

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

Impareremo ad usare `pygame`, un modulo che permette di creare giochi completi di animazioni, suoni e comandi con il mouse.

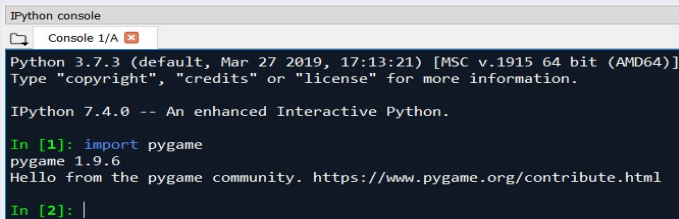
Outline

- 1 Installare pygame
- 2 Primi comandi
- 3 Tuple e colori
- 4 Scrivere testi in pygame
- 5 Disegnare con pygame
- 6 Display e Game loop

Installare pygame

Come installare pygame

- Per installare pygame sul proprio computer è sufficiente andare su **Start > anaconda3 > anaconda prompt** e digitare nella finestra che appare il comando `pip install pygame`
- Per controllare se è stato correttamente installato scriviamo sulla shell di Python il comando `import pygame` e verifichiamo che l'output sia simile a quello in figura:



```
IPython console
Console 1/A
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.4.0 -- An enhanced Interactive Python.

In [1]: import pygame
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html

In [2]: |
```

- Sui pc della scuola è già installato, quindi ora iniziamo a vedere qualche comando!

Outline

- 1 Installare pygame
- 2 Primi comandi**
- 3 Tuple e colori
- 4 Scrivere testi in pygame
- 5 Disegnare con pygame
- 6 Display e Game loop

Import e impostazione della finestra

Innanzitutto importiamo e inizializziamo il modulo `pygame` nel nostro programma con queste semplici istruzioni:

```
import pygame, sys
from pygame.locals import *

pygame.init()
```



Un *pixel* è il più piccolo punto dello schermo del PC che si illumina di tutti i colori. Tutti i pixel lavorano insieme per mostrare l'immagine che vogliamo!

Per impostare la finestra che utilizzeremo il metodo `set_mode()` del modulo `display`, al quale ci si accede dal modulo `pygame`

```
windowSurface = pygame.display.set_mode((500,
    400), 0, 32)
```

In questo modo, viene impostata una finestra larga **500** pixel e alta **400** pixel. Gli altri due parametri (0 e 32), per questo corso, li copiamo senza approfondire.

Outline

- 1 Installare pygame
- 2 Primi comandi
- 3 Tuple e colori**
- 4 Scrivere testi in pygame
- 5 Disegnare con pygame
- 6 Display e Game loop

Tuple e colori 1/3

Tuple

- Una tupla è simile ad una lista, con la differenza che si usano le parentesi `()` al posto che le `[]`. Non possono essere modificate.

```
>>> spam = ('Life', 'Universe', 'Everything', 42)
>>> spam[0]
>>> spam[3]
>>> spam[3] = 'Hello'
```

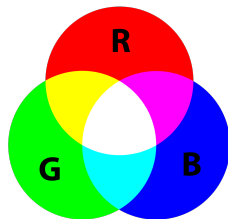
Eseguiamo, una per volta queste righe e osserviamo il comportamento! All'interno del metodo `set_mode()` abbiamo usato una tupla per definire la grandezza della finestra in pixel. Vedremo che ci saranno utili in seguito!

Tuple e colori 2/3

Colori

- Ogni pixel viene illuminato da 3 colori primari quali **rosso**, **verde** e **blu** (RGB). Combinando questi tre semplici colori, si è in grado di ottenere qualsiasi altro colore.
- In pygame, per definire i colori useremo delle tuple formate dai tre valori (**R**ed - **G**reen - **B**lue)

```
# Set up the colors.  
BLACK = (0, 0, 0)  
WHITE = (255, 255, 255)  
RED = (255, 0, 0)  
GREEN = (0, 255, 0)  
BLUE = (0, 0, 255)
```



Tuple e colori 3/3

Color	RGB value
Black	(0, 0, 0)
Blue	(0, 0, 255)
Gray	(128, 128, 128)
Green	(0, 128, 0)
Lime	(0, 255, 0)
Purple	(128, 0, 128)
Red	(255, 0, 0)
Teal	(0, 128, 128)
White	(255, 255, 255)
Yellow	(255, 255, 0)

Colori

- Ogni numero della tupla rappresenta la "quantità" di ciascun colore, che può andare da 0 a 255.
- Il primo numero, quindi, rappresenta la quantità di rosso, il secondo la quantità di verde e il terzo quella del blu.

Outline

- 1 Installare pygame
- 2 Primi comandi
- 3 Tuple e colori
- 4 Scrivere testi in pygame**
- 5 Disegnare con pygame
- 6 Display e Game loop

Una parentesi: gli oggetti

Oggetti

- Un oggetto è semplicemente un altro nome per chiamare i data type che hanno dei metodi.
- Per esempio, il data type *string* è un oggetto, siccome possiede dei metodi come `lower()`.
- Il metodo `set_mode()` visto in precedenza permette di creare un *Surface object* che rappresenta la finestra. Su di esso potremo evocare altri metodi per modificare tale finestra.

```
# Set up the fonts.  
basicFont = pygame.font.SysFont(None, 48)
```

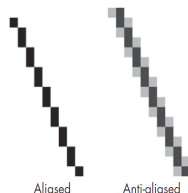
Impostare un font

- Il metodo `pygame.font.SysFont()` permette di creare un *Font object*. Questa funzione prende in input due parametri: il primo è il nome del font (in questo caso viene impostato quello di default) e il secondo parametro è la grandezza.
- Su questo *Font object*, che abbiamo salvato nella variabile `basicFont`, possiamo invocare alcuni metodi, tra cui `render()`

```
# Set up the text.  
text = basicFont.render('Hello world!', True,  
    WHITE, BLUE)
```

Impostare un font

- Questo metodo permette di creare un *Surface object* con il testo indicato nel primo parametro del metodo.
- Il secondo parametro permette di creare un effetto "*blur*" al testo, chiamato anti-alias, come in figura.
- Il terzo e quarto paragrafo indicano il colore del testo e del backgroun, (sfondo), rispettivamente.



Come disporre un testo nella finestra

```
textRect = text.get_rect()  
textRect.centerx =  
    windowSurface.get_rect().centerx  
textRect.centery =  
    windowSurface.get_rect().centery
```

Disposizione testo

- Dal momento che abbiamo creato un *Font object*, dobbiamo predisporre un *Rect object* che servirà per disporre all'interno della finestra il nostro testo. Per fare ciò, utilizziamo il metodo `get_rect()`.
- Possiamo accedere ad alcuni **attributi** di questo oggetto.
- Un **attributo** rappresenta una variabile all'interno dell'oggetto, il cui valore può essere cambiato.

Tabella di attributi Rect object

Ecco alcuni attributi di questo oggetto, che potranno esserci utili in futuro.

Table 17-2: Rect Attributes

pygame.Rect attribute	Description
myRect.left	Integer value of the x-coordinate of the left side of the rectangle
myRect.right	Integer value of the x-coordinate of the right side of the rectangle
myRect.top	Integer value of the y-coordinate of the top side of the rectangle
myRect.bottom	Integer value of the y-coordinate of the bottom side of the rectangle
myRect.centerx	Integer value of the x-coordinate of the center of the rectangle
myRect.centery	Integer value of the y-coordinate of the center of the rectangle
myRect.width	Integer value of the width of the rectangle
myRect.height	Integer value of the height of the rectangle
myRect.size	A tuple of two integers: (width, height)
myRect.topleft	A tuple of two integers: (left, top)
myRect.topright	A tuple of two integers: (right, top)
myRect.bottomleft	A tuple of two integers: (left, bottom)
myRect.bottomright	A tuple of two integers: (right, bottom)
myRect.midleft	A tuple of two integers: (left, centery)
myRect.midright	A tuple of two integers: (right, centery)
myRect.midtop	A tuple of two integers: (centerx, top)

Outline

- 1 Installare pygame
- 2 Primi comandi
- 3 Tuple e colori
- 4 Scrivere testi in pygame
- 5 Disegnare con pygame**
- 6 Display e Game loop

Colorare un oggetto

fill()

- E' possibile riempire un oggetto di colore tramite il metodo `fill()`.
- Questa funzione prende in ingresso una tupla che rappresenta un colore.
- Nel nostro esempio, abbiamo usato questa funzione per impostare lo sfondo della finestra.

```
# Draw the white background onto the surface.  
windowSurface.fill(WHITE)
```

Disegnare un poligono

polygon()

Questa funzione permette di creare qualsiasi tipo di poligono. Vediamone gli argomenti in input:

- la finestra su cui disegnare il poligono;
- il colore;
- una tupla di n tuple (coppie di valori x,y) che indicano i punti del poligono. L'ultimo punto si collegherà automaticamente al primo;
- un intero, opzionale, che determina lo spessore del contorno. Senza di esso, il poligono è riempito.

```
# Draw a green polygon onto the surface.  
pygame.draw.polygon(windowSurface, GREEN, ((146,  
    0), (291, 106),  
    (236, 277), (56, 277), (0, 106)))
```

Disegnare una linea

line()

Questa funzione permette di creare una linea. Vediamone gli argomenti in input:

- la finestra su cui disegnare la linea;
- il colore;
- una tupla di valori x,y che indica un capo della linea;
- una tupla di valori x,y che indica l'altro capo della linea;
- un intero, opzionale, che indica lo spessore della linea

```
pygame.draw.line(windowSurface, BLUE, (60, 60),  
                 (120, 60), 4)  
pygame.draw.line(windowSurface, BLUE, (120, 60),  
                 (60, 120))
```

Disegnare un cerchio

circle()

Questa funzione permette di creare un cerchio. Vediamone gli argomenti in input:

- la finestra su cui disegnare il cerchio;
- il colore;
- una tupla di valori x,y che indica il centro del cerchio;
- un intero che indica il raggio;
- un intero, opzionale, che indica lo spessore della linea. Se è 0, il cerchio viene riempito.

```
# Draw a blue circle onto the surface.  
pygame.draw.circle(windowSurface, BLUE, (300,  
    50), 20, 0)
```

Disegnare un ellisse

ellipse()

Questa funzione permette di creare un ellisse. Vediamone gli argomenti in input:

- la finestra su cui disegnare l'ellisse;
- il colore;
- una tupla contenente l'angolo di sinistra e in alto, la larghezza e l'altezza dell'ellisse;
- un intero, opzionale, che indica lo spessore della linea. Se è 0, l'ellisse viene riempito.

```
# Draw a red ellipse onto the surface.  
pygame.draw.ellipse(windowSurface, RED, (300,  
    250, 40, 80), 1)
```

Disegnare un rettangolo

circle()

Questa funzione permette di creare un cerchio. Vediamone gli argomenti in input:

- la finestra su cui disegnare il cerchio;
- il colore;
- una tupla con 4 valori, rispettivamente le coordinate x,y per gli spigoli alto-sinistra, larghezza e altezza

```
pygame.draw.rect(windowSurface, RED,  
    (textRect.left - 20,  
textRect.top - 20, textRect.width + 40,  
    textRect.height + 40))
```

Disegnare una finestra dentro l'altra

blit()

Questo metodo permette di disegnare due finestre, una dentro l'altra. Nel nostro esempio, infatti, abbiamo la finestra contenente la scritta e la finestra contenente le figure. Vediamo come sovrapporle.

```
# Draw the text onto the surface.  
windowSurface.blit(text, textRect)
```

In questo caso, la finestra contenente la scritta viene inserita all'interno della finestra con le figure.

Facciamo bene attenzione quindi all'ordine con cui sovrapponiamo le due finestre!

Outline

- 1 Installare pygame
- 2 Primi comandi
- 3 Tuple e colori
- 4 Scrivere testi in pygame
- 5 Disegnare con pygame
- 6 Display e Game loop**

Stampare a schermo le nostre creazioni

```
# Draw the window onto the screen.  
pygame.display.update()
```

Con questo semplice metodo riusciamo a rendere visibile sul nostro display tutto ciò che abbiamo creato fino ad ora!

Eventi e game loop

Eventi

- Un evento è qualcosa che accade all'interno del nostro programma, come un click o movimento del mouse o la pressione di un tasto.
- Quando si verifica un evento, viene generato un *Event object*, il quale può essere intercettato tramite la funzione `get()`.
- L'attributo `type` dell'oggetto *Event* contiene alcuni tipi di eventi predefiniti, come `QUIT`.

```
# Run the game loop.  
while True:  
    for event in pygame.event.get():  
        if event.type == QUIT:  
            pygame.quit()  
            sys.exit()
```

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

