# Importing modules + pygame intro

CHAPTER 3 (P.49-53)

*(THE BOOK DOESN'T HAVE PYGAME MATERIAL☹ )*

# modules

- **core language** vs modules.
  - *millions* of **modules**.
    - Networking
    - Graphics
    - computer vision
    - ...
- To use these, you need to **import** them (usually at the top of your script)

```
import math
import pygame
```

# modules, cont.

- Sources of modules:
  1. shipped with python distribution
  2. downloaded modules
  3. Modules that you write yourself.
- Accessing module contents:

*module_name.item_in_the_module.*

- Example

# random Module

- random.randint(*low_num, high_num*)
- random.uniform(*low_num, high_num)*
- random.gauss(*center_num, std_dev*)
- A "rookie" mistake:
  - Often, people see a line like this:

    x = random.randint(5, 10)
  - And think we're making x a variable which changes all the time – we're not.
    - x gets its value (which will be different on each run)
    - But…it never changes, until we assign a new value to it.

```
print(x)                # Let's say it was 7 this time
print(x)                # still 7
print(x)                # STILL 7
…
```

# time module

- time.sleep(*num_seconds*)
- time.time()
  - Example use


# Part I

Reference: **www.pygame.org**

# What is it?

A multimedia python library

- Window Management
- Graphics
  - geometric shapes
  - bitmaps (sprites)
- Input
  - Mouse
  - Keyboard
  - Gamepad / Joystick
- Sound
  - SoundFX
  - Music

# Overview

- Pygame (sub-)module organization
  - Connection to documentation
- Pygame <= SDL <= [DirectX / OpenGL / ...]

# Surfaces

- A 2d array of (RGB) pixels
  - An image (sprite)
  - Rendered (at run-time) text
  - A mini-map
  - A HUD
  - An **interface to the window's contents**.
  - In pygame, Surfaces are **objects**.
    - Look up the methods in the "Surfaces" section of pygame docs.
- Double Buffering
  - Monitor refreshes
  - Life without D.B.
  - D.B. in pygame

# Examples

- [Basic window]
- [Drawing]
  - [Look at the pygame docs]

# Sprites / Images

- Images (in pygame) are Surface objects
- File formats loadable by pygame:
  - JPG, BMP, PCX, TGA, TIF, PBM, PGM, XPM
  - GIF, PNG  # Support transparency
- Loading:

```
s = pygame.image.load("img.jpg")
s = pygame.image.load("sprites\\img.jpg")
s = pygame.image.load("..\\..\\media\\img.jpg")
s = pygame.image.load("c:\\game\\img.jpg")
s = pygame.image.load("/usr/bin/img.jpg")
import os
print(os.sep)   # '\\' on Windows, '/' on *nix
```

# Sprites / Images, cont.

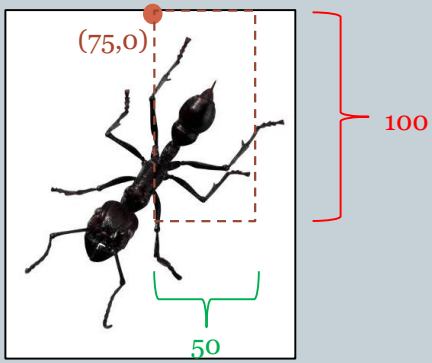
- Converting:
  - Color / bit encoding…
  - Why it's good to convert…
  - Command:

```
s = pygame.image.load("img.jpg")
s = s.convert()
t = pygame.image.load("img2.png").convert_alpha()
```

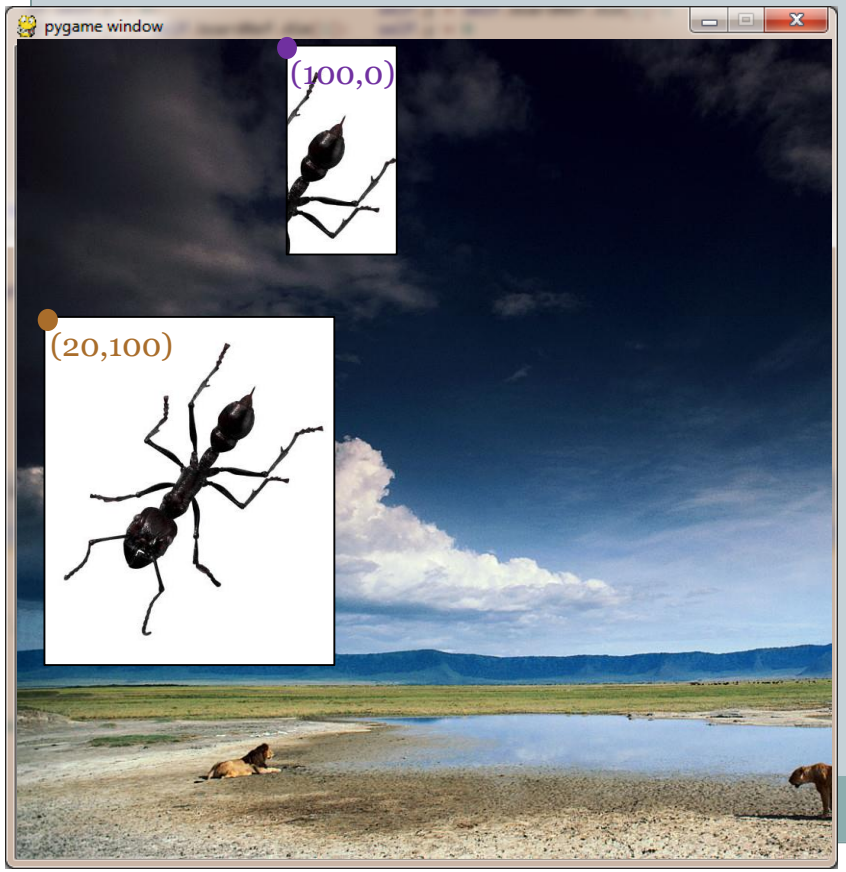
# Blitting

- Copies contents of one (part of) a surface to another.

ant_img (150x180)




screen (600x600)




```
dest_surf.blit(src_surf, pos)
dest_surf.blit(src_surf, pos,
      src_rect)
```

```
screen.blit(ant_img, (20,100))
screen.blit(ant_img, (100,0),
      (75,0,50,100))
```

# Transforming surfaces

```
pygame.transform.scale(orig_surf, (new_width, new_height))
pygame.transform.rotate(orig_surf, degrees)
```

- Returns a new surface
  - Must be blitted to screen to be seen
- Original surface is unchanged!
- [Example]
- Rotating "in-place"

# Fonts

- Rendering text in pygame
  - As opposed to the print statement
- Two steps:
  1. Create a font object (just done once)
     a. From a .ttf file
     b. From a built-in font
  2. Render text
     - Creates a new surface
     - …which must be blitted to the screen.
- [Example]