



INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL



Cofinanciado pela  
União Europeia

Os Fundos Europeus mais próximos de si.



REPÚBLICA  
PORTUGUESA

TRABALHO, SOLIDARIEDADE  
E SEGURANÇA SOCIAL

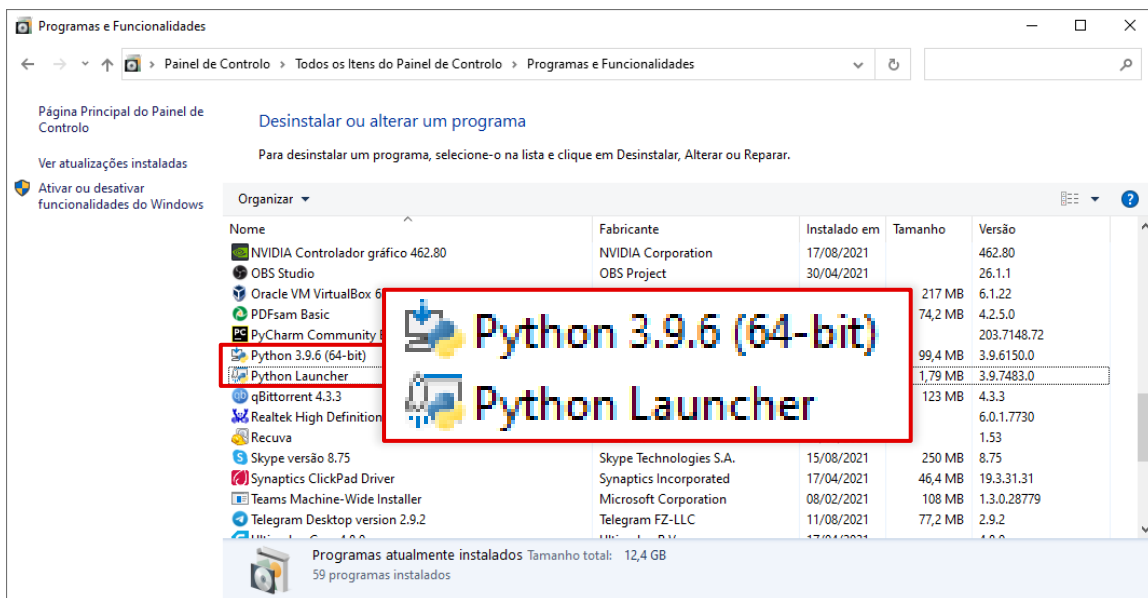


UFCD 10794 - Programação avançada com  
Python

# Instalação do Python

# Instalação do Python

Para garantir que o Python fica devidamente configurado no sistema, **desinstalar todas as versões que estejam instaladas** para fazer uma instalação nova.

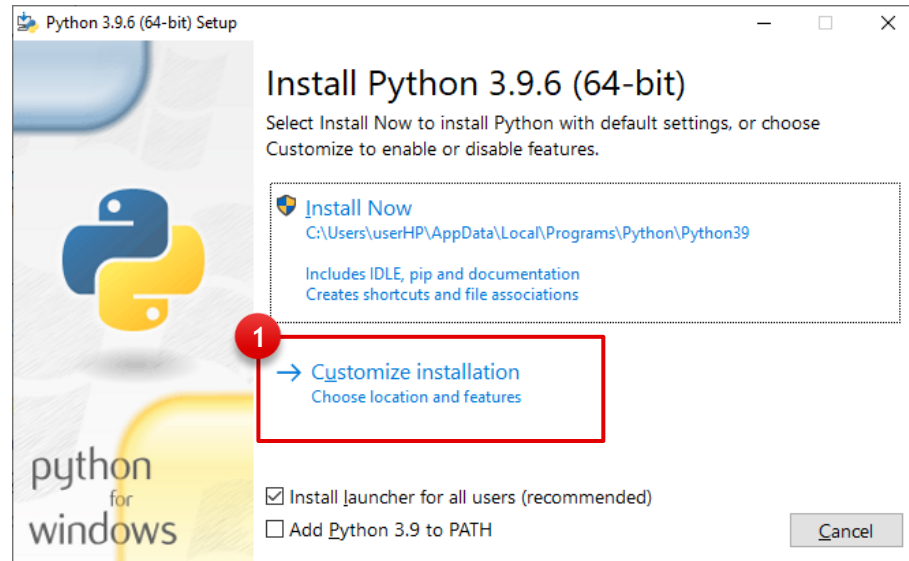


# Instalação do Python

Depois de fazer download do Python no site:

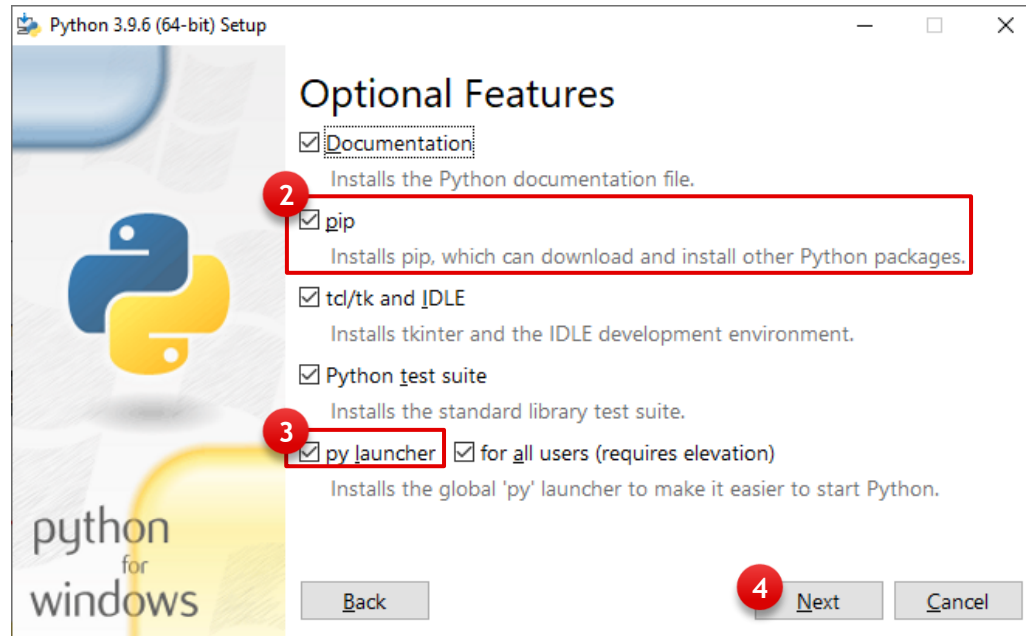
<https://www.python.org/downloads/>

- Optar pela instalação personalizada.



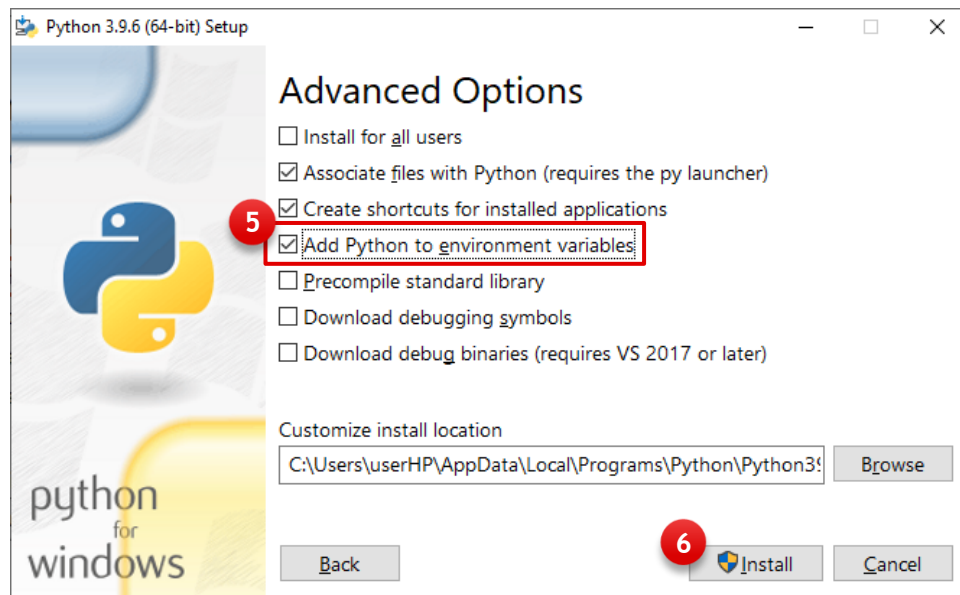
# Instalação do Python

- Durante a instalação garantir que estas opções estão selecionadas:



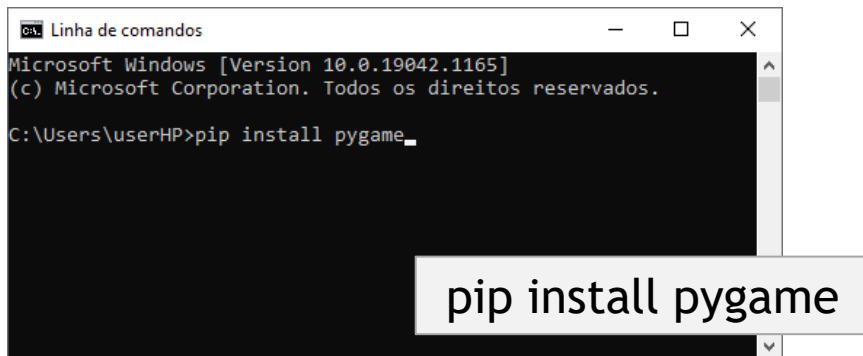
# Instalação do Python

- E no passo seguinte garantir esta opção também selecionada:



# Instalar a biblioteca pygame

- Abrir a consola e correr o comando:



A screenshot of a Windows Command Prompt window. The title bar reads 'C:\> Linha de comandos'. The window content shows the following text: 'Microsoft Windows [Version 10.0.19042.1165]', '(c) Microsoft Corporation. Todos os direitos reservados.', and 'C:\Users\userHP>pip install pygame\_'. A white callout box with the text 'pip install pygame' is positioned over the command line.

```
C:\> Linha de comandos
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. Todos os direitos reservados.
C:\Users\userHP>pip install pygame_
```

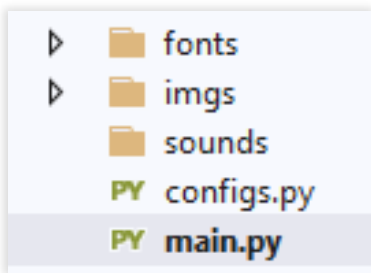
pip install pygame

# Primeiros passos



# Estrutura do código

- Depois de criar o seu projeto, e antes de começar a desenvolver o jogo, garanta que tem os seguintes ficheiros e pastas:



- **fonts**: tipos de letra a importar
- **imgs**: imagens a usar no jogo
- **sounds**: sons a usar no jogo
- **configs.py**: ficheiro de configurações
- **main.py**: ficheiro principal do jogo

# Estrutura do código

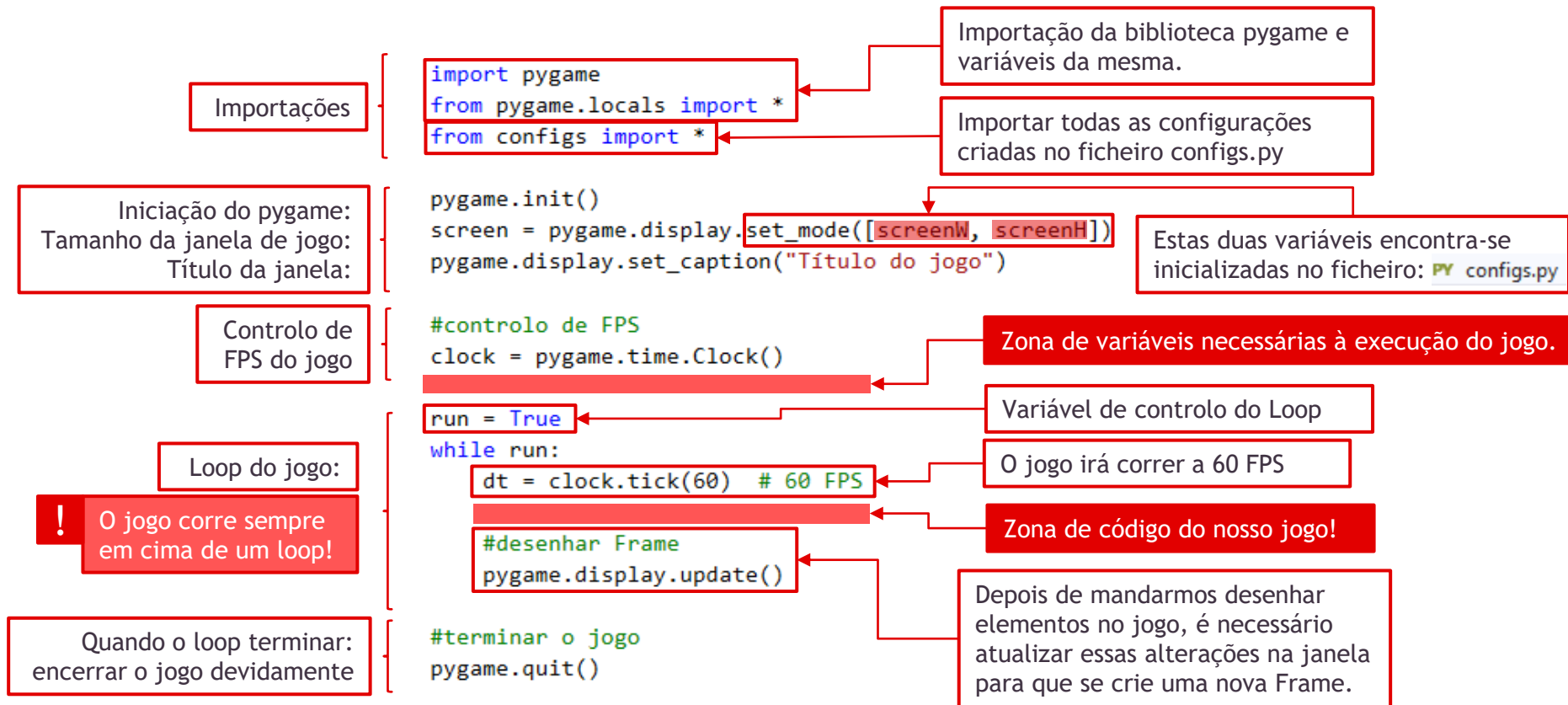
- No ficheiro `configs.py` garanta o seguinte código:

```
1 screenW = 600
2 screenH = 301
```

Tamanho da  
janela de jogo

# Estrutura do código

- No ficheiro `main.py` garanta o seguinte código:



# Estrutura do código

- Para evitar que os elementos individuais no jogo pisquem, as frames são primeiramente desenhadas num ecrã temporário e só depois de concluídas são colocadas na janela:

É criada um “ecrã temporário” que vai recebendo alterações.

Quando esta linha é executada, a imagem do ecrã temporário é transferida para a janela do jogo.

`pygame.display.update()`



Ecrã temporário

! Onde primeiramente pintamos (sem que o utilizador veja)



Janela do jogo

! Ao fazer “update”, a imagem é transferida da tela de pintura para a janela do jogo.

```
import pygame
from pygame.locals import *
from configs import *
```

```
pygame.init()
screen = pygame.display.set_mode([screenW, screenH])
pygame.display.set_caption("Título do jogo")
```

```
#controlo de FPS
clock = pygame.time.Clock()
```

```
run = True
while run:
    dt = clock.tick(60) # 60 FPS
```

```
pygame.draw.rect(screen, pygame.Color('red'), (50, 20, 120, 100))
```

```
#desenhar Frame
pygame.display.update()
```

```
#terminar o jogo
pygame.quit()
```

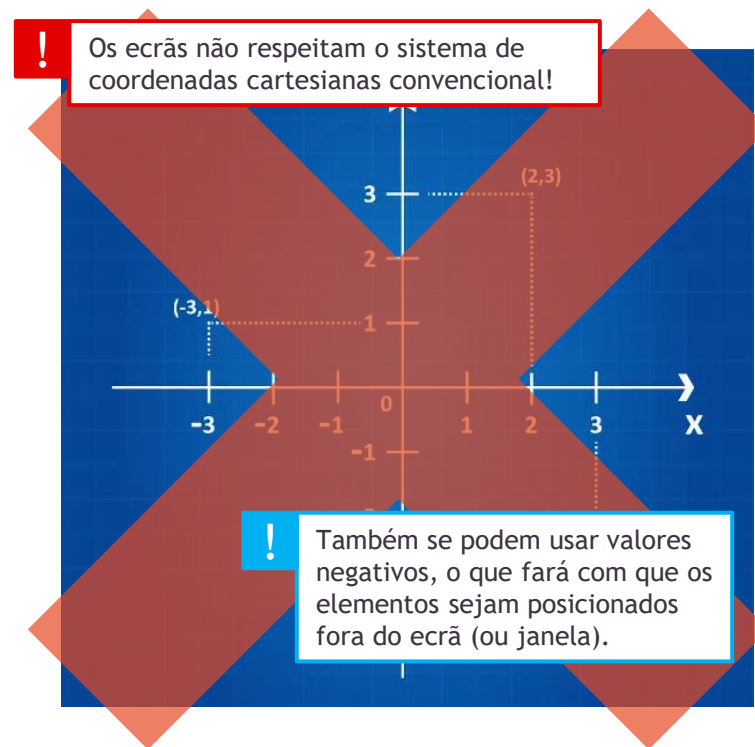
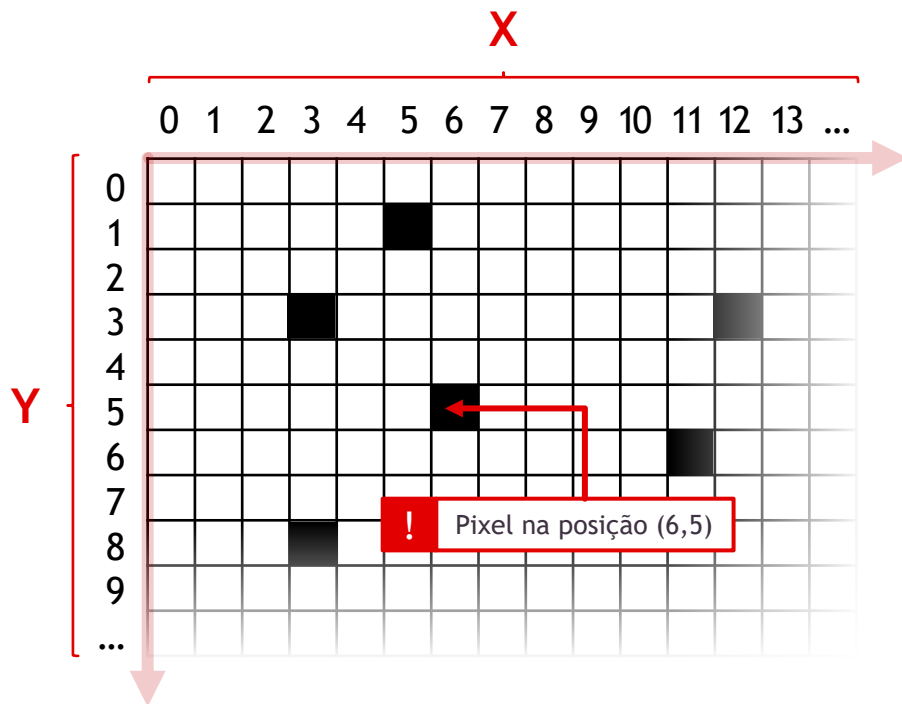
i

Serve para desenhar um quadrado vermelho com o tamanho 50x20px na posição X, Y: 120-100



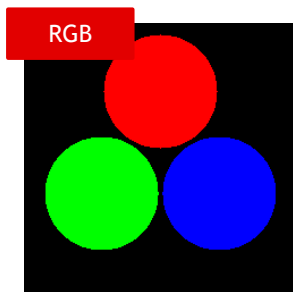
# Entender as coordenadas da tela

Antes de avançar é importante entender as coordenadas da tela, uma vez que diferem das convencionais coordenadas cartesianas.



# Sistema de cores

- Sempre que algum elemento necessite de cores, pode optar pelo **sistema RGB**:



! Sintaxe: ( R, G, B )

```
screen.fill( (255, 0, 0) )
```



Screen.fill ( xxx, xxx, xxx ) atribuí uma cor de fundo à janela

RGB (Red, Green, Blue)

Cada um destes valores está compreendido entre 0 e 255, que corresponde respetivamente a 0 e 100% da intensidade de cada cor.

- Ou um sistema de **código de cores**:

Cores mais usadas	white	green
	yellow	dark green
	orange	brown
	red	tan
	magenta	light grey
	purple	medium grey
	blue	dark grey
	cyan	black

```
screen.fill( pygame.Color('green') )
```



Estas cores são compatíveis com a grande maioria do sistema de nome cores usado no HTML:

<https://htmlcolorcodes.com/color-names/>

# Sistema de cores

- Ou o **sistema de cores Hexadecimal** (que consiste numa abreviação das cores RGB no sistema hexadecimal: **#RRGGBB**):

The diagram illustrates the process of selecting a color using a web-based color picker and applying it in a Python program. On the left, a screenshot of a Google search for "colorpicker" shows the "Colorpicker" tool. A red box highlights the HEX color code "#2b5ed6" at the bottom of the tool. A red arrow points from this box to a code editor on the right. The code editor shows the line `screen.fill(pygame.Color("#2b5ed6"))`, with a red box around the color string. Another red arrow points from the code editor to a game window titled "Título do jogo", which displays a solid blue rectangle, representing the result of the color fill operation.

! Pesquise "Colorpicker" no google para ter acesso a esta ferramenta.

```
screen.fill(pygame.Color("#2b5ed6"))
```

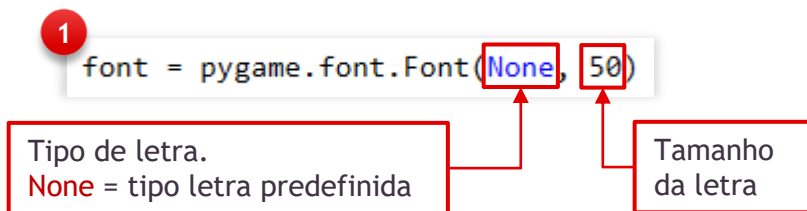
Título do jogo

**Desenhar texto**

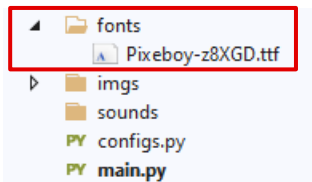


# Desenhar texto

- Para desenhar texto no ecrã, é necessário primeiramente definir um tipo de letra:



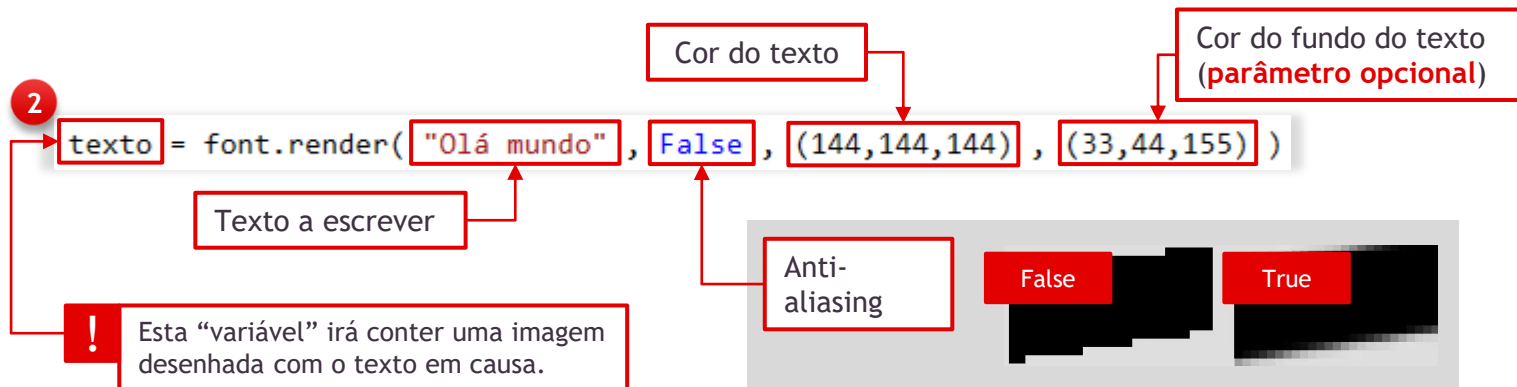
- É também possível importar tipos de letra (de dentro de uma pasta por exemplo):



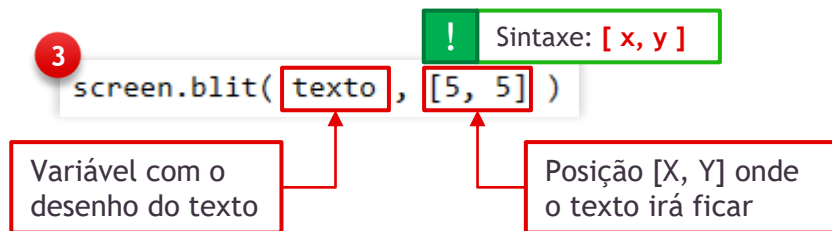
Site para download gratuito de tipos de letra:  
<https://fonts.google.com/>

# Desenhar texto

- Uma vez definido o tipo de letra, necessitamos seguidamente de indicar o texto:



- E adicionar o mesmo à tela:



# Desenhar texto

- Exemplo de onde colocar o código:

```
import pygame
from pygame.locals import *
from configs import *

pygame.init()
screen = pygame.display.set_mode([screenW, screenH])
pygame.display.set_caption("Título do jogo")
```

1

```
font = pygame.font.Font(None, 150)
```

```
#controlo de FPS
clock = pygame.time.Clock()
```

```
run = True
```

```
while run:
```

```
    dt = clock.tick(60) # 60 FPS
```

2

```
    texto = font.render( "Olá mundo" , False , (144,144,144) , (33,44,155) )
```

3

```
    screen.blit( texto , [5, 5] )
```

```
#desenhar Frame
```

```
pygame.display.update()
```

```
#terminar o jogo
```

```
pygame.quit()
```

Tipo de letra  
(fora do loop)

Desenhar o texto com o tipo de  
letra escolhido (dentro do loop)



# Desenhar texto

## Exercício 1

Faça download do tipo de letra “Roboto” no site do Google Fonts.

Guarde na sua pasta “fonts” apenas o ficheiro “Roboto-Light.ttf”.

Escreva no seu ecrã o seu nome pessoal respeitando:

- Tamanho da letra: 20px
- Cor da letra: Verde
- Fundo: castanho
- Posição: 10,10 (X,Y)
- Anti-aliasing: Não



Pedro Ferreira

**Nota:** O seu aplicativo irá bloquear, é normal, já iremos resolver isso!

► Cácula:

(localização , tamanho)

```
font = pygame.font.Font('fonts/Pixeboy-z8XGD.ttf', 50)
```

(Texto, Anti-aliasing?, cor da letra, cor de fundo)

```
texto = font.render("Olá mundo" , False , (144,144,144) , (33,44,155) )
```

(Desenho do texto, posição: [X, Y])

```
screen.blit( texto , [5, 5] )
```



Google Fonts

<https://fonts.google.com/>

white	green
yellow	dark green
orange	brown
red	tan
magenta	light grey
purple	medium grey
blue	dark grey
cyan	black

```
screen.fill( pygame.Color('green') )
```

# Desenhar texto

## Exercício 1 (solução)

```
import pygame
from pygame.locals import *
from configs import *

pygame.init()
screen = pygame.display.set_mode([screenW, screenH])
pygame.display.set_caption("Título do jogo")

font = pygame.font.Font('fonts/Roboto-Light.ttf', 20)

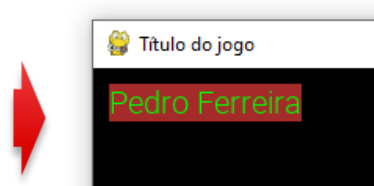
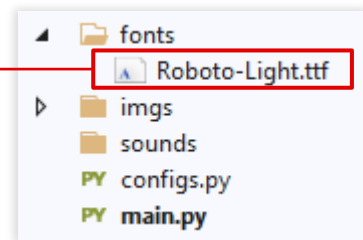
clock = pygame.time.Clock()

run = True
while run:
    dt = clock.tick(60) # 60 FPS

    texto = font.render("Pedro Ferreira", False, "green", "Brown")
    screen.blit(texto, [10, 10])

    pygame.display.update()

pygame.quit()
```



# Eventos da janela

# Eventos da janela

- O seu programa encontra-se a bloquear porque não existem eventos associados à janela.

```
import pygame
from pygame.locals import *
from configs import *

pygame.init()
screen = pygame.display.set_mode([screenW, screenH])
pygame.display.set_caption("Título do jogo")

clock = pygame.time.Clock()

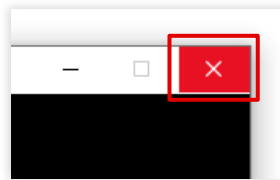
run = True
while run:
    dt = clock.tick(60) # 60 FPS


    #eventos a correr na janela
    for event in pygame.event.get():
        if event.type == pygame.QUIT: # Botão X da janela
            run = False

    {
        pygame.display.update()
    }

pygame.quit()
```

Evento associado ao botão de fechar a janela.



! Quando for premido o botão  da janela, o ciclo do jogo é mandado encerrar (pela variável “run” ficar a False) e o jogo seguidamente terminado.

Zona de acréscimo de novo código  
(depois de verificar os eventos da janela)

# Desenho de primitivas



# Desenho de primitivas

- No pygame é possível desenhar as seguintes formas primitivas:

`pygame.draw.` {  
    `aaline`  
    `aalines`  
    `arc`  
    `circle`  
    `ellipse`  
    `line`  
    `lines`  
    `polygon`  
    `rect`  
} (ecrã, cor, ... )



Mais detalhes em:

<https://www.pygame.org/docs/ref/draw.html>

# Desenho de primitivas

## Linhas



### Linha reta

```
pygame.draw.line (...);  
pygame.draw.aaline (...);
```



### Linhas retas seguidas

```
pygame.draw.lines (...);  
pygame.draw.aalines (...);
```

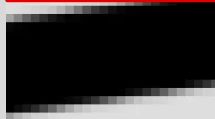


“aa” = anti-aliasing

line, lines:



aaline, aalines:



## Formas



### Retângulo

```
pygame.draw.rect (...);
```



### Polígono

```
pygame.draw.polygon (...);
```



### Círculo

```
pygame.draw.circle (...);
```



### Elipse

```
pygame.draw.ellipse (...);
```



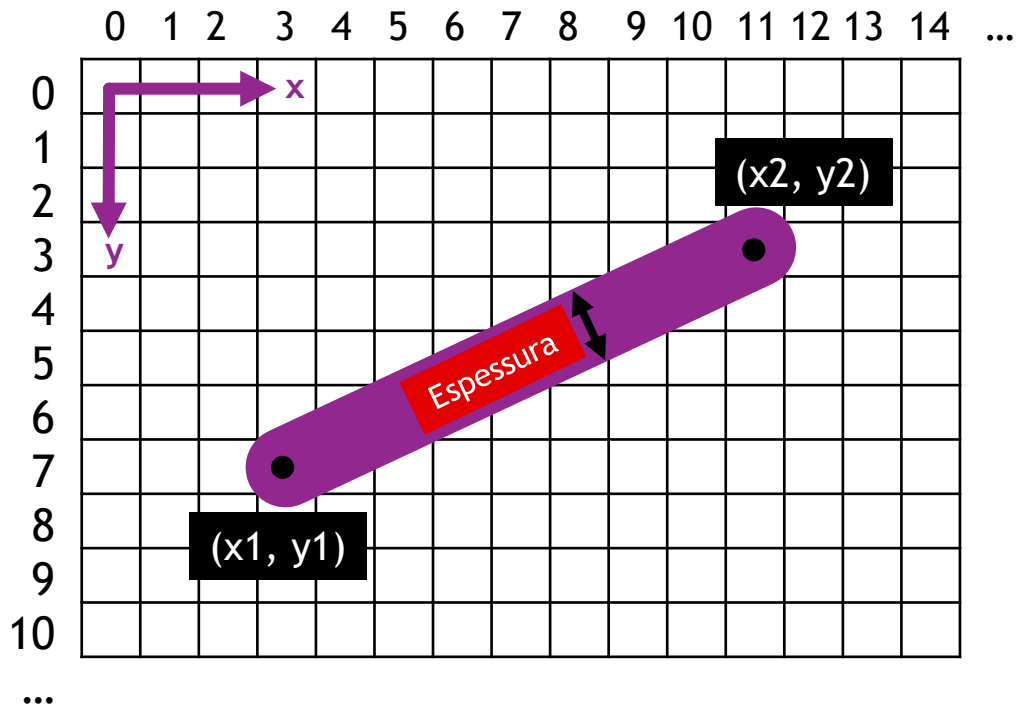
### Arco

```
pygame.draw.arc (...);
```

# Desenho de primitivas

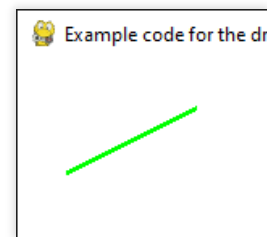
`pygame.draw.line(ecrã, cor, [x1, y1], [x2, y2], espessura)`

`pygame.draw.aaline(ecrã, cor, [x1, y1], [x2, y2], espessura)`



Exemplo:

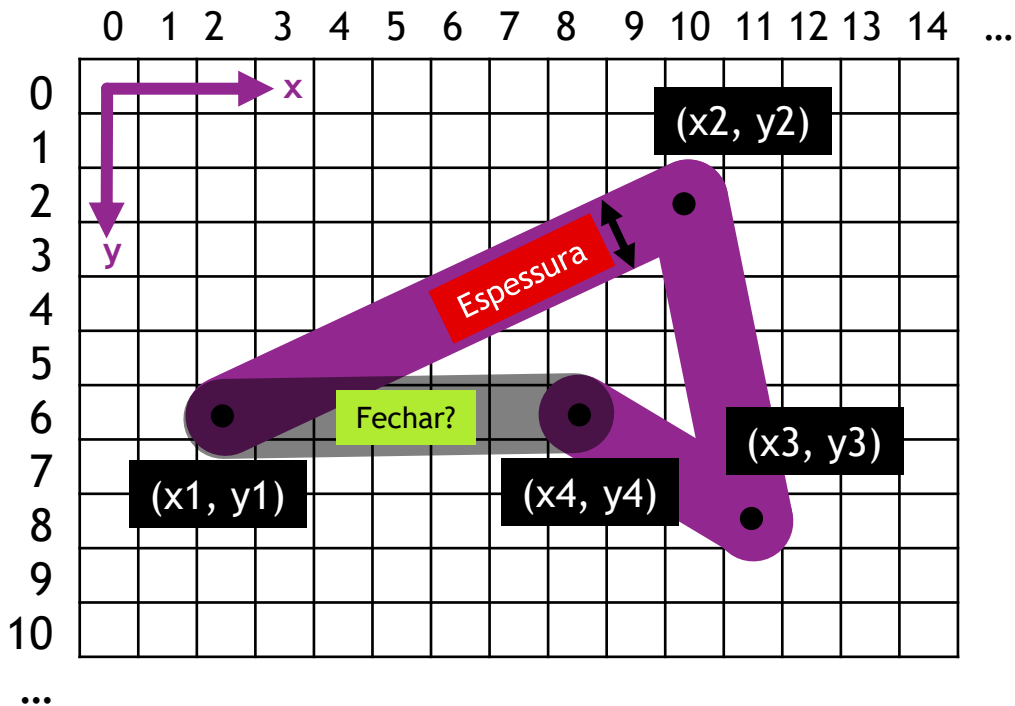
```
pygame.draw.line(  
    screen,  
    (0, 255, 0),  
    [30, 70],  
    [110, 30],  
    3  
)
```



# Desenho de primitivas

`pygame.draw.lines(ecrã, cor, fechar?, [[x1, y1], [x2, y2], [x3, y3], [x..., y...]], espessura)`

`pygame.draw.aalines(ecrã, cor, fecha?, [[x1, y1], [x2, y2], [x3, y3], [x..., y...]], espessura)`

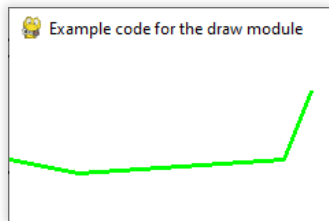


Exemplo:

```
pygame.draw.lines(  
    screen,  
    (0, 255, 0),  
    False,  
    [  
        [0, 80],  
        [50, 90],  
        [200, 80],  
        [220, 30]  
    ],  
    3  
)
```

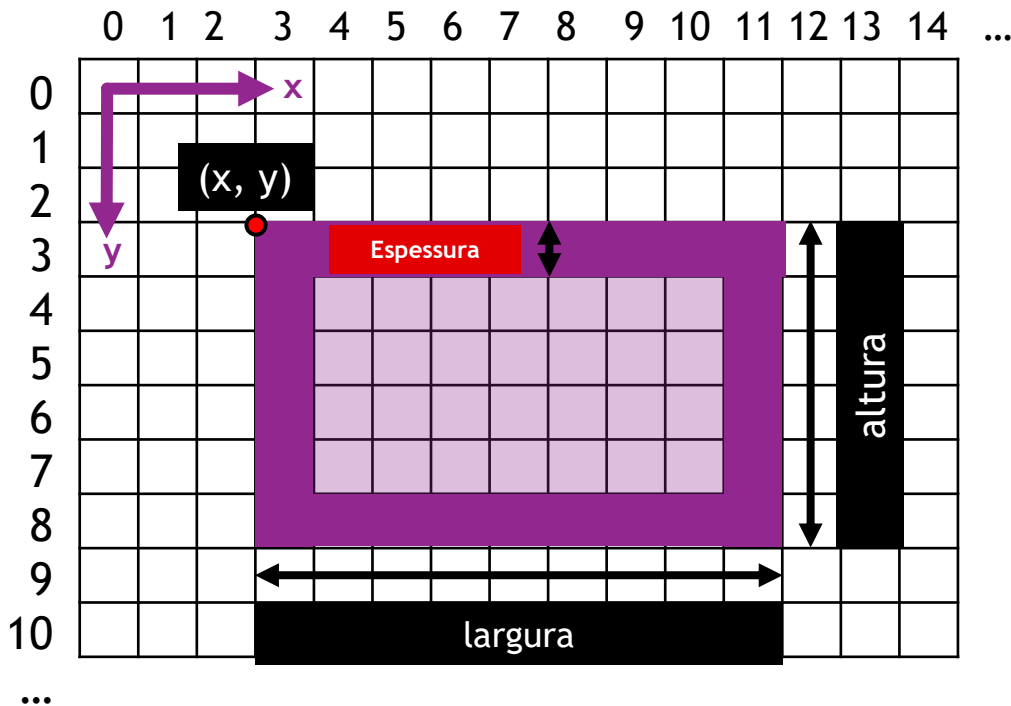


**True:** une o último ponto ao primeiro, fechando a forma.  
**False:** Deixa a forma aberta



# Desenho de primitivas

`pygame.draw.rect(ecrã, cor, [x, y, largura, altura], espessura)`

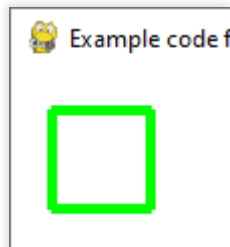


! **Espessura > 0:** Apenas o contorno (interno) nessa espessura  
**Espessura = 0 (ou parâmetro inexistente):** Preenchido

Exemplo:

```
pygame.draw.rect(  
    screen,  
    (0, 255, 0),  
    [20, 20, 50, 50],  
    5  
)
```

```
pygame.draw.rect(  
    screen,  
    (0, 255, 0),  
    [20, 20, 50, 50]  
)
```



# Desenho de primitivas

- ▶ O retângulo pode ainda ser arredondado nos cantos com um valor geral:

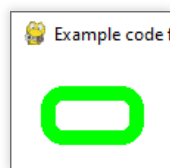
**border\_radius:** raio (geral) em pixels dos cantos arredondados

- ▶ Ou com valores em cantos específicos:

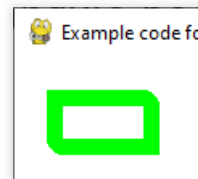


- ▶ Exemplo:

```
pygame.draw.rect(  
    Ecrã: screen,  
    Cor: (0, 255, 0),  
    [x, y, largura, altura]: [20, 20, 70, 40],  
    Espessura: 10,  
    Raio do arredondado: border_radius = 15  
)
```

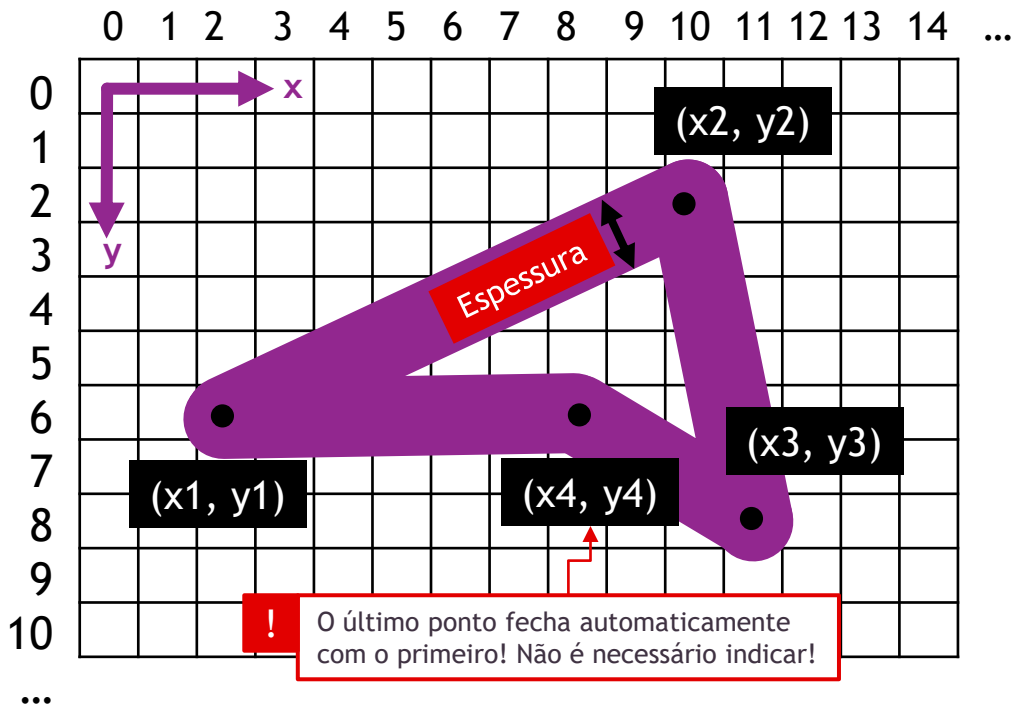


```
pygame.draw.rect(  
    screen,  
    (0, 255, 0),  
    [20, 20, 70, 40],  
    10,  
    border_radius = 10,  
    border_top_left_radius = 0,  
    border_bottom_right_radius = 0  
)
```



# Desenho de primitivas

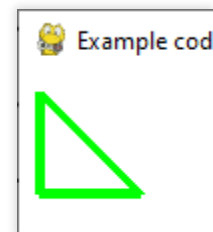
`pygame.draw.polygon(ecrã, cor, [[x1, y1], [x2, y2], [x3, y3], [x..., y...]], espessura)`



**!** **Espessura > 0:** Apenas o contorno (interno) nessa espessura  
**Espessura = 0 (ou parâmetro inexistente):** Preenchido

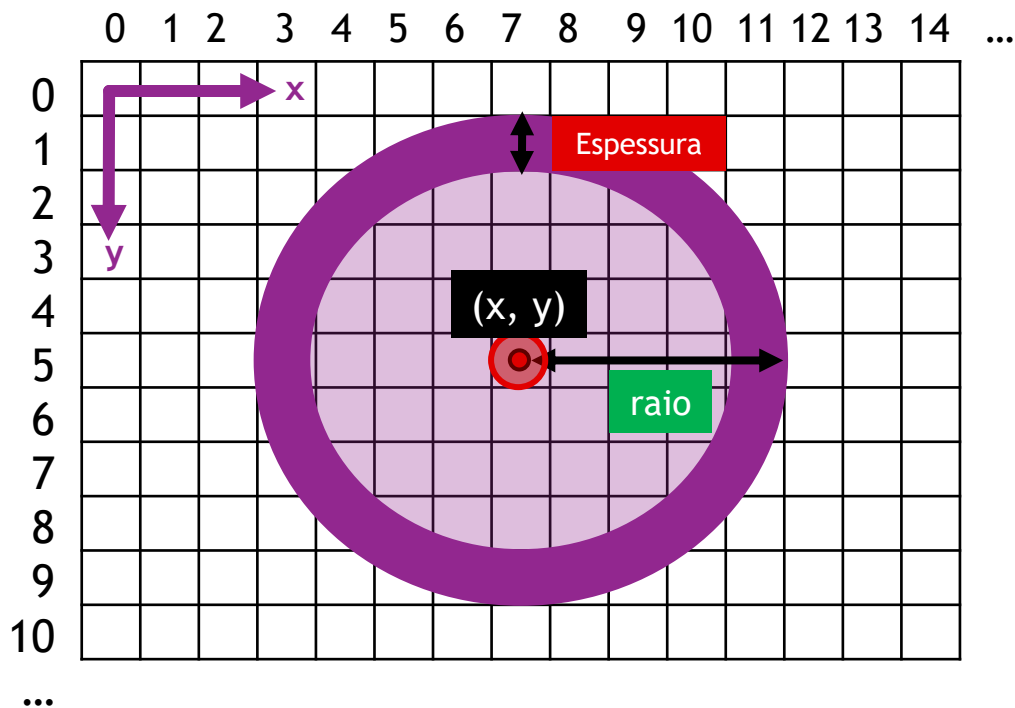
Exemplo:

```
pygame.draw.polygon(  
    screen,  
    (0, 255, 0),  
    [  
        [10, 10],  
        [10, 60],  
        [60, 60]  
    ],  
    5  
)
```



# Desenho de primitivas

```
pygame.draw.circle(ecrã, cor, [x, y], raio, espessura)
```

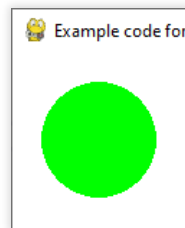
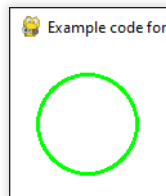


! **Espessura > 0:** Apenas o contorno (interno) nessa espessura  
**Espessura = 0 (ou parâmetro inexistente):** Preenchido

Exemplo:

```
pygame.draw.circle(  
    screen,  
    (0, 255, 0),  
    [60, 60],  
    40,  
    3  
)
```

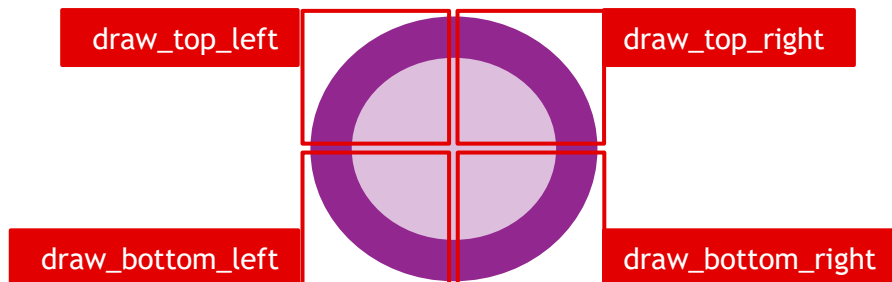
```
pygame.draw.circle(  
    screen,  
    (0, 255, 0),  
    [60, 60],  
    40  
)
```





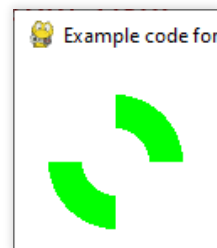
# Desenho de primitivas

- O círculo pode ainda ser desenhado por quadrantes:



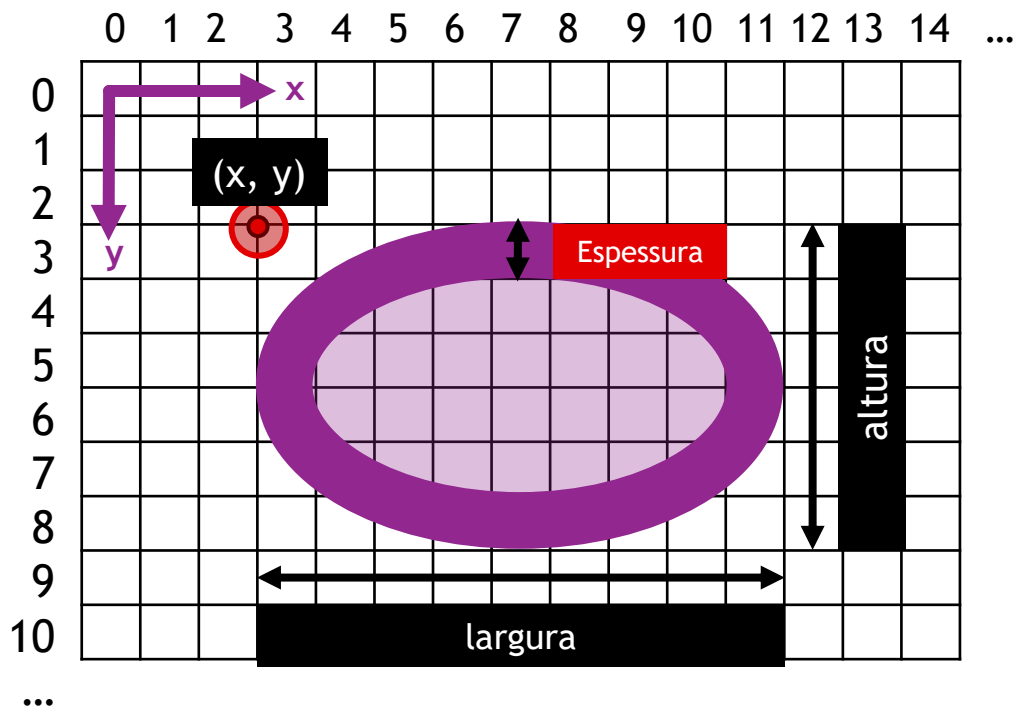
- Exemplo:

```
pygame.draw.circle(  
    Ecrã: screen,  
    Cor: (0, 255, 0),  
    [x, y]: [60, 60],  
    Raio: 40,  
    Espessura: 20,  
    Mostrar quadrantes: draw_top_right = True,  
                        draw_bottom_left = True  
)
```



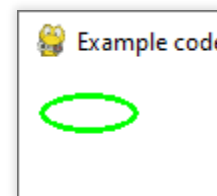
# Desenho de primitivas

`pygame.draw.ellipse(ecrã, cor, [x, y, largura, altura], espessura)`



Exemplo:

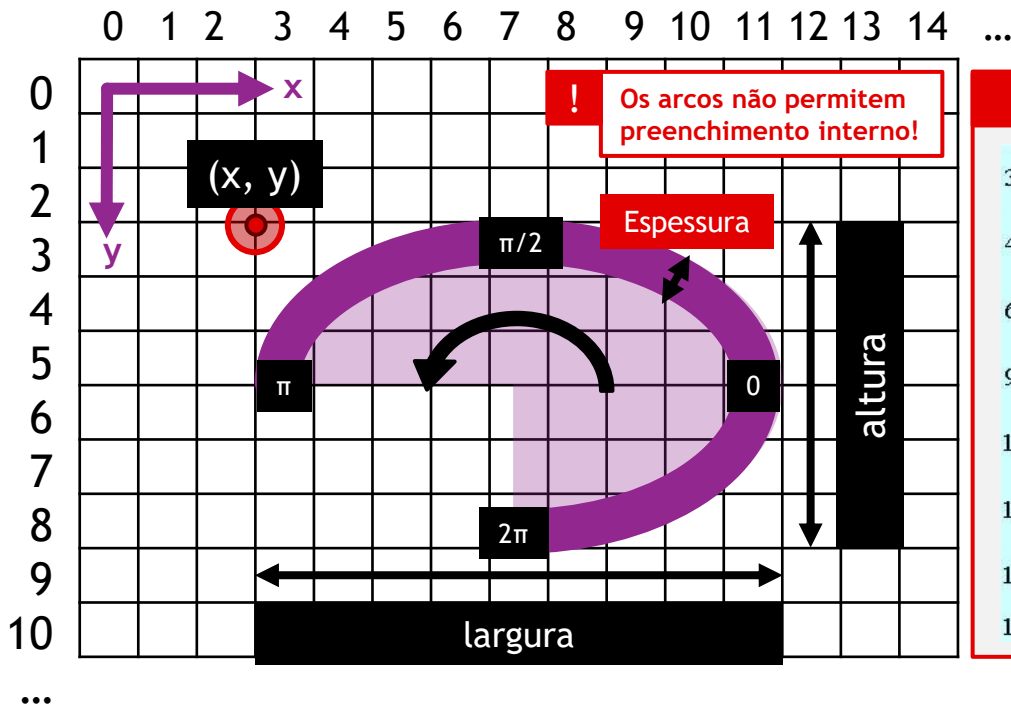
```
pygame.draw.ellipse(  
    screen,  
    (0, 255, 0),  
    [10, 10, 50, 20],  
    3  
)
```



# Desenho de primitivas

`pygame.draw.arc(ecrã, cor, [x, y, largura, altura], ângulo de início, ângulo de fim, espessura)`

! Em radianos



Graus → Radianos:

$30^\circ = \frac{\pi}{6}$	$210^\circ = \frac{7\pi}{6}$
$45^\circ = \frac{\pi}{4}$	$225^\circ = \frac{5\pi}{4}$
$60^\circ = \frac{\pi}{3}$	$240^\circ = \frac{4\pi}{3}$
$90^\circ = \frac{\pi}{2}$	$270^\circ = \frac{3\pi}{2}$
$120^\circ = \frac{2\pi}{3}$	$300^\circ = \frac{5\pi}{3}$
$135^\circ = \frac{3\pi}{4}$	$315^\circ = \frac{7\pi}{4}$
$150^\circ = \frac{5\pi}{6}$	$330^\circ = \frac{11\pi}{6}$
$180^\circ = \pi$	$360^\circ = 2\pi$

Exemplo:

! Biblioteca para usar o PI

`from math import pi`

```
pygame.draw.arc(
    screen,
    (0, 255, 0),
    [20, 20, 100, 100],
    0,
    pi/2,
    3
)
```

Example code for the



# Desenho de primitivas

- ▶ Para simplificar, poderá mandar o Python converter de graus em radianos:

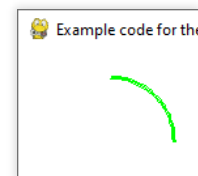
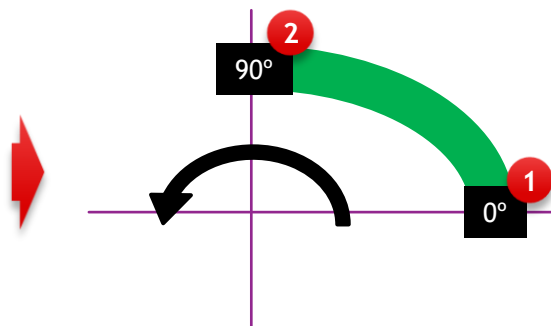
`import math`  
`math.radians(90)`

! Graus

- ▶ Exemplo:

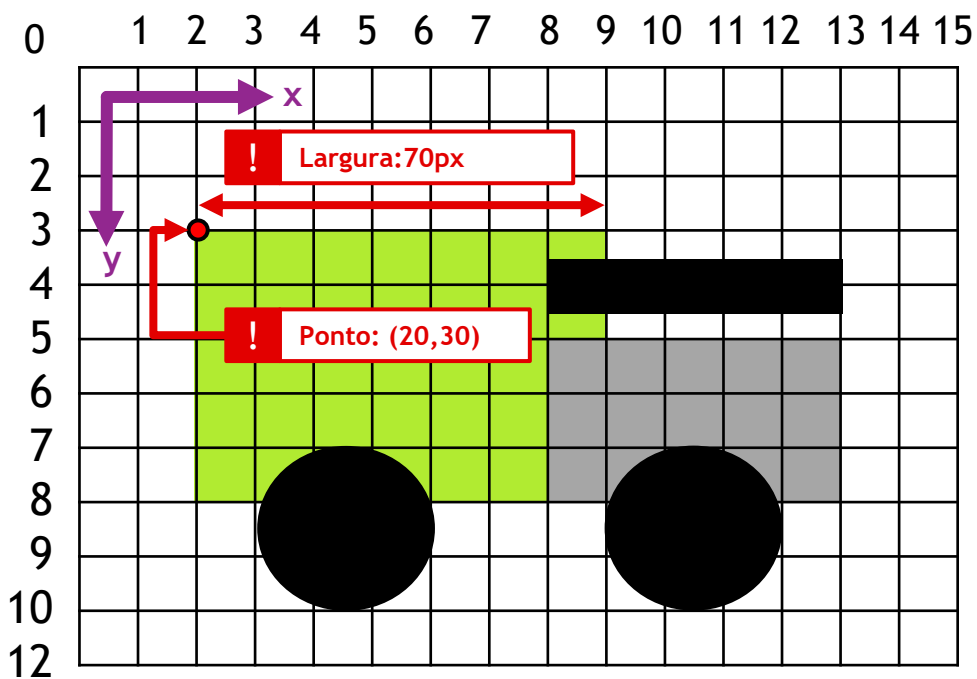
`import math`

```
pygame.draw.arc(
    screen,
    (0, 255, 0),
    [20, 20, 100, 100],
    1 0,
    2 math.radians(90),
    3
)
```



# Exercício 2

- Crie uma janela de 150x150px e desene o seguinte (multiplique as medidas por **x10**):



- Cábula:

```
screen.fill( pygame.Color( 'green' ) )
```

`pygame.draw.`

```
line(eocrã, cor, [x1, y1], [x2, y2], espessura)  
lines(eocrã, cor, fechar?, [[x1, y1], [x2, y2], [x..., y...]], espessura)  
rect(eocrã, cor, [x, y, largura, altura], espessura)  
polygon(eocrã, cor, [[x1, y1], [x2, y2], [x..., y...]], espessura)  
circle(eocrã, cor, [x, y], raio, espessura)  
ellipse(eocrã, cor, [x, y, largura, altura], espessura)  
arc(eocrã, cor, [x, y, largura, altura], ângulo de início, ângulo de fim ,  
espessura)
```

# Exercício 2

## ► Solução

```
import pygame

pygame.init()
screen = pygame.display.set_mode([150, 150])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()

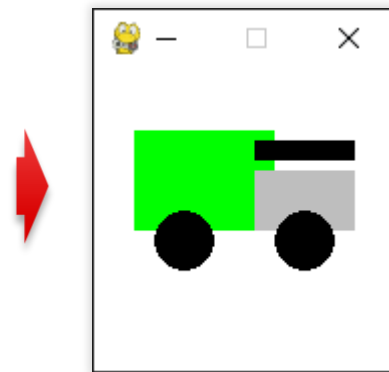
done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    screen.fill(pygame.Color("white"))

    pygame.draw.rect(screen, pygame.Color("green"), [20, 30, 70, 50])
    pygame.draw.rect(screen, pygame.Color("gray"), [80, 50, 50, 30])
    pygame.draw.rect(screen, pygame.Color("black"), [80, 35, 50, 10])
    pygame.draw.circle(screen, pygame.Color("black"), [45, 85], 15)
    pygame.draw.circle(screen, pygame.Color("black"), [105, 85], 15)

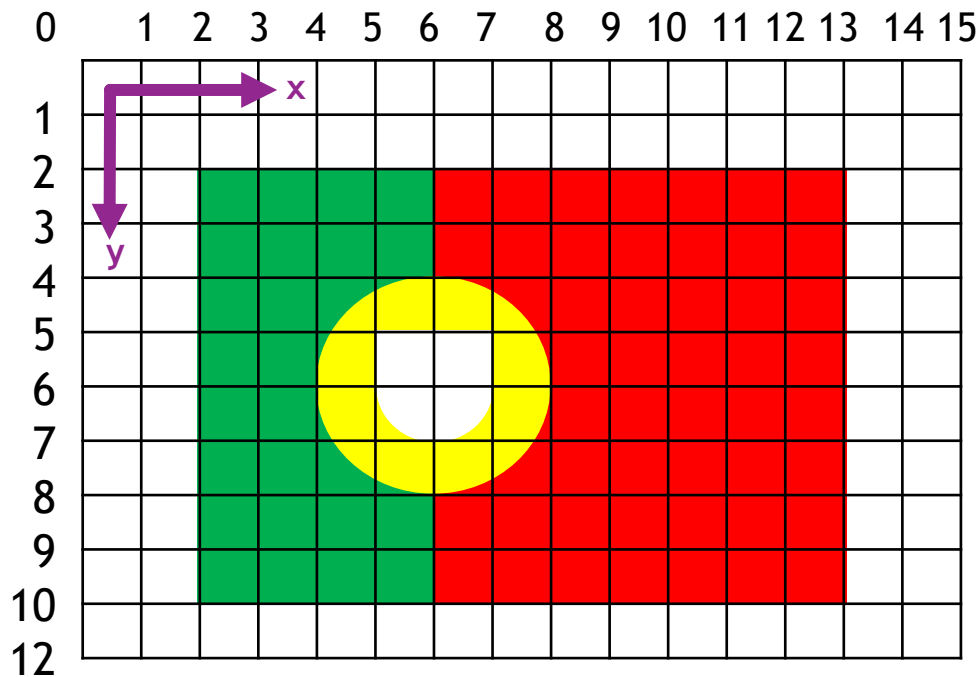
    pygame.display.update()

pygame.quit()
```



# Exercício 3

- Crie uma janela de 150x150px e desene o seguinte (multiplique as medidas por **x10**):



- Cábula:

```
screen.fill(pygame.Color('green'))
```

`pygame.draw.`

- `line(ecrã, cor, [x1, y1], [x2, y2], espessura)`
- `lines(ecrã, cor, fechado?, [[x1, y1], [x2, y2], [x..., y...]], espessura)`
- `rect(ecrã, cor, [x, y, largura, altura], espessura)`
- `polygon(ecrã, cor, [[x1, y1], [x2, y2], [x..., y...]], espessura)`
- `circle(ecrã, cor, [x, y], raio, espessura)`
- `ellipse(ecrã, cor, [x, y, largura, altura], espessura)`
- `arc(ecrã, cor, [x, y, largura, altura], ângulo de início, ângulo de fim, espessura)`

# Exercício 3

## ► Solução

```
import pygame

pygame.init()
screen = pygame.display.set_mode([150, 150])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()

done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    screen.fill(pygame.Color("white"))

    pygame.draw.rect(screen, pygame.Color("green"), [20, 20, 40, 80])
    pygame.draw.rect(screen, pygame.Color("red"), [60, 20, 70, 80])
    pygame.draw.circle(screen, pygame.Color("yellow"), [60, 60], 20)
    pygame.draw.circle(screen, pygame.Color("white"), [60, 60], 10)
    pygame.draw.rect(screen, pygame.Color("white"), [50, 50, 20, 10])

    pygame.display.update()

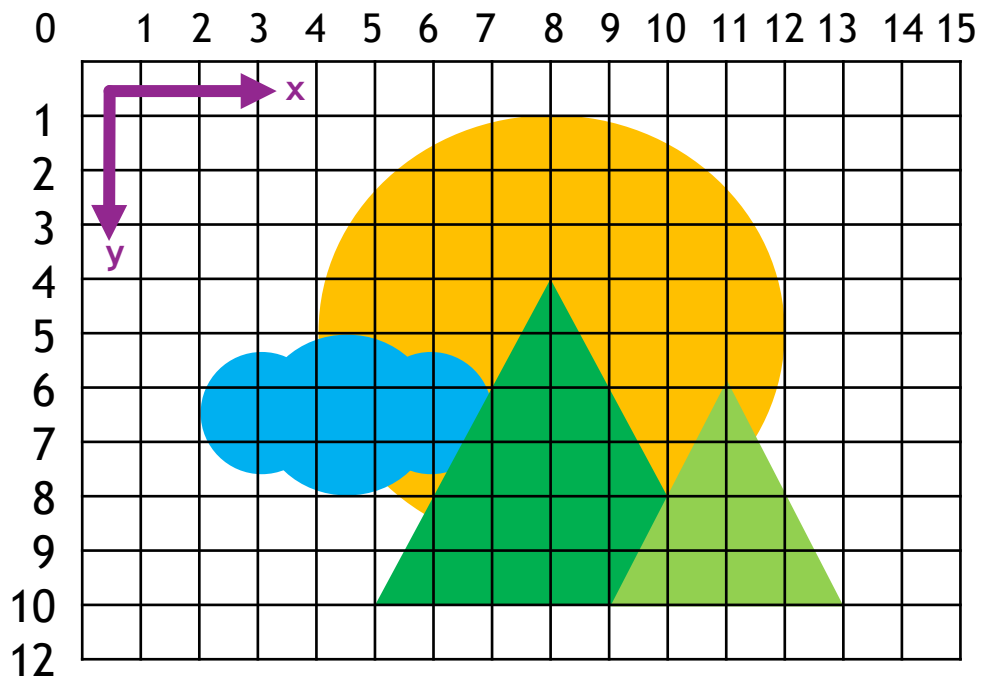
pygame.quit()
```





# Exercício 4

- Crie uma janela de 150x150px e desene o seguinte (multiplique as medidas por **x10**):



```
screen.fill(pygame.Color("#2b5ed6"))
```

#ffc000

#00b0f0

#00b050

#92d050

`pygame.draw.`

`line(ecrã, cor, [x1, y1], [x2, y2], espessura)`

`lines(ecrã, cor, fechado?, [[x1, y1], [x2, y2], [x..., y...]], espessura)`

`rect(ecrã, cor, [x, y, largura, altura], espessura)`

`polygon(ecrã, cor, [[x1, y1], [x2, y2], [x..., y...]], espessura)`

`circle(ecrã, cor, [x, y], raio, espessura)`

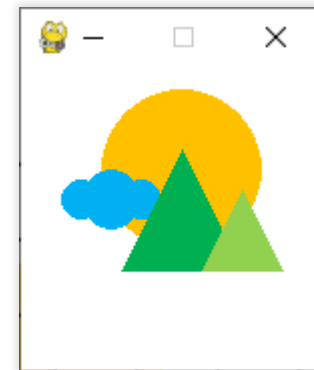
`ellipse(ecrã, cor, [x, y, largura, altura], espessura)`

`arc(ecrã, cor, [x, y, largura, altura], ângulo de início, ângulo de fim, espessura)`

# Exercício 4

## ► Solução

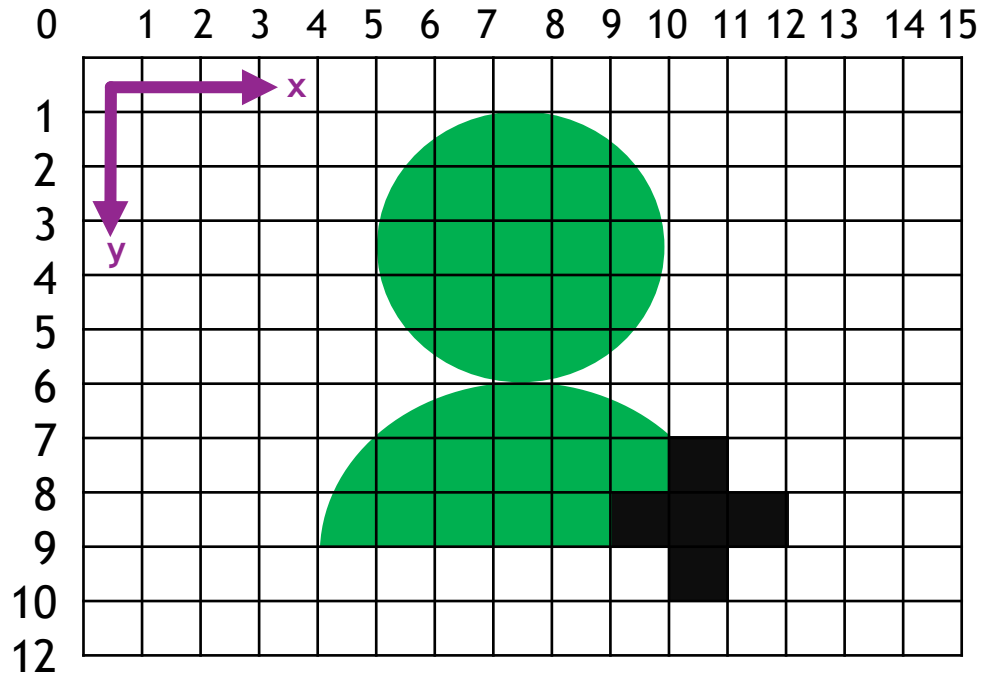
```
(...)  
pygame.draw.circle(screen, pygame.Color("#ffc000"), [80, 50], 40)  
pygame.draw.circle(screen, pygame.Color("#00b0f0"), [30, 65], 10)  
pygame.draw.circle(screen, pygame.Color("#00b0f0"), [45, 65], 15)  
pygame.draw.circle(screen, pygame.Color("#00b0f0"), [60, 65], 10)  
pygame.draw.polygon(  
    screen,  
    pygame.Color("#00b050"),  
    [  
        [80,40],  
        [50, 100],  
        [110, 100]  
    ]  
)  
pygame.draw.polygon(  
    screen,  
    pygame.Color("#92d050"),  
    [  
        [110, 60],  
        [90, 100],  
        [130, 100]  
    ]  
)  
(...)
```





## Exercício 6

- Crie uma janela de 150x150px e desenhe o seguinte (multiplique as medidas por x10):

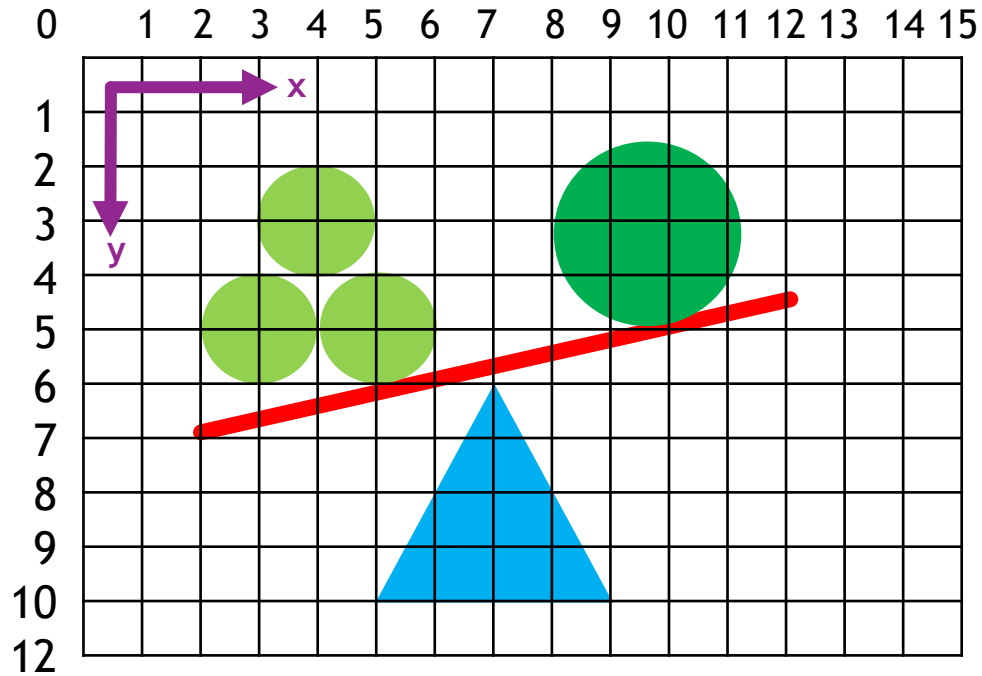






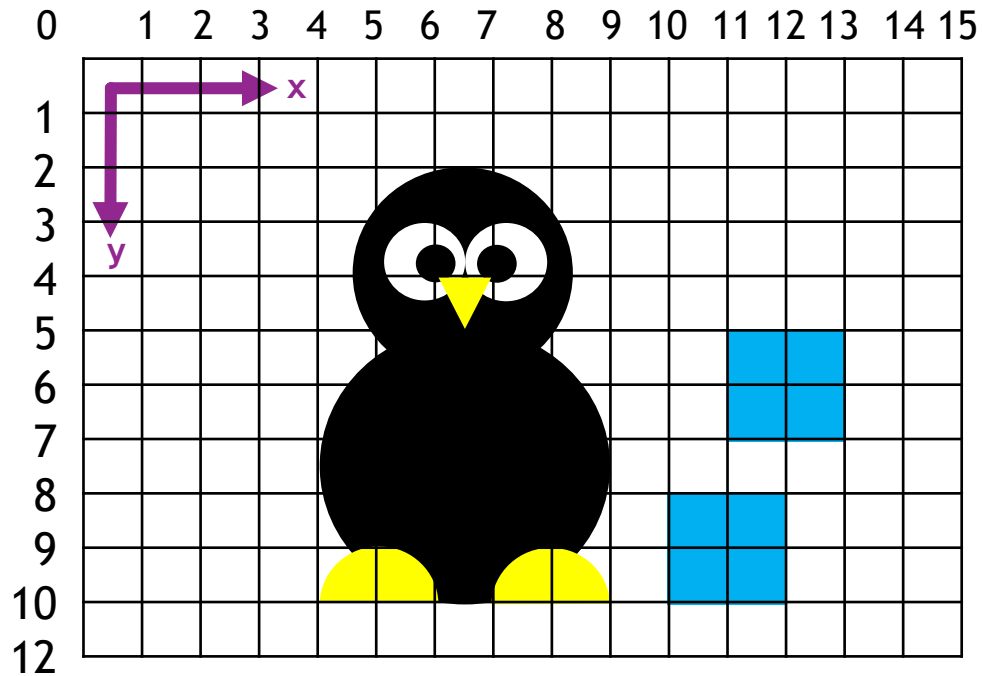
## Exercício 9

- Crie uma janela de 150x150px e desenhe o seguinte (multiplique as medidas por x10):



# Exercício 10

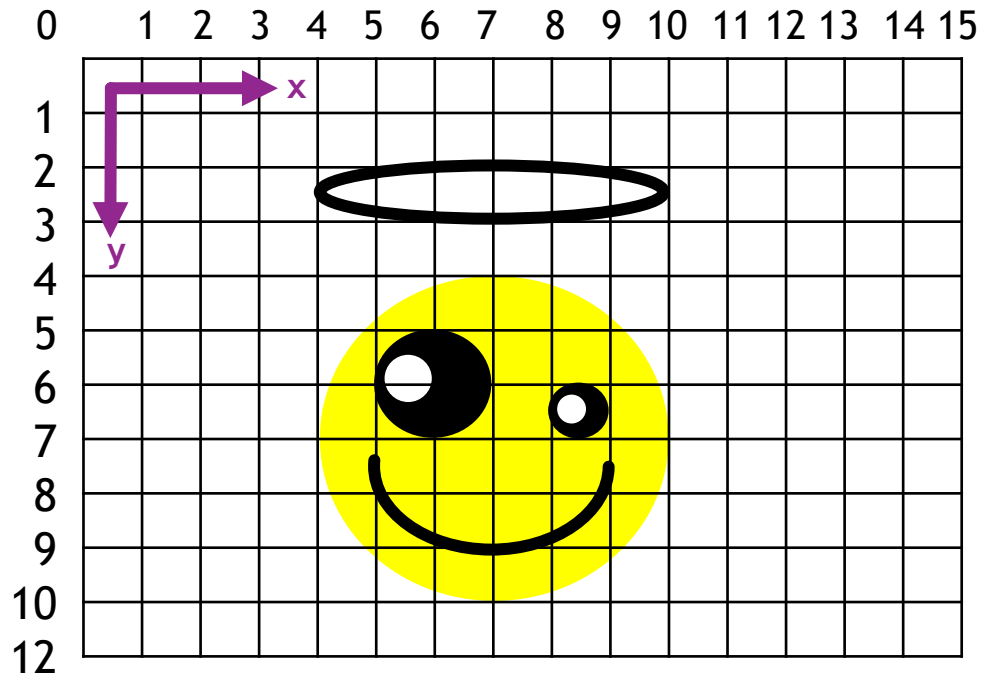
- Crie uma janela de 150x150px e desene o seguinte (multiplique as medidas por x10):





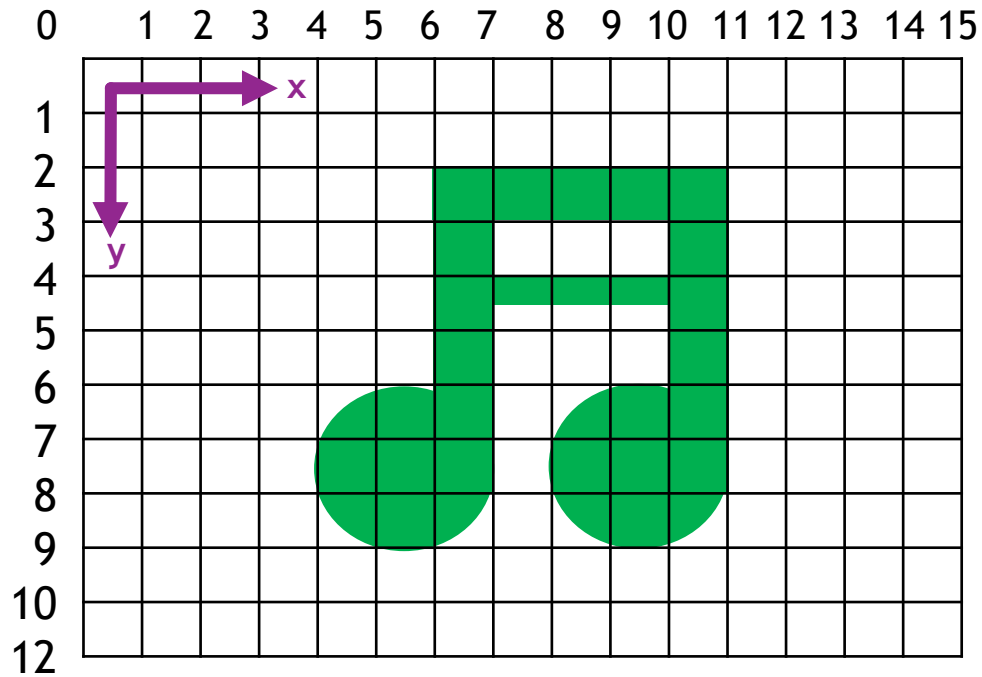
## Exercício 11

- Crie uma janela de 150x150px e desenhe o seguinte (multiplique as medidas por x10):



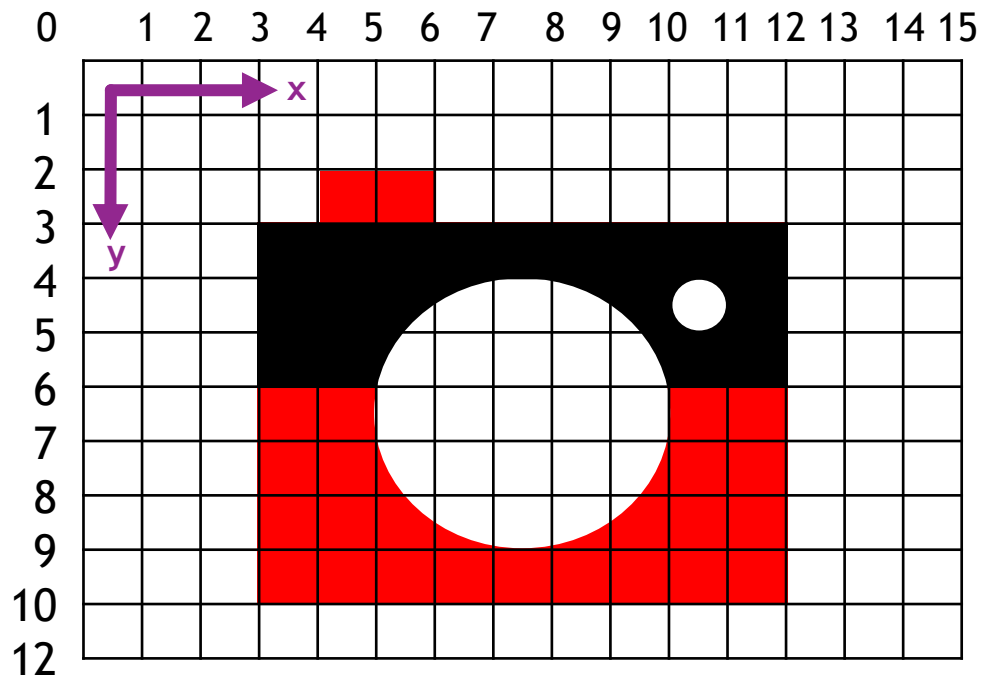
## Exercício 12

- Crie uma janela de 150x150px e desenhe o seguinte (multiplique as medidas por x10):



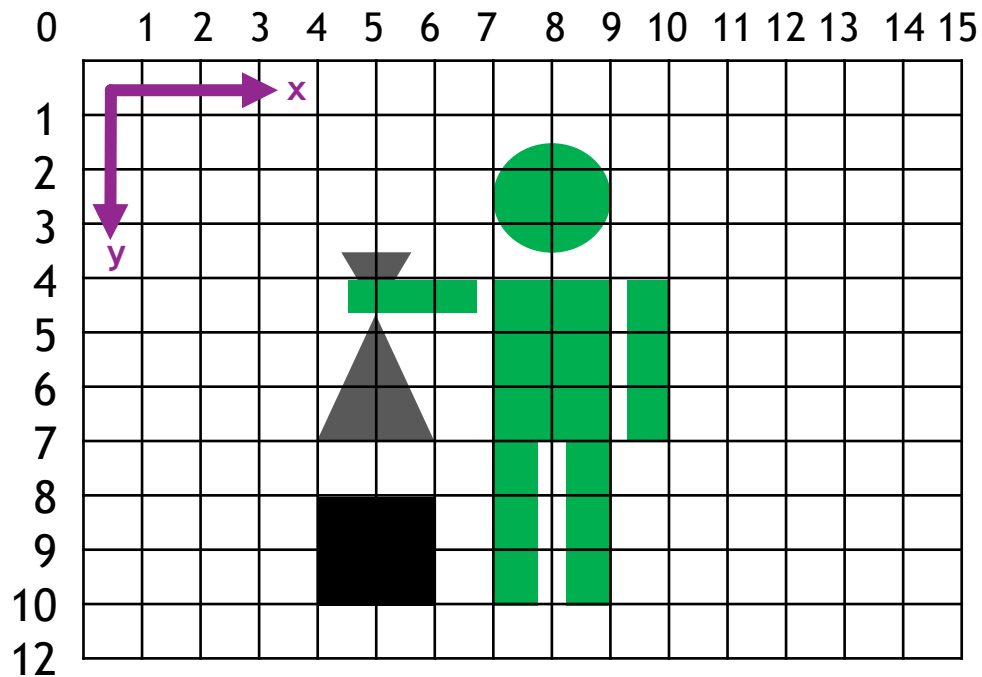
# Exercício 13

- Crie uma janela de 150x150px e desene o seguinte (multiplique as medidas por x10):



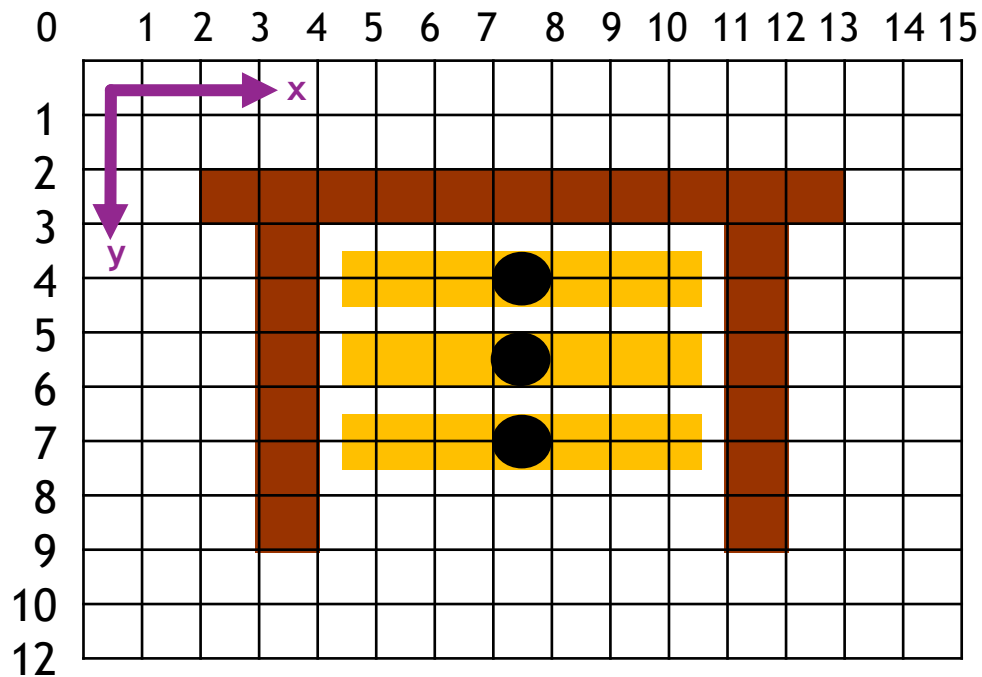
# Exercício 14

- Crie uma janela de 150x150px e desene o seguinte (multiplique as medidas por x10):



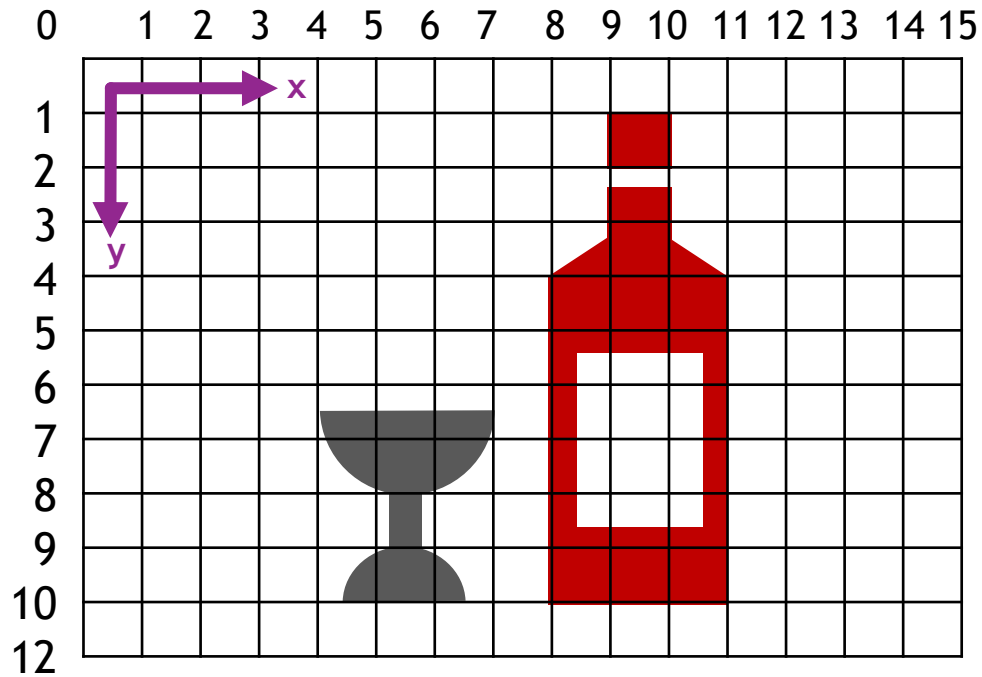
# Exercício 15

- Crie uma janela de 150x150px e desenha o seguinte (multiplique as medidas por x10):



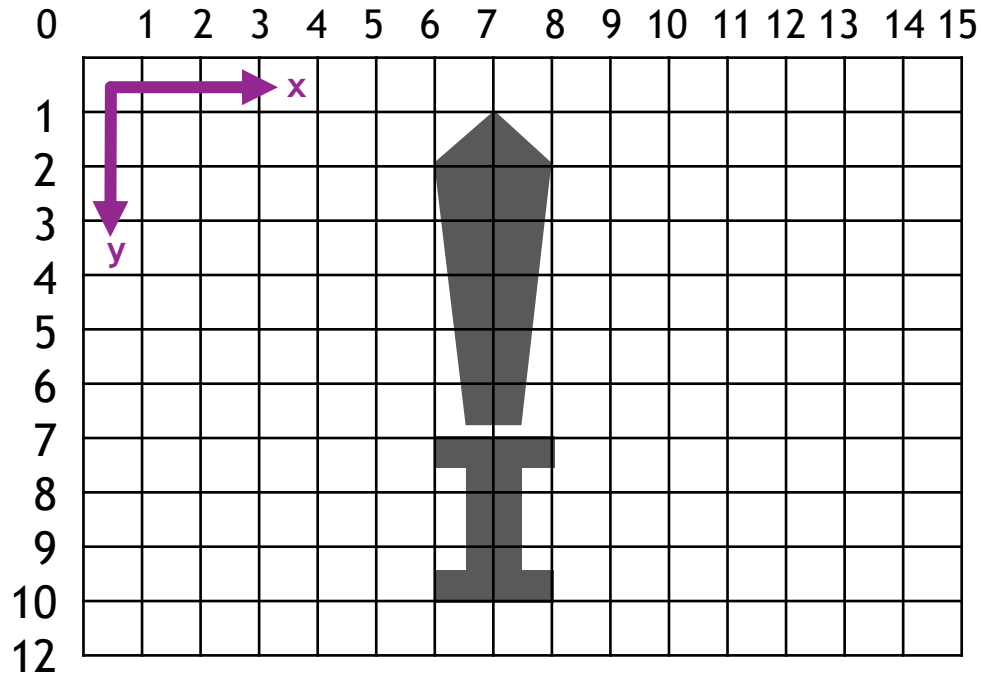
## Exercício 16

- Crie uma janela de 150x150px e desenhe o seguinte (multiplique as medidas por x10):



## Exercício 17

- Crie uma janela de 150x150px e desenhe o seguinte (multiplique as medidas por x10):

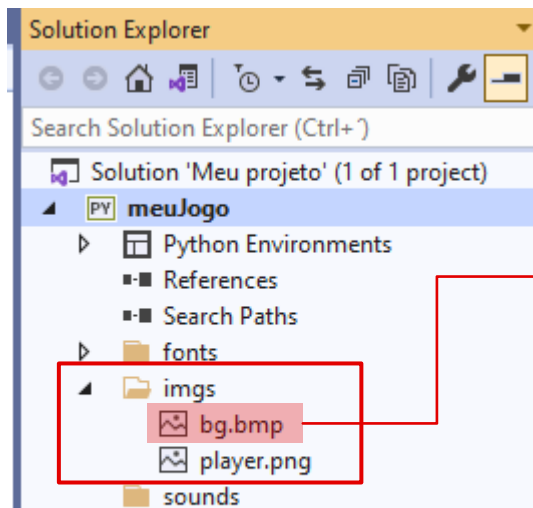


# Imagens



# Imagens

Além de se poderem desenhar formas a partir de primitivas, também é possível importar imagens para dentro do nosso projeto.



! As imagens devem ser carregadas para a pasta “imgs”, para mais tarde poderem ser importadas.

i É possível o uso de imagens PNG com fundo transparente - a transparência será respeitada pelo Python.

```
import pygame

pygame.init()
screen = pygame.display.set_mode([600, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()
```

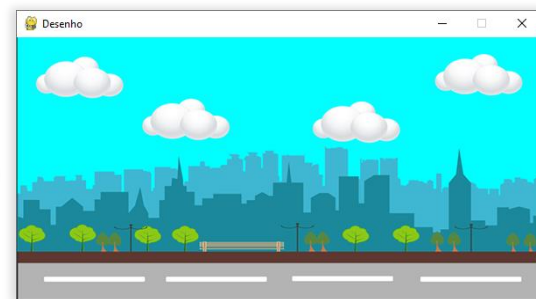
```
#ler imagem
bgImage = pygame.image.load("imgs/bg.bmp")
```

```
done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True
```

```
#aplicar imagem
screen.blit(bgImage, [0,0])
```

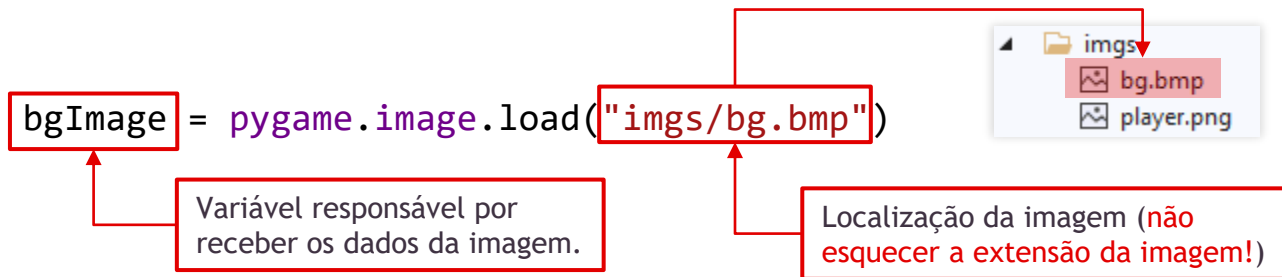
```
pygame.display.update()
pygame.quit()
```

! As imagens devem ser carregadas fora do ciclo e usadas dentro deste (para não sobrecarregar o sistema, se não a cada volta iríamos ler a imagem do disco, colocar em RAM e desenhar no ecrã).

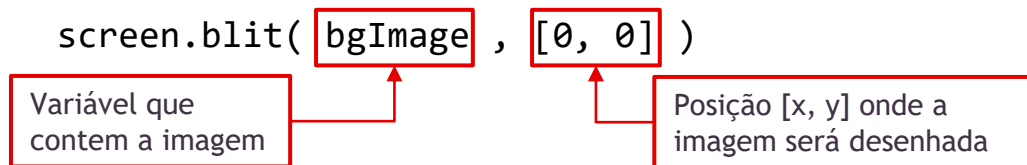


# Imagens

Leitura da imagem para memória:



Desenhar a imagem no ecrã:



# Imagens

## Exercício 18

Importe uma imagem do google (com uma dimensão rasurável) para dentro do seu projeto e aplique-a.

O tamanho da sua janela deverá contemplar o tamanho da sua imagem.

A imagem deverá estar alocada na posição 0, 0 (X, Y).

► Cácula:

```
import pygame

pygame.init()
screen = pygame.display.set_mode([600, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()

#ler imagem
bgImage = pygame.image.load("imgs/bg.bmp")

done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    #aplicar imagem
    screen.blit(bgImage, [0,0])

    pygame.display.update()
pygame.quit()
```

# Imagens

## Exercício 18 (solução)

```
import pygame

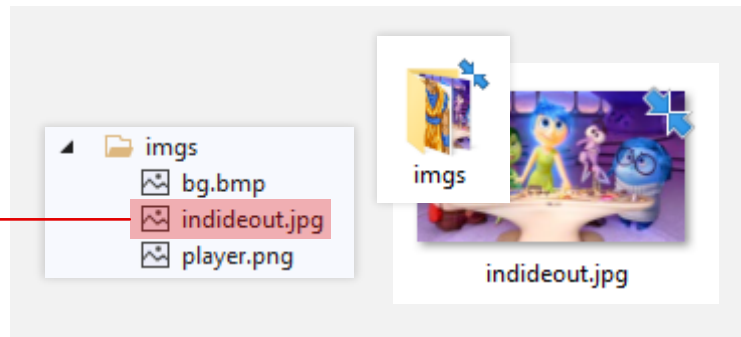
pygame.init()
screen = pygame.display.set_mode([600, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()

bgImage = pygame.image.load("imgs/indideout.jpg")

done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    screen.blit(bgImage, [0,0])

    pygame.display.update()
pygame.quit()
```



# Imagens

Por forma a não ser necessário editar o ficheiro original, é possível transformar as imagens do seguintes modos:

Inverter a orientação:

(imagem, inverter horizontal?, inverter vertical?)

```
bgImage = pygame.transform.flip(bgImage, True, False)
```

Escalar:

(imagem, (largura, altura))

```
bgImage = pygame.transform.scale(bgImage, (200, 200))
```

Rodar:

(imagem, graus)

```
bgImage = pygame.transform.rotate(bgImage, 45)
```



**Nota:** Se rodar uma imagem sem fundo transparente, as áreas libertas ficam preenchidas na cor do 1º pixel.



Mais modos de transformação disponíveis em:  
<https://www.pygame.org/docs/ref/transform.html>

# Imagens

Exemplo:

```
import pygame

pygame.init()
screen = pygame.display.set_mode([600, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()

bgImage = pygame.image.load("imgs/indideout.jpg")

bgImage = pygame.transform.flip(bgImage, True, False)
bgImage = pygame.transform.scale(bgImage, (200, 200))
bgImage = pygame.transform.rotate(bgImage, 45)

done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    screen.blit(bgImage, [0,0])

    pygame.display.update()
pygame.quit()
```

Imagem original:



Imagem invertida horizontalmente, redimensionada a 200x200 e rodada a 45°:



# Imagens

## Exercício 19

Reaproveitando o exercício anterior, inverta a sua imagem tanto na horizontal como na vertical.

(imagem, inverter horizontal?, inverter vertical?)

```
bgImage = pygame.transform.flip(bgImage, True, False)
```

### ► Cábula:

```
import pygame

pygame.init()
screen = pygame.display.set_mode([600, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()

bgImage = pygame.image.load("imgs/indideout.jpg")

bgImage = pygame.transform.flip(bgImage, True, False)
bgImage = pygame.transform.scale(bgImage, (200, 200))
bgImage = pygame.transform.rotate(bgImage, 45)

done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    screen.blit(bgImage, [0,0])

    pygame.display.update()
pygame.quit()
```

# Imagens

## Exercício 19 (solução)

```
import pygame

pygame.init()
screen = pygame.display.set_mode([600, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()

bgImage = pygame.image.load("imgs/indideout.jpg")

bgImage = pygame.transform.flip(bgImage, True, True)

done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    screen.blit(bgImage, [0,0])

    pygame.display.update()
pygame.quit()
```

Imagem original:

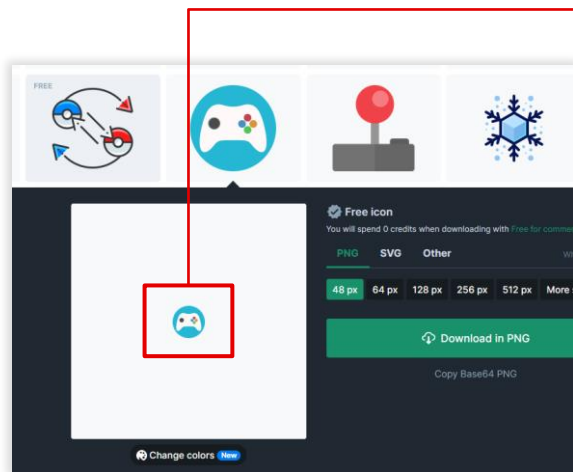




Ícone da janela

# Ícone da janela

Em vez de usarmos o ícone predefinido da janela do pygame, podemos adicionar um nosso personalizado.



Para ter acesso a ícones grátis:  
<https://www.iconfinder.com/>

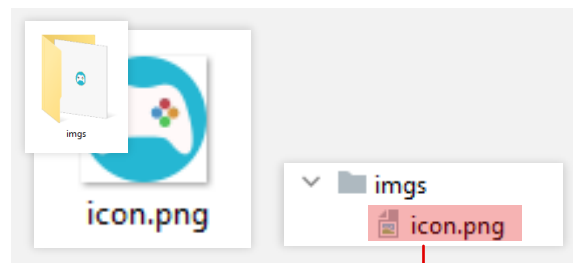


```
import pygame
from pygame.locals import *
from configs import *

pygame.init()
screen = pygame.display.set_mode([screenW, screenH])
pygame.display.set_caption("Desenho")
pygame.display.set_icon( pygame.image.load('imgs/icon.png') )
```

```
clock = pygame.time.Clock()
```

```
run = True
while run:
    (...)
```



! Adicione um ficheiro à pasta de imagens e importe-o no código.  
**Nota:** o código de inclusão do ícone deve estar fora do loop!

# Ícone da janela

## Exercício 20

Reaproveitando o exercício anterior, faça download de um ícone do site Iconfinder.



Para ter acesso a ícones grátis:  
<https://www.iconfinder.com/>

Seguidamente adicione o mesmo à janela da sua aplicação.

► Cábula:

```
import pygame
from pygame.locals import *
from configs import *

pygame.init()
screen = pygame.display.set_mode([screenW, screenH])
pygame.display.set_caption("Desenho")
pygame.display.set_icon(pygame.image.load('imgs/icon.png'))

clock = pygame.time.Clock()

run = True
while run:
    (...)
```

# Ícone da janela

## Exercício 20 (solução)

```
import pygame

pygame.init()
screen = pygame.display.set_mode([600, 300])
pygame.display.set_caption("Desenho")
pygame.display.set_icon(pygame.image.load('imgs/icon.png'))

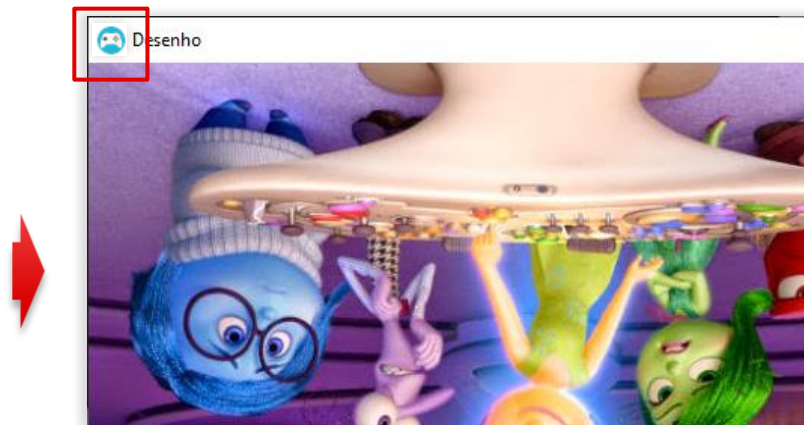
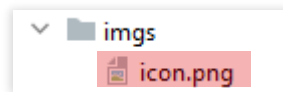
clock = pygame.time.Clock()

bgImage = pygame.image.load("imgs/indideout.jpg")
bgImage = pygame.transform.flip(bgImage, True, True)

done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    screen.blit(bgImage, [0,0])

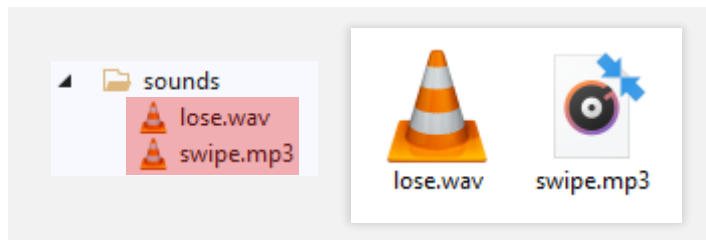
    pygame.display.update()
pygame.quit()
```



Som

# Som

É ainda possível reproduzir som, bastando para isso importar o ficheiro de som para dentro da respetiva pasta, tal como foi feito com as imagens.

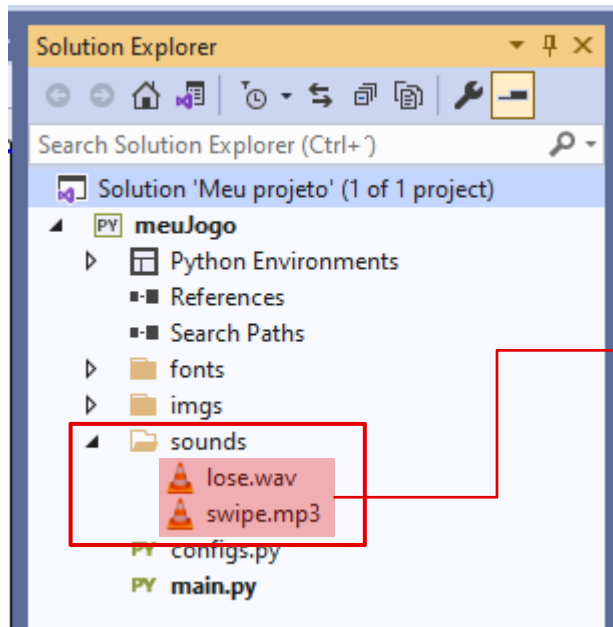


## Formatos suportados:

- **WAV** - Áudio em formato bruto e não compactado (**muito pesado**)
- **MP3** - Áudio compactado e com perda de qualidade - **este formato é limitado**, e **poderá causar instabilidade em alguns sistemas** (exemplo: Debian Linux) - considerar usar o formato OGG nestes casos.
- **OGG** - Áudio compactado e com perda de qualidade, melhor compatibilidade entre sistemas utilizando o pygame.

# Som

Para reproduzirmos áudio é necessário atender a esta sequência de código:



```
import pygame
```

```
pygame.init()
screen = pygame.display.set_mode([600, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()
```

```
pygame.mixer.init() #inicializar o mixer
```

```
intro = pygame.mixer.Sound("sounds/swipe.mp3") #Carregar o ficheiro de som
lose = pygame.mixer.Sound("sounds/lose.wav") #Carregar o ficheiro de som
```

```
intro.play() #Reproduzir swipe.mp3
lose.play() #Reproduzir lose.wav
```

```
done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True
```

```
pygame.display.update()
pygame.quit()
```

! Os ficheiros de som devem ser carregados fora do ciclo para não sobrecarregar o sistema.

! Os 2 clips serão reproduzidos praticamente ao mesmo tempo!

Nota: Dependendo das necessidades, a reprodução de áudio pode acontecer fora ou dentro do loop.

# Som

Inicialização do mixer:

```
pygame.mixer.init()
```

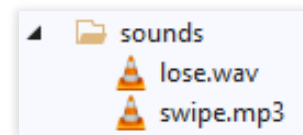
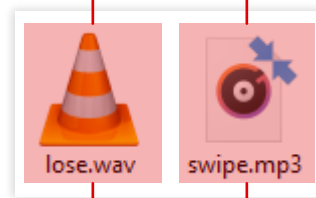
Carregar os ficheiros de som (**cada um na sua variável**)

```
lose = pygame.mixer.Sound("sounds/lose.wav")  
intro = pygame.mixer.Sound("sounds/swipe.mp3")
```

Reproduzir os sons carregados:

```
intro.play()
```

```
lose.play()
```





# Som

Outros comandos importantes para utilizar com Som:

```
pygame.mixer.init()
meuSom = pygame.mixer.Sound("sounds/swipe.mp3")
```

Comandos	Descrição
meuSom.play()	Reproduzir o som (sem parâmetros = reproduz uma vez sem qualquer manipulação)
meuSom.play(loops=2, maxtime=5000, fade_ms=500)	Reproduzir o som: <ul style="list-style-type: none"><li>• <b>loops</b> = reproduz n vezes além da inicial (ex: loops=2 irá reproduzir 3 vezes). Se definido como -1, o som irá <b>repetir infinitamente</b>.</li><li>• <b>maxtime</b> = tempo (em ms) máximo de vida da reprodução (inclui as repetições).</li><li>• <b>fade_ms</b> = Incremento gradual do som (desde o volume zero até ao total) - tempo em ms</li></ul>
(Não é necessário indicar todos os parâmetros)	
meuSom.stop()	Parar de reproduzir
meuSom.fadeout(500)	Desvanecer e parar o som (tempo em ms) no preciso momento em que ele reproduz.
meuSom.set_volume(0.2)	Definir o volume (valor entre 0.0 e 1.0)
print( meuSom.get_volume() )	Retorna o valor do volume atual
print( meuSom.get_length() )	Retorna o tamanho do som (em segundos)
print( meuSom.get_num_channels() )	Retorna a quantidade de vezes que já foi reproduzido

! 1 segundo = 1000ms



Mais detalhes em:

<https://www.pygame.org/docs/ref/mixer.html#pygame.mixer.Sound>

# Som

## Exercício 21

Faça download de dois clipes de som do site mixkit:



Download de efeitos de som grátis em:

<https://mixkit.co/free-sound-effects/>

Faça reproduzir ambos os clipes, sendo que:

- O primeiro deverá reproduzir 2 vezes
- O segundo infinitamente durante o máximo de tempo de 5 segundos.

**Nota:** Para que surta o efeito desejado, coloque os seus sons a reproduzir fora do ciclo.

► Cábula:

```
import pygame

pygame.init()
screen = pygame.display.set_mode([600, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()

pygame.mixer.init()
intro = pygame.mixer.Sound("sounds/swipe.mp3")
lose = pygame.mixer.Sound("sounds/lose.wav")

intro.play()
lose.play()

done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    pygame.display.update()
pygame.quit()
```

```
meuSom.play(loops=2, maxtime=5000, fade_ms=500)
```

# Som

## Exercício 21 (solução)

```
import pygame

pygame.init()
screen = pygame.display.set_mode([600, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()

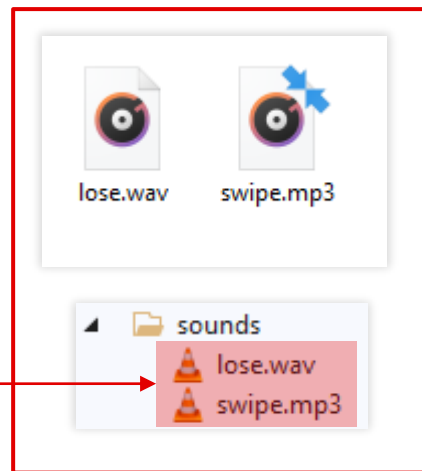
pygame.mixer.init()

lose = pygame.mixer.Sound("sounds/lose.wav")
swipe = pygame.mixer.Sound("sounds/swipe.mp3")

lose.play(loops=1)
lose.play(loops=-1, maxtime=5000)

done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    pygame.display.update()
pygame.quit()
```



# Eventos

# Eventos

O utilizador interage (disputa eventos) no sistema com o teclado e\ou rato. Relativamente ao teclado, é possível avaliar se uma determinada tecla (ou combinações das mesmas), foi premida ou solta:

```
import pygame

pygame.init()
screen = pygame.display.set_mode([300, 300])
pygame.display.set_caption("A minha aplicação")
clock = pygame.time.Clock()
```

```
done = False
while not done:
    clock.tick(5)
```

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_m:
                print("Tecla M premida")
        elif event.type == pygame.KEYUP:
            if event.key == pygame.K_m:
                print("Tecla M solta")
```

```
    pygame.display.update()
    pygame.quit()
```

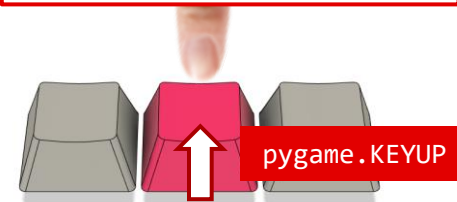
Procura no sistema se alguma tecla foi premida ou solta, para depois dar um rumo a determinadas teclas que disputaram o evento.

Já conhecemos este evento! Serve para encerrar a aplicação quando premido o "X" da janela.

Ocorre quando apertamos uma tecla



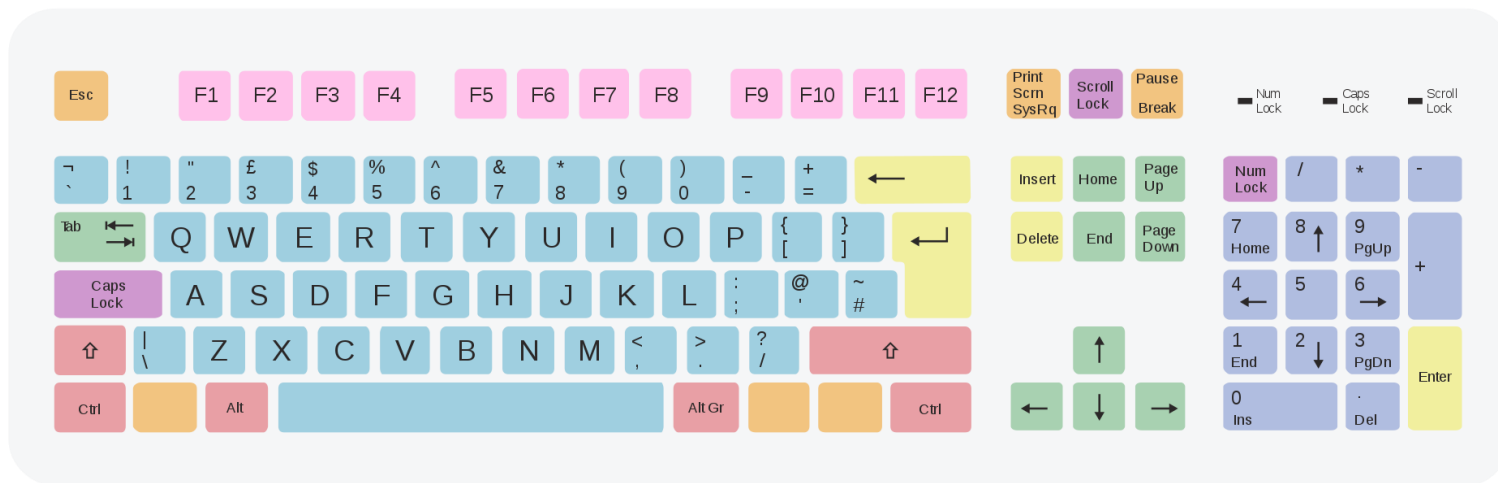
Ocorre quando libertamos uma tecla anteriormente premida



Os eventos ocorrem apenas 1 vez por cada premir (ou libertar) de tecla, sendo obrigatório repor o estado da tecla para que possa ser disputado novamente o evento.

# Eventos

Antes de avançarmos, convém entendermos as secções do teclado, uma vez que representam características diferentes:



Teclas de função



Teclas modificadoras



Teclas de sistema  
e de interface



Teclas de comando  
e de edição



Teclas de bloqueio



Teclas numéricas



Teclas alfanuméricas



Teclas de navegação

# Eventos

Teclas possíveis de serem detetadas (as mais usadas):

## Alfanuméricos:

K\_0 até K\_9  
K\_a até K\_z

## Símbolos:

K\_PLUS  
K\_COMMA  
K\_MINUS  
K\_PERIOD  
K\_SLASH  
K\_SEMICOLON  
K\_LESS  
K\_BACKSLASH

## Teclas de função:

K\_F1 até K\_F15

## Teclas do numpad:

K\_KP0 até K\_KP9  
K\_KP\_PERIOD  
K\_KP\_DIVIDE  
K\_KP\_MULTIPLY  
K\_KP\_MINUS  
K\_KP\_PLUS  
K\_KP\_ENTER  
K\_KP\_EQUALS

## Navegação:

K\_UP  
K\_DOWN  
K\_RIGHT  
K\_LEFT  
K\_HOME  
K\_END  
K\_PAGEUP  
K\_PAGEDOWN  
K\_TAB

## Bloqueio:

K\_NUMLOCK  
K\_CAPSLOCK  
K\_SCROLLOCK

## Edição e comando:

K\_INSERT  
K\_DELETE  
K\_BACKSPACE  
K\_RETURN  
K\_SPACE

## Interface:

K\_PAUSE  
K\_ESCAPE  
K\_PRINT

## Teclas modificadoras:

K\_RSHIFT } Shift  
K\_LSHIFT }

K\_RCTRL } Control  
K\_LCTRL }

K\_RALT } Alt  
K\_LALT }

K\_LSUPER } Tecla do Windows  
K\_RSUPER }



Lista completa em:

<https://www.pygame.org/docs/ref/key.html>

# Eventos

**Nota:** O uso de eventos, além de obrigar a que o estado da tecla seja sempre reposto, não nos permite analisar se combinações de teclas do tipo: “A+Espaço” estão a ocorrer.

Para resolver este problema, em vez de utilizarmos eventos, podemos pedir o estado de todas as teclas ao sistema, analisando posteriormente se uma ou mais teclas se encontram a ser premidas naquele instante.

```
import pygame

pygame.init()
screen = pygame.display.set_mode([600, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()

done = False
while not done:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    key = pygame.key.get_pressed()

    if key[pygame.K_SPACE] and key[pygame.K_a]:
        print("Tecla Espaço + A!")
    elif key[pygame.K_RIGHT]:
        print("Tecla para a direita")

    pygame.display.update()
pygame.quit()
```

Obtém o estado de todas as teclas do teclado

Avaliar o estado de teclas em particular



Combinação incomum de teclas que os eventos não iriam conseguir resolver.



# Eventos

É ainda possível com esta metodologia, verificar se estão a ocorrer combinações com teclas modificadoras (Ctrl, Alt e\ou Shift):

```
import pygame

pygame.init()
screen = pygame.display.set_mode([300, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()

done = False
while not done:
    clock.tick(5)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    key = pygame.key.get_pressed()

    if key[pygame.K_LCTRL] and key[pygame.K_a]:
        print("CTRL+A!")

    pygame.display.update()
pygame.quit()
```

"CTRL" e "A" ao mesmo tempo

## Tecclas de combinação:




K_RSHIFT	}	Shift
K_LSHIFT		
K_RCTRL	}	Control
K_LCTRL		
K_RALT	}	Alt
K_LALT		

# Eventos

## Assim, nos eventos:

- São avaliadas as teclas premidas
- Cada tecla disputa o seu evento
- Enquanto o estado da tecla não for repostado, o evento para essa mesma tecla não é disputado novamente
- É possível avaliar combinações de teclas desde que respeitem na sua combinação: CTRL, SHIFT e\ou ALT.

Para não estarmos dependentes das características acima, podemos avaliar as teclas premidas no momento, obtendo as seguintes vantagens:

- Não existe dependência na reposição do estado das teclas - permitindo-nos por exemplo: manter um personagem do jogo a mover-se para a frente enquanto é premida a tecla 
- É possível combinar teclas incomuns - permitindo-nos por exemplo: mover um personagem para a frente com a tecla  enquanto premimos a tecla de  para saltar.

# Eventos

## Exercício 22

Crie um programa (**tirando recurso dos eventos**) que indique no seu ecrã qual das setas do seu teclado foi premida (cima, baixo, esquerda ou direita) - se qualquer outra tecla premida, indique: Tecla desconhecida - esta informação deverá aparecer com o texto a preto.

**Nota:** Não se esqueça de limpar o texto da tecla anterior sempre que queira apresentar uma nova tecla (de outra forma irá ver uma sobreposição de pixéis).

(Tipo de letra, tamanho)

! Escrever no ecrã

```
font = pygame.font.Font(None, 150)

(Texto, Anti-aliasing?, cor da letra, cor de fundo)
texto = font.render( "Olá mundo" , False , (144,144,144) , (33,44,155) )

(Desenho do texto, posição: [X, Y])
screen.blit( texto , [5, 5] )
```

## ► Cábula:

```
import pygame

pygame.init()
screen = pygame.display.set_mode([300, 300])
pygame.display.set_caption("A minha aplicação")
clock = pygame.time.Clock()

done = False
while not done:
    clock.tick(5)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_m:
                print("Tecla M premida")
        elif event.type == pygame.KEYUP:
            if event.key == pygame.K_m:
                print("Tecla M solta")

    pygame.display.update()
pygame.quit()
```

! Pintar o fundo

```
screen.fill(pygame.Color("white"))
```

! Teclas de navegação

```
K_UP
K_DOWN
K_RIGHT
K_LEFT
```

# Eventos

## Exercício 22 (solução)

```
import pygame

pygame.init()
screen = pygame.display.set_mode([300, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()
```

Preparar o tipo de letra e  
pintar o ecrã a branco

```
font = pygame.font.Font(None, 20)
screen.fill(pygame.Color("white"))
```

```
done = False
while not done:
    clock.tick(10)
```

Limpar tudo o que existe desenhado (por  
cada vez que uma tecla é premida).

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        done=True
    elif event.type == pygame.KEYDOWN:
```

Variável que recebe  
o texto a apresentar  
mais à frente

```
        screen.fill(pygame.Color("white"))
        txt = ""
```

```
        if event.key == pygame.K_UP:
            txt = "Seta para cima"
        elif event.key == pygame.K_DOWN:
            txt = "Seta para baixo"
        elif event.key == pygame.K_LEFT:
            txt = "Seta para esquerda"
        elif event.key == pygame.K_RIGHT:
            txt = "Seta para direita"
        else:
            txt = "Tecla desconhecida"
```

Avaliação  
das teclas  
premidas.

```
        texto = font.render(txt, True, (0, 0, 0))
        screen.blit(texto, [5, 5])
```

Desenhar  
o texto  
no ecrã

```
    pygame.display.update()
pygame.quit()
```

i Ao premir uma tecla, é  
apresentado no ecrã a  
informação sobre a mesma.  
Como se trata de um  
evento, se nenhuma tecla  
for premida, esta estrutura  
de código não será  
executada, mantendo assim  
sempre o texto da última  
premida.



# Eventos

## Exercício 23

Reaproveitando o exercício anterior, desenvolva a mesma ideia, mas em vez de usar eventos, veja de forma geral quais as teclas que estão a ser premidas no momento.

**Nota:** Não se esqueça que a forma como deteta as teclas mudou:

**Antes:**

```
if event.key == pygame.K_UP:  
    txt = "Seta para cima"
```

**Agora:**

```
if key[pygame.K_UP]:  
    txt = "Seta para cima"
```

(Siga o exemplo do lado direito)

► Cábula:

```
import pygame  
  
pygame.init()  
screen = pygame.display.set_mode([600, 300])  
pygame.display.set_caption("Desenho")  
clock = pygame.time.Clock()  
  
done = False  
while not done:  
    clock.tick(10)  
    for event in pygame.event.get():  
        if event.type == pygame.QUIT:  
            done=True  
  
    key = pygame.key.get_pressed()  
  
    if key[pygame.K_LEFT]:  
        print("Tecla para a esquerda")  
    elif key[pygame.K_RIGHT]:  
        print("Tecla para a direita")  
  
    pygame.display.update()  
pygame.quit()
```

# Eventos

## Exercício 23 (solução)

i

Como não se trata de um evento, este código será sempre realizado, atendendo sempre ao estado das teclas atualmente premidas. Assim, caso se trate de uma tecla que não as setas ou não exista uma tecla premida, aparecerá sempre: “Desconhecida”.

```
import pygame

pygame.init()
screen = pygame.display.set_mode([300, 300])
pygame.display.set_caption("Desenho")
clock = pygame.time.Clock()
```

```
font = pygame.font.Font(None, 20)
```

```
done = False
while not done:
    clock.tick(5)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True
```

Limpar tudo o que existe desenhado antes de apresentar a tecla premida.

```
screen.fill(pygame.Color("white"))
```

```
key = pygame.key.get_pressed()
txt = ""
if key[pygame.K_UP]:
    txt = "Seta para cima"
elif key[pygame.K_DOWN]:
    txt = "Seta para baixo"
elif key[pygame.K_LEFT]:
    txt = "Seta para esquerda"
elif key[pygame.K_RIGHT]:
    txt = "Seta para direita"
else:
    txt = "Desconhecida"
```

Avaliação das teclas premidas.

Apresentar resultados

```
texto = font.render(txt, True, (0, 0, 0))
screen.blit(texto, [5, 5])
```

```
pygame.display.update()
pygame.quit()
```



# Organização do jogo

# Organização do jogo

- Como já foi mencionado, o seu jogo será gerado dentro de um ciclo, que apenas deve terminar quando o jogo acabar.
- Dentro do ciclo deverão acontecer as seguintes ordens de trabalho:

```
run = True  
while run:
```

- 1 Gerir todos os eventos (janela, teclado, rato)
- 2 Atualizar o estado do jogo (baseado nos eventos?)
- 3 Desenhar o estado do jogo no ecrã

Gerir eventos e  
atualizar estados



Desenhar  
os estados



**Eventos:** Quando o utilizador interage com o sistema

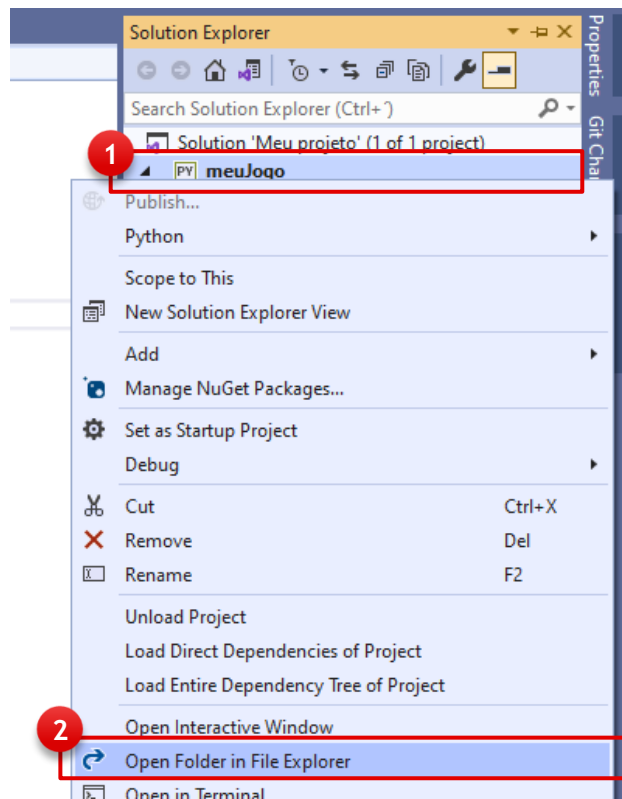
**Atualizar estado do jogo:** Mostrar, ocultar, remover, adicionar, etc. elementos ao jogo

**Desenhar o estado do jogo:** Apresentar visualmente as alterações respeitantes ao passo anterior

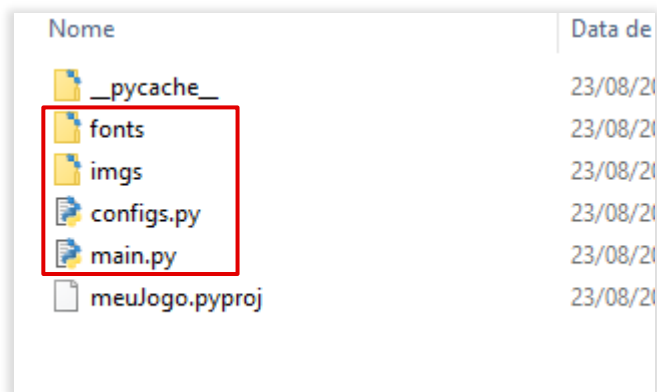


Criar um executável

# Criar um executável

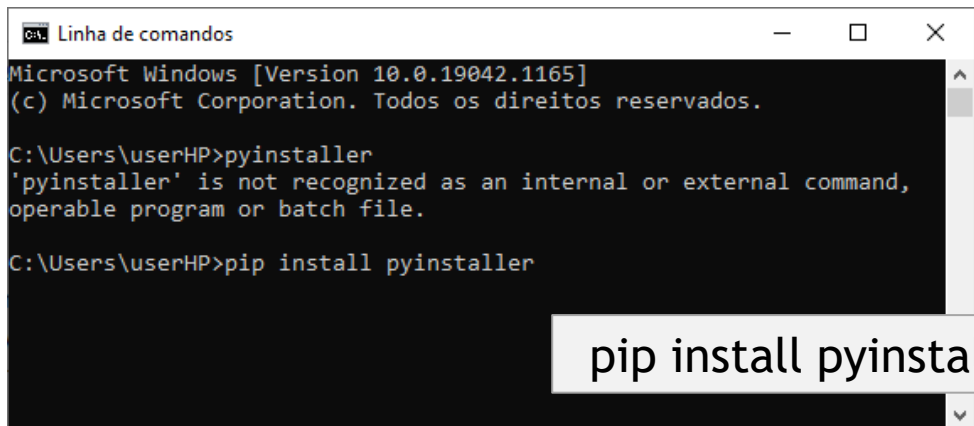


No explorador de ficheiros do Visual Studio, botão direito sobre o projeto para encontrarmos a pasta do mesmo



# Criar um executável

- Abrir a consola e correr o comando:



A screenshot of a Windows Command Prompt window titled "Linha de comandos". The window shows the following text:

```
Microsoft Windows [Version 10.0.19042.1165]  
(c) Microsoft Corporation. Todos os direitos reservados.  
  
C:\Users\userHP>pyinstaller  
'pyinstaller' is not recognized as an internal or external command,  
operable program or batch file.  
  
C:\Users\userHP>pip install pyinstaller
```

A callout box with a light gray background and a black border is positioned over the bottom right of the command prompt window, containing the text "pip install pyinstaller".

# Criar um executável

- Apontar na consola o diretório para a pasta do nosso projeto (aberto anteriormente)



```
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>cd C:\Users\userHP\source\repos\Meu projeto\Meu projeto
```

Exemplo:

```
cd c:\Users\{utilizador}\source\repos\Meu projeto\Meu projeto
```

# Criar um executável

- Criar o ficheiro EXE:

```
cmd: Linha de comandos
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>
C:\Users\userHP>cd C:\Users\
C:\Users\userHP\source\repos\Meu projeto\Meu projeto>pyinstaller main.py --onefile --noconsole
```

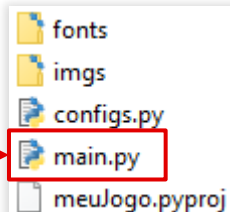
pyinstaller [ficheiro Python principal] --onefile --noconsole

--onefile: Gerar apenas um ficheiro

--noconsole: Não apresentar a consola com a abertura do executável

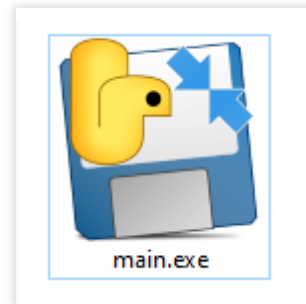
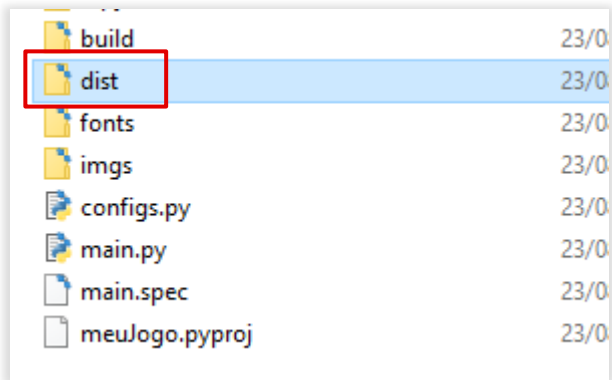
Exemplo:

Pyinstaller **main.py** --onefile --noconsole



# Criar um executável

- Com este processo foi gerada uma nova pasta chamada **dist** (distribution) com o ficheiro executável lá dentro:



# Criar um executável

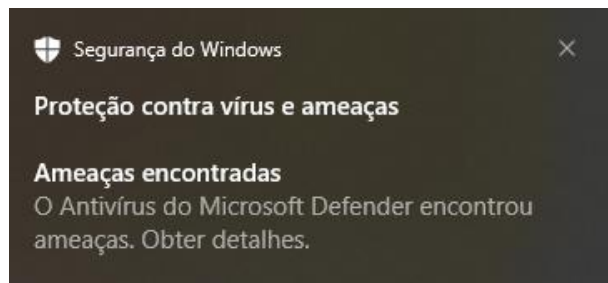
- Note que o seu antivírus poderá detetar este ficheiro como sendo um **Trojan Horse**:

The screenshot shows the VirusTotal interface for a file. At the top, a red circle indicates that 21 out of 68 security vendors have flagged the file as malicious. A red box highlights the text 'Resultados do VirusTotal'. The file's SHA-256 hash is displayed, along with its size (9.81 MB) and the upload date (2021-08-23 01:15:17 UTC). Below this, there are tabs for '64bits', 'assembly', 'invalid-rich-pe-linker-version', 'overlay', and 'peexe'. The 'DETECTION' tab is selected, showing a table of results from various antivirus engines.

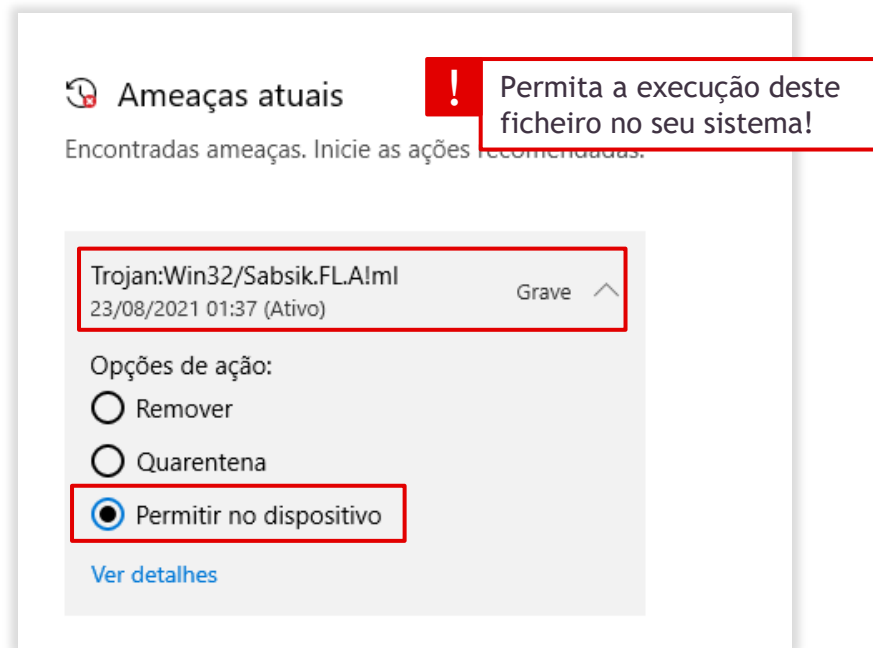
DETECTION	DETAILS	BEHAVIOR	COMMUNITY
Ad-Aware	ⓘ Gen:Variant.Bulz.604855	ALYac	ⓘ Gen:Variant.Bulz.604855
Antiy-AVL	ⓘ Trojan/Generic.ASMalwS.34493BB	SecureAge APEX	ⓘ Malicious
Arcabit	ⓘ Trojan.Bulz.D93AB7	Avast	ⓘ Win64:Trojan-gen
AVG	ⓘ Win64:Trojan-gen	BitDefender	ⓘ Gen:Variant.Bulz.604855
Cynet	ⓘ Malicious (score: 100)	Cyren	ⓘ W64/Bulz.BI.gen!Eldorado
Emsisoft	ⓘ Gen:Variant.Bulz.604855 (B)	eScan	ⓘ Gen:Variant.Bulz.604855

# Criar um executável

- Poderá desativar o mesmo de ser detetado como vírus no seu computador (note que ainda assim, no computador de outros utilizadores continuará a ser detetado como falso positivo).
- Por exemplo no caso do Windows Defender:



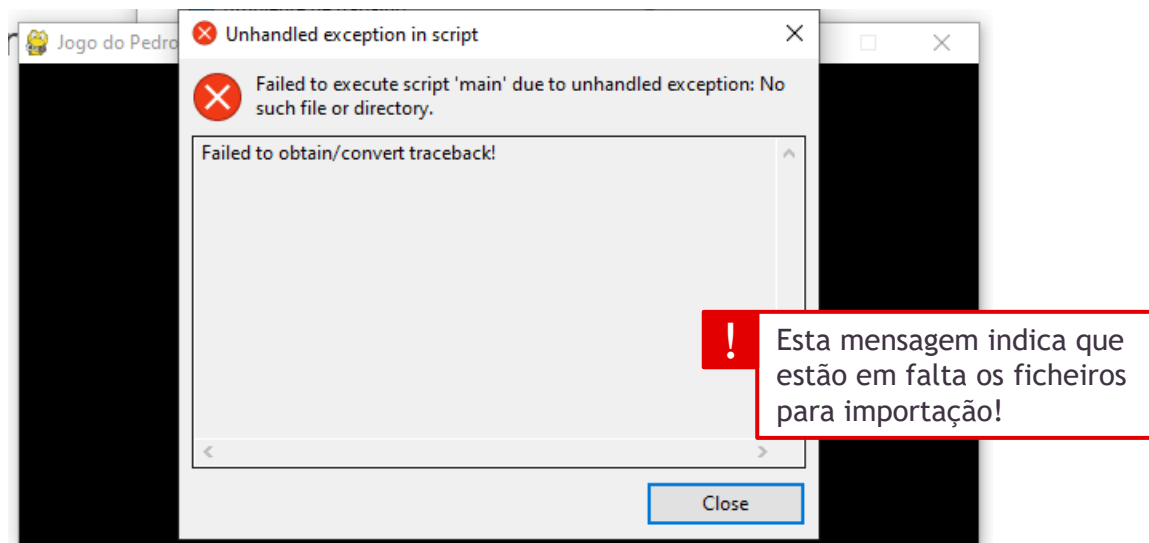
! A razão deve-se ao facto de ter sido gerado um ficheiro com múltiplos outros dentro (comportamento semelhante a um Trojan Horse).





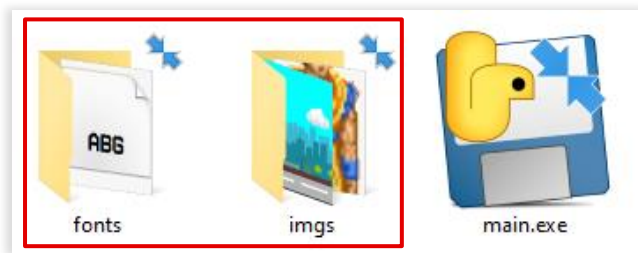
# Criar um executável

- **Importante:** Se o seu projeto necessitar de importar imagens, sons, tipos de letra, etc. para funcionar, se tentar abrir o ficheiro EXE neste momento, irá aparecer a seguinte mensagem:



# Criar um executável

- Para que funcione, necessita de copiar todos os elementos que são importáveis dentro do seu jogo (imagens, sons, tipos de letra, etc.).
- (Não necessita de copiar os ficheiros .py)



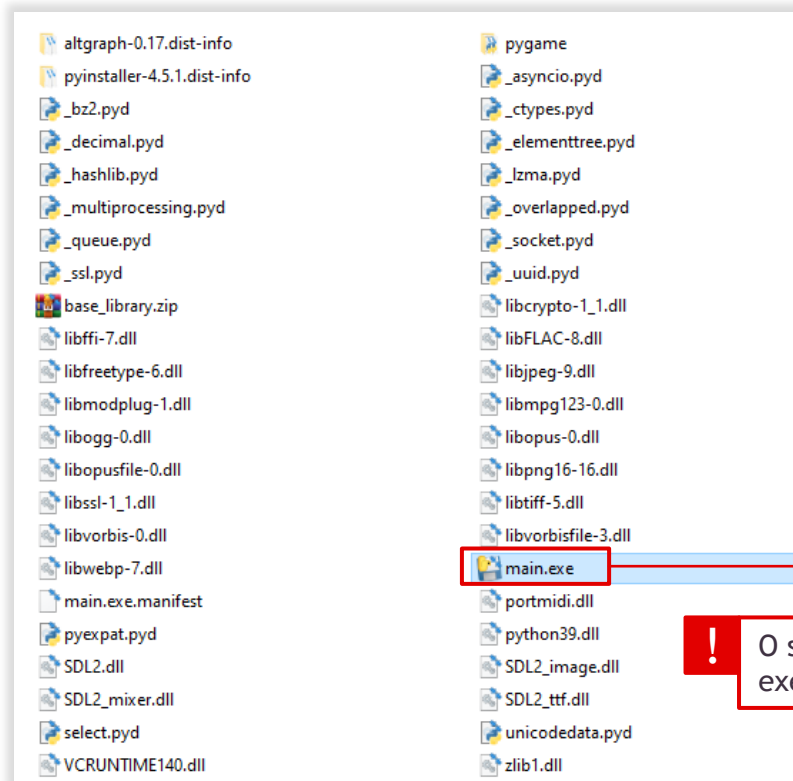
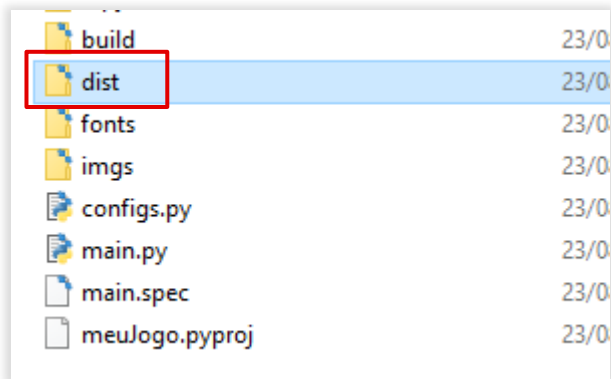
Para partilhar o seu jogo com outra pessoa apenas necessita de enviar estes ficheiros. (A outra pessoa necessita de ter o interpretador de Python instalado)





# Criar um executável

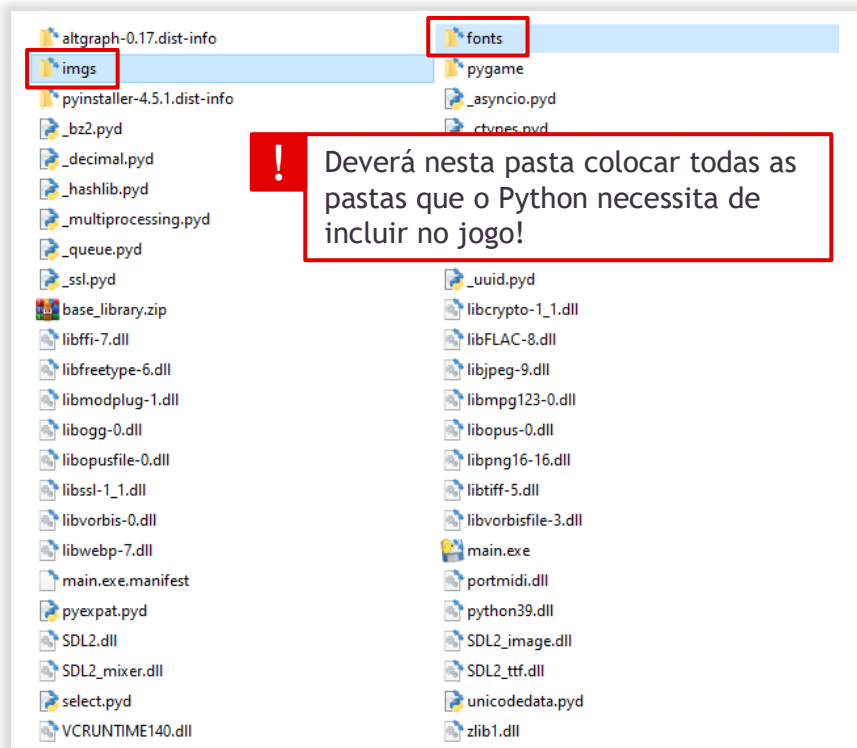
- Ao executar este processo, a pasta “dist” passará a ter os seguintes ficheiros:



! O seu ficheiro executável!

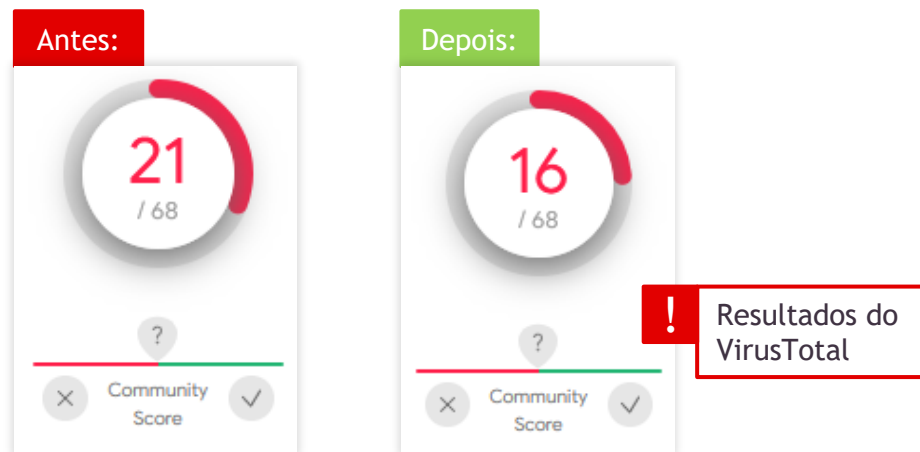
# Criar um executável

- Não se esqueça de copiar também as pastas dos conteúdos a importar pelo seu jogo!



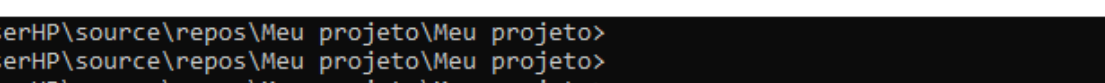
# Criar um executável

- Note que **ainda poderão haver alguns antivírus que não gostam deste ficheiro!** Mas serão menos em comparação com o processo anterior:



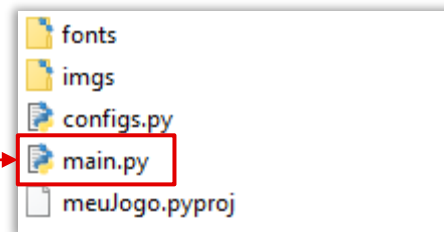
# Criar um executável

- Se ao construir o seu executável a permitir mostrar a consola, a deteção de falsos positivos nos antivírus será ainda inferior:



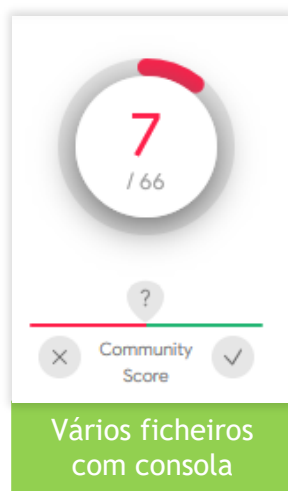
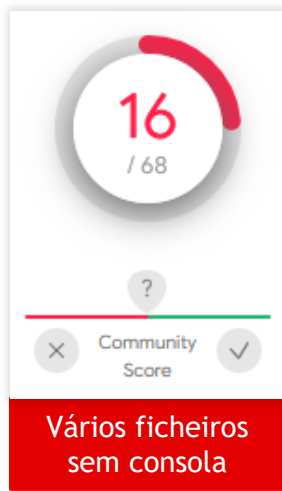
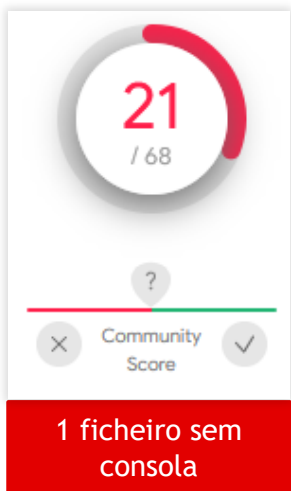
pyinstaller [ficheiro Python principal]

Exemplo:

Pyinstaller **main.py**

# Criar um executável

- Resultado comparativo:

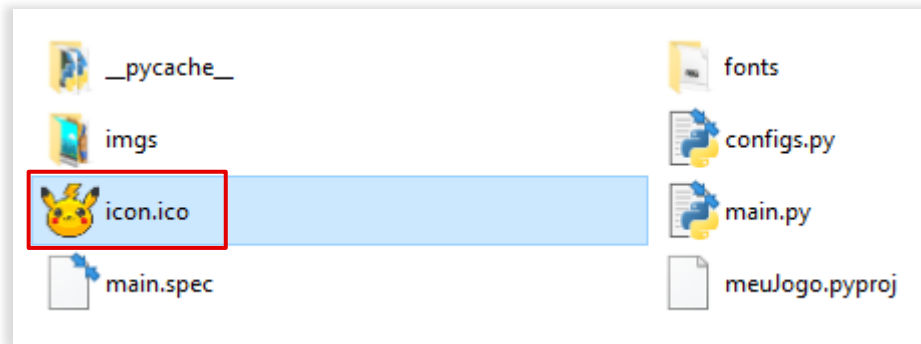




Atribuir um ícone

# Atribuir um ícone

- Se pretender configurar o ícone da aplicação, primeiramente crie um ficheiro .ICO e guarde-o na raiz da pasta do seu projeto.
- Caso pretenda converter uma imagem em ícone, use este website: <https://convertico.com/>



# Atribuir um ícone

- Na consola faça gerar um Executável mas com a indicação de onde se encontra o ficheiro de ícone.

```
Ca\ Linha de comandos
9452 INFO: Copying icons from ['icon.ico']
9531 INFO: Writing RT_GROUP_ICON 0 resource with 20 bytes
9531 INFO: Writing RT_ICON 1 resource with 20 bytes
9540 INFO: Appending icon to resource
10210 INFO: Building COLLECT
10213 INFO: checking COLLECT
10214 INFO: Building COLLECT because COLLECT-00.toc is non existent
10215 INFO: Building COLLECT COLLECT-00.toc
10697 INFO: Building COLLECT COLLECT-00.toc completed successfully.

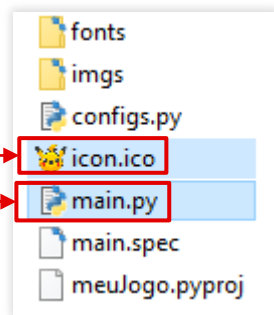
C:\Users\userHP\source\repos\Meu projeto\Meu projeto>Pyinstaller main.py --icon=icon.ico --onefile --noconsole
```

Pyinstaller [ficheiro Python principal] --icon=[ícone.ico] --onefile --noconsole

! Poderá usar também descartar os parâmetros **-onefile** e **--noconsole**

Exemplo:

Pyinstaller **main.py** --icon=**icon.ico** --onefile --noconsole



Resultado final:

