tu!

- Spostati nella shell di Python
- Importa il modulo `random` tramite il comando `import random` e premi invio
- Invoca la funzione `randint()` tramite il comando `random.randint()` inserendo come parametri della funzione il range che vuoi tu!



# Outline

- 1 Moduli, import e funzione per generare numeri casuali
- 2 Istruzioni di controllo di flusso e organizzazione in blocchi
- 3 Conversione di valori e Boolean data type
- 4 Operatori di comparazione e if statement
- 5 Carattere di escape e uso di apici e virgolette

```
print('Well, ' + myName + ', I am thinking of a  
      number between 1 and 20.')
```

```
for guessesTaken in range(6):  
    print('Take a guess.')
```

```
    guess = input()  
    guess = int(guess)
```

# Istruzioni di controllo di flusso e organizzazione in blocchi

## Blocchi

- Il codice può essere organizzato in *blocchi*, ovvero delle linee di codice che hanno lo stesso numero di spazi prima dell'inizio

Riprendiamo il codice che abbiamo appena visto:

```
for guessesTaken in range(6):  
    ***print('Take a guess.')  
    ***guess=input()  
    ***guess=int(guess)
```

- Consideriamo ogni \* come uno spazio, che chiameremo *indentazione*.

# Istruzioni di controllo di flusso e organizzazione in blocchi

## Blocchi

- Ogni riga che è indentata con lo stesso numero di spazi, fa parte di un **blocco**.
- Nel nostro esempio, le istruzioni che sono indentate con 4 spazi (\*) fanno tutte parte del blocco che parte dal comando **for**
- Il comando **for** segna l'inizio di un **loop** o **ciclo**

# Istruzioni di controllo di flusso e organizzazione in blocchi

## For statement

- Quando il compilatore incontra il comando **for**, entra nel blocco che segue il comando, esegue tutte le istruzioni del blocco e riparte dall'inizio del blocco.
- Questo loop viene eseguito tante volte quanto il numero passato alla funzione **range()**
- Vediamo un esempio

# Istruzioni di controllo di flusso e organizzazione in blocchi

Ora prova tu!

Ora proviamo a copiare questo codice nella console di Python per vedere cosa succede!

```
for j in range(4):  
    print('Hello! The variable j is set to', j)
```

# Outline

- 1 Moduli, import e funzione per generare numeri casuali
- 2 Istruzioni di controllo di flusso e organizzazione in blocchi
- 3 Conversione di valori e Boolean data type
- 4 Operatori di comparazione e if statement
- 5 Carattere di escape e uso di apici e virgolette

# Conversione di valori e Boolean data type

## Conversione di valori o **casting**

- La funzione `int()` restituisce il valore passato in input come un `integer`, un valore intero.
- Vediamo qualche esempio per capire

```
>>> int('42')  
>>> 3 + int('2')  
>>> int('forty-two')
```

Copiamo ciascuna riga premendo subito invio sulla console di Python, e osserviamo cosa succede per ognuna delle 3 istruzioni



# Conversione di valori e Boolean data type

## Conversione di valori o **casting**

- Come abbiamo notato, la funzione `int()` accetta come argomenti anche delle stringhe, purchè esse contengano solo numeri
- Oltre alla funzione di `int()`, le funzioni `float()` e `str()` agiscono nello stesso modo, convertendo l'argomento in valori **float** e **string** rispettivamente.

```
>>> float('42')
>>> float(96)
>>> str(58)
>>> str(58.36)
```

Copiamo ciascuna riga premendo subito invio sulla console di Python, e osserviamo cosa succede per ognuna delle istruzioni

# Conversione di valori e Boolean data type

## Boolean data type

- Abbiamo visto in precedenza i "data type" che ogni valore può assumere, tra cui *integer*, *float* e *string*.
- Il data type Boolean, ***bools*** per gli amici, ha due valori: **True** o **False**

```
>>> var = True
>>> var
>>> var = False
>>> var
```

Copiamo ciascuna riga premendo subito invio sulla console di Python, e osserviamo cosa succede per ognuna delle istruzioni

# Outline

- 1 Moduli, import e funzione per generare numeri casuali
- 2 Istruzioni di controllo di flusso e organizzazione in blocchi
- 3 Conversione di valori e Boolean data type
- 4 Operatori di comparazione e if statement**
- 5 Carattere di escape e uso di apici e virgolette

```
****if guess < number:  
*****print('Your guess is too low.')
```

```
****if guess > number:  
*****print('Your guess is too high.')
```

```
****if guess == number:  
*****break
```

# Operatori di comparazione e if statement

## Cosa vuol dire?

- Le tre istruzioni **if** sono simili tra loro, e ciascuna di esse costituisce un **sotto-blocco** del blocco **for**
- **RICORDA!** **guess** e **number** sono due variabili che contengono entrambe un numero!
- L'istruzione **if**, **se** in italiano, indica una condizione per cui:
  - **se è vera** viene eseguito il codice all'interno del sotto-blocco
  - **altrimenti** viene eseguita quella successiva, **SENZA** eseguire l'istruzione all'interno.
- Vediamo di capire meglio con un esempio

# Operatori di comparazione e if statement - Esempio

if guess < number

se  
il valore che  
contiene la  
variabile "guess"  
è  
minore  
del valore che  
contiene la variabile  
"number"

print('Your guess is too low.')

stampa a video la stringa 'Your guess is too low.'

if guess > number

se  
il valore che  
contiene la  
variabile "guess"  
è  
maggiore  
del valore che  
contiene la variabile  
"number"

print('Your guess is too high.')

stampa a video la stringa 'Your guess is too high.'

if guess == number

se  
il valore che  
contiene la  
variabile "guess"  
è uguale  
del valore che  
contiene la variabile  
"number"

break

esce dal ciclo for

# Operatori di comparazione e if statement

## Operatori di comparazione

- Il comando **break** permette di interrompere l'esecuzione del ciclo **for** in anticipo, prima della sua fine naturale
- Abbiamo visto alcuni operatori di comparazione, vediamo ora i principali:

Operatore	Significato
<	minore di
>	maggiore di
<=	minore uguale di
>=	maggiore uguale di
==	uguale
!=	diverso

# Outline

- 1 Moduli, import e funzione per generare numeri casuali
- 2 Istruzioni di controllo di flusso e organizzazione in blocchi
- 3 Conversione di valori e Boolean data type
- 4 Operatori di comparazione e if statement
- 5 Carattere di escape e uso di apici e virgolette



```
if guess == number:
    guessesTaken = str(guessesTaken + 1)
    print('Good job, ' + myName + '! You
          guessed my number in ' +
          guessesTaken + ' guesses!')

if guess != number:
    number = str(number)
    print("That\'s too bad. The number I was
          thinking of was " + number + ".")
```

# Caratteri di escape e uso di apici e virgolette

## Tiriamo le fila!

- ...Ed eccoci arrivati all'ultimo frammento di codice di questo gioco interattivo!
- Ritroviamo due costrutti `if` che servono per mandare un messaggio finale all'utente:
  - **SE** il giocatore ha vinto, ci congratuliamo (se l'è meritato!) e stampiamo quanti tentativi ci ha messo per indovinare il numer;
  - **SE** il giocatore non ha vinto, stampiamo semplicemente il numero che aveva pensato il programma
- Vi ricordate `str()` cosa fa?
- Avete notato qualcosa di diverso tra queste due ultime `print()`?

# Caratteri di escape e uso di apici e virgolette

## Ultimi concetti

- Queste due ultime `print()` hanno qualcosa di diverso tra loro...
- Nella prima `print()` sono stati gli apici (') mentre per la seconda `print()` sono state usate le doppie virgolette (") per racchiudere le stringhe da stampare
- Quindi.... quale usiamo?
- ...Tutte e due! Basta che non le mischiamo!

```
>>> print('Hello world')
>>> print("Hello world")
>>> print('Hello world')
```

Copiamo ciascuna riga premendo subito invio sulla console di Python, e osserviamo cosa succede per ognuna delle 3 istruzioni!

# Caratteri di escape e uso di apici e virgolette

## Ultimi concetti

- C'è un'altra cosa strana in una delle due `print()`... A cosa serve quel `\` prima dell'apice tra *That* e *s*?
- Viene usato, all'interno di stringhe, come **carattere di escape**
- Cos'è un carattere di escape? Serve per poter stampare i caratteri speciali che hanno già un loro significato, come l'apice (`'`), che viene interpretato come inizio stringa
- Vediamo ora cosa può permettere di stampare questo carattere di escape:

Carattere di escape	Cosa viene stampato
<code>\\</code>	Backslash ( <code>\</code> )
<code>\'</code>	Apice ( <code>'</code> )
<code>\"</code>	Doppie virgolette ( <code>"</code> )
<code>\n</code>	Andata a capo
<code>\t</code>	Tabulazione