Python for videogames

Videogames Technology
Asignatura transversal

Departamento de Automática

Universidad
de Alcalá

Objectives

1. First contact with Arcade.
2. Understand modules and packages.
3. Basic package management.
4. Introduce virtual environments.

Bibliography

- The Python Tutorial. Chapter 6: Modules. (Link)
- Paul Vincent Craven. Easy 2D game creation with Arcade. (Link)
- Paul Vincent Craven. Learn to Program with Arcade. (Link)

Table of Contents

Universidad
de Alcalá

Introduction

Why modules?

- **Main function**: Organization.

- **Reuse**: To provide software solutions, that have been proven to work, to solve similar problems.

Modules

Creation

A module is just a Python script with `.py` extension

fibo.py

```python
def fib(n):
    """ Print a Fibonacci series up to n """
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

def fib2(n):
    """ Print a Fibonacci series up to n """
    result = [] # Declare a new list
    a, b = 0, 1
    while a < n:
        result.append(a) # Add to the list
        a, b = b, a+b
    return result
```

Modules

Using modules (I)

```
>>> import fibo
>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibo.fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo.__name__
'fibo'
>>> fib = fibo.fib
>>> fib(100)
1 1 2 3 5 13 21 34 55 89
```

Modules

Using modules (II)

A module can import other modules

- Name conflicts may arise: Each module has a symbol table
- It means you should invoke it as `modname.itemname`

It is possible to import items directly

- `from module import name1, name2`
- `from module import *`
- It uses the global symbol table (no need to use the modname)

```
>>> from fibo import fib, fib2
>>> fib(100)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Modules

Using modules: Example

Modules usually are used as libraries

List zip file contents (file.zip must exist. Open in read mode)

```python
import zipfile

file = zipfile.ZipFile("file.zip", "r")

# list filenames
for name in file.namelist():
    print(name)

# list file information
for info in file.infolist():
    print(info.filename, info.date_time, info.file_size)
```

Several examples here: http://pymotw.com/2/PyMOTW-1.132.pdf

Introduction
○
Modules
○○○○●○○
Packages
○○○○○○○○○
Virtual environments
○
Other cool code examples
○○○○○
Arcade
○○○○○○○○○○

Modules

Modules as scripts (I)

When a module is imported, its statements are executed

- It declares functions, classes, variables ...
- ... and also executes code
- It serves to initialize the module

Very useful to use modules as programs and libraries

- Good to reuse code

Modules

Modules as scripts (II)

fibo2.py

```python
def fib(n):
    """ Print a Fibonacci series up to n """
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

if __name__ == "__main__":
    fib(100)
```

__name__ is a special variable set to the module's name

Linux/PowerShell console

```
$ python3 fibo2.py 50
1 1 2 3 5 8 13 21 34
```

Python interpreter

```
>>> import fibo2
>>> fibo2.fib(100)
1 1 2 3 5 8 13 21 34
```

Introduction
○
Modules
○○○○○○○●
Packages
○○○○○○○○○
Virtual environments
○
Other cool code examples
○○○○○
Arcade
○○○○○○○○○

Modules
The dir() function

`dir()`: Built-in function that returns the names defined in a module

- Without arguments, it returns your names

```
>>> import fibo, sys
>>> dir(fibo)
['__name__', 'fib', 'fib2']
>>> dir()
['__builtins__', ... , '__spec__']
>>> variable = 'Hello'
>>> dir()
['__builtins__', ... , '__spec__', 'variable']
```

Packages
Package concept (I)

If a module gets too big, many problems arise

- Name collisions
- It is good to organize modules in a bigger structure: Packages

Packages can be seen as "dotted module names"

- It is just a module that contains more modules
- Make life easier in big proyects
- The name `A.B` designates a submodule `B` in a package named `A`

Must contain an `__init__.py` file in the root directory

- Executed when the package is imported for the first time

A package may be distributed with pip

- Arcade is an example of package

Packages
Package concept (II)

```
arcade/
├── __init__.py
├── __main__.py
├── sprites/
│   ├── __init__.py
│   ├── sprite.py
│   ├── animated.py
│   └── ...
├── types/
│   ├── __init__.py
│   ├── color.py
│   └── ...
├── color/
│   └── __init__.py
└── ...
```

(Arcade 3.3.3 source code)

Packages

Importing a package (I)

```
arcade/
├── __init__.py
├── __main__.py
├── sprites/
│   ├── __init__.py
│   ├── sprite.py
│   ├── animated.py
│   └── ...
├── types/
│   ├── __init__.py
│   ├── color.py
│   └── ...
├── color/
│   └── __init__.py
└── ...
```

Importing the whole package

```
import arcade
red = arcade.types.Color(255, 0, 0, 255)
```

Alternative way to import an individual module

```
from arcade.types import Color
red = Color(255, 0, 0, 255)
```

Give an alias to a namespace

```
import arcade.types as foo
red = foo.Color(255, 0, 0, 255)
```

Packages

Importing a package (II)

Imagine we run `from arcade.tilemap import *`

- In theory, it would import the whole package
- In practice, it would take too much time
- ... and mess the namespace

`__init__.py` may define a list named `__all__`

- It contains the names that will be exposed by that package
  - Functions, variables, modules and classes

```
arcade/tilemap/__init__.py

...
__all__ = ["TileMap", "load_tilemap"]
...
```

Introduction
○

Modules
○○○○○○○

Packages
○○○○○●○○○

Virtual environments
○

Other cool code examples
○○○○○

Arcade
○○○○○○○○○

Modules

The __main__.py file

The file `__main__.py` is an alternative entry point in a package

- Useful in command line mode and IDEs

> **Linux/PowerShell console**
>
> ```
> $ python3 -m mymodule
> ```

Try to run `python -m arcade`!

- (arcade/__main__.py source code)

Introduction
○

Modules
○○○○○○○

Packages
○○○○○○●○○

Virtual environments
○

Other cool code examples
○○○○○

Arcade
○○○○○○○○○○

Modules

Imports good practices

Imports style convention

1. Standard library
   Build-in Python modules
2. Third-party libraries
   Installed with pip
3. Local modules
   Modules you created

Good practices

- Avoid wildcards
  `from module import *`
- Alphabetic order within each subgroup

arcade/tilemap/__init__.py

```python
"""
Example module with well-
    organized imports
"""

# Standard library
import json
import os

# Third-party libraries
import arcade
import numpy as np

# Local modules
from config import SETTINGS
from utils.helpers import
    process_data

# Rest of the code
def main():
    pass
```

Packages

Installing packages

Command-line automatic tool: `pip` (sometimes `pip3`)

- Very similar to `apt-get` in Linux
- Automatic dependencies management

pip usage (from OS terminal)

```
$ pip install SomePackage
```

```
$ pip install Pillow
```

In VS Code, you should use pip in the terminal

Introduction
○

Modules
○○○○○○○

Packages
○○○○○○○●

Virtual environments
○○○○○
○

Other cool code examples
○○○○○

Arcade
○○○○○○○○○○

Packages

requirements.txt

A project may contain a `requirements.txt` file

- It recreates an environment
- Plain text file with dependences
- Can define fixed or minimum version

Supported by different tools

- ... including VS Code

---

requirements.txt

```
arcade==2.6.17
seaborn>=0.12
numpy
pandas
```

---

pip usage (from OS terminal)

```
$ pip install -r requirements.txt
```
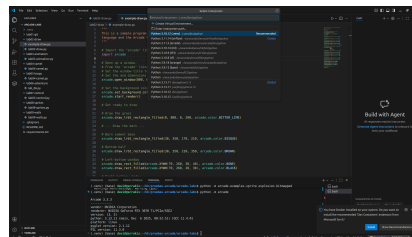
Universidad
de Alcalá

Virtual environments

Versioning is problematic

- Python version (2.x, 3.x)
- Packages version

Solution: virtual environment

- Self-contained directory with a Python installation
- Particular version of Python and packages

Different solutions: venv and conda

Great with `requirements.txt`!!!

Cool code examples

Example 1: Open a web browser

browser.py

```python
import webbrowser

url = input('Give me an URL: ')

webbrowser.open(url)
```

Cool code examples

Example 2: Create a thumbnail

thumbnail.py

```
1  from PIL import Image
2
3  size = (128, 128)
4  saved = "africa.jpg"
5
6  im = Image.open("africa.tif")
7  im.thumbnail(size)
8  im.save(saved)
9  im.show()
```

(Source)

africa.jpg

Introduction
○

Modules
○○○○○○○

Packages
○○○○○○○○○

Virtual environments
○

Other cool code examples
○○●○○

Arcade
○○○○○○○○○

Cool code examples

Example 3: Send an email with Gmail

gmail.py

```python
"""The first step is to create an SMTP object,
each object is used for connection
with one server."""

import smtplib
server = smtplib.SMTP('smtp.gmail.com', 587)

# Next, log in to the server
server.login("youremailusername", "password")

# Send the mail
msg = "\nHello!" # /n separates the message from the headers
server.sendmail("you@gmail.com", "target@example.com", msg)
```
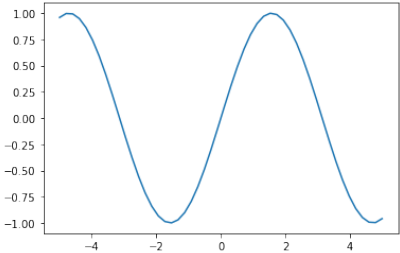
(Source)

Modules

Example 4: Plot

plot.py

```python
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5, 5)
plt.plot(x, np.sin(x))
```
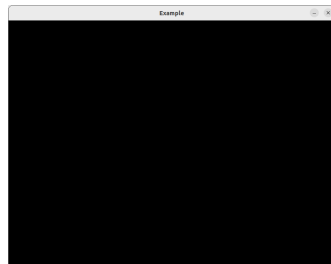
Modules

Example 5: Arcade

arcade.py

```python
import arcade

WIDTH = 600
HEIGHT = 800

arcade.open_window(WIDTH, HEIGHT, "
    Example")

arcade.run()
```

- (API documentation)
- (Arcade source code)

Arcade

Introduction

Arcade is an easy-to-use 2D motor engine (i.e. a Python package)

- Created by Paul Vincent Craven
- Based on Python
- More or less painless game development
- Didactic
- Free software

Requires

- Python 3.9+
- OpenGL capable hardware

Dependences

- Pyglet - Mutimedia library for Python

(Arcade web site)

Introduction
○

Modules
○○○○○○○

Packages
○○○○○○○○○

Virtual environments
○

Other cool code examples
○○○○○

Arcade
○●○○○○○○○○

Arcade

Open a Window (I)

arcade.py

```
1  import arcade
2
3  WIDTH = 600
4  HEIGHT = 800
5
6  arcade.open_window(WIDTH, HEIGHT, "Example")
7
8  arcade.run()
```
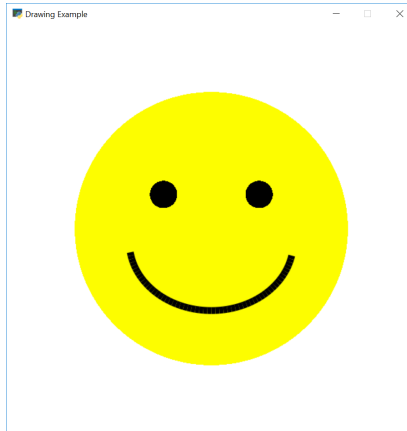
Arcade

Drawing setup

drawing.py

```python
import arcade

WIDTH = 600
HEIGHT = 800

arcade.open_window(WIDTH, HEIGHT, "Example")

arcade.set_background_color(arcade.color.WHITE)

arcade.start_render()

# Drawing here

arcade.finish_render()

arcade.run()
```

Arcade

Drawing (I)

(Source code)

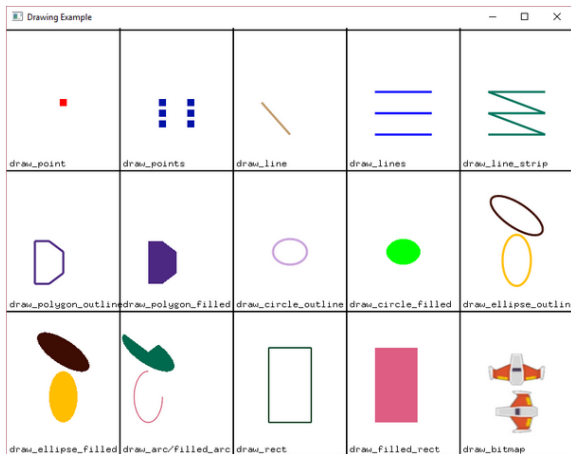Universidad
de Alcalá

Arcade

Drawing (II)

smile.py

```python
# Draw the face
x = 300; y = 300; radius = 200
arcade.draw_circle_filled(x, y, radius, arcade.color.YELLOW)

# Draw the right eye
x = 370; y = 350; radius = 20
arcade.draw_circle_filled(x, y, radius, arcade.color.BLACK)

# Draw the left eye
x = 230; y = 350; radius = 20
arcade.draw_circle_filled(x, y, radius, arcade.color.BLACK)

# Draw the smile
x = 300; y = 280; width = 120; height = 100
start_angle = 190; end_angle = 350; line_width = 10
arcade.draw_arc_outline(x, y, width, height, arcade.color.BLACK,
                        start_angle, end_angle, line_width)
```

Introduction
O

Modules
OOOOOOO

Packages
OOOOOOOOO

Virtual environments
O

Other cool code examples
OOOOO

Arcade
OOOOOO●OOOO

Arcade

Drawing primitives (I)

(Source code)

Arcade

Drawing primitives (II)

```
draw_text()

draw_rect_filled()
draw_rect_outline()
draw_lrbt_rectangle_filled()
draw_lrbt_rectangle_outline()

draw_polygon_filled()
draw_polygon_outline()

load_texture()
draw_texture_rect()

draw_triangle_filled()
draw_triangle_outline()
```

```
draw_arc_filled()
draw_arc_outline()

draw_circle_filled()
draw_circle_outline()

draw_ellipse_filled()
draw_ellipse_outline()

draw_line()
draw_line_strip()
draw_lines()

draw_point()
draw_points()
```
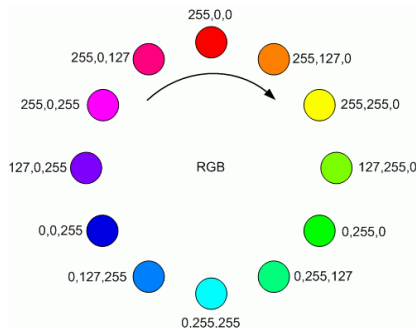
(Arcade API)

Arcade
Colors

How can we know which colors has Arcade available?

- The reference API is your friend!
- (`arcade.color` reference documentation)

Arcade

Colors: RGB

(Source code)

Universidad
de Alcalá