

ACKNOWLEDGEMENT

Flappy Bird is a popular mobile game that has been adapted to various platforms, including desktop and web browsers. In this project, we developed a Flappy Bird game using Python programming language, which is an interpreted, high-level, general-purpose programming language. The game was developed using the Pygame library, which is a cross-platform set of Python modules designed for video game development.

The goal of the game is to control a bird and navigate it through a series of obstacles by flapping its wings. The player must time the bird's flapping to avoid obstacles and progress as far as possible. The game is over if the bird collides with an obstacle or falls to the ground.

One of the key abstractions used in this project is object-oriented programming (OOP). OOP is a programming paradigm that models real-world

objects as software objects. In our Flappy Bird game, we used OOP to model the game's elements, including the bird, obstacles, and game screen. This approach allowed us to encapsulate the game's logic and data, making it more modular and easier to maintain.

Another abstraction used in this project is event-driven programming. Event-driven programming is a programming paradigm where the flow of the program is determined by user actions or external events. In our Flappy Bird game, we used event-driven programming to handle user input, such as keyboard or mouse clicks, and update the game's state accordingly.

We also used abstraction to separate the game's logic from its presentation. The Pygame library provides a set of functions for rendering graphics and handling user input. By separating the game's logic from its presentation, we could update the game's state without affecting the user interface.

To implement the Flappy Bird game in Python, we started by creating a Game class that initializes the game's elements, including the bird, obstacles, and game screen. We then created methods to handle user input, update the game's state, and render the game's graphics. These methods are called in response to user input or external events, allowing the game to progress and respond to player actions.

In conclusion, the Flappy Bird game developed in Python using Pygame library is a great example of using abstractions to simplify a complex system. By using OOP, event-driven programming, and separating the game's logic from its presentation, we were able to create a modular, maintainable game that provides a fun and engaging user experience. With the growing demand for Python programming language, this project can be a great starting point for game development beginners to learn Python and game development.

ABSTRACT

Like music and movies, video games are rapidly becoming integral to our lives. Over the years, you've yearned for every new gaming console, mastered each blockbuster within weeks after its release, and even won a local gaming competition or two. But lately, you've been spending a lot of time thinking about a game idea of your own, or are exploring the possibility of making a career in this vibrant and growing industry. But where should you begin? *Beginning Game Development with Python and Pygame* is written with the budding game developer in mind, introducing games development through the Python programming language and the popular Pygame games development library. Authored by industry veteran and Python expert Will McGugan, who worked on the *MotorStorm* game for PlayStation 3, you'll be privy to insights that will help you exploit Pygame to its maximum potential, but also make you a more creative and knowledgeable games developer all round. • Learn how to create advanced games by taking advantage of the popular open-source Python programming

language and Pygame games development library •
Learn about coding gaming preferences, sound,
visual effects, and joystick/keyboard interaction •
Discover the concepts that are crucial to success in
today's gaming industry, such as support for
multiple platforms, and granting users the ability
to extend and customize your games.



TABLE OF CONTENTS

1.0-INTRODUCTION.....	06
2.0-PREREQUISITES.....	
2.1-PYTHON.....	
2.1.1-INSTALLATION.....	
2.1.2-MODULES.....	
2.1.2.1-PYGAME.....	
2.1.2.2-SYS.....	
2.1.2.3-RANDOM.....	
2.1.3-PACKAGES.....	
2.1.3.1-IMPORTANT PACKAGES.....	
2.2-IDE (Editor).....	
3.0-PROBLEM IDENTIFICATION AND OBJECTIVE.....	
4.0-METHODOLOGY.....	
5.0-CONCLUSION.....	
6.0-EXPERIMENTAL AND RESULT.....	
7.0-SCOPE OF FUTURE WORK.....	

INTRODUCTION

Python is a powerful and popular programming language that is widely used for developing games. Python has a simple syntax and an easy-to-learn structure, which makes it ideal for developing games even for beginners. In this mini project, we will develop a simple game using Python. The game we will develop is called "Snake Game". It is a classic game that was first introduced in the 1970s and has been popular ever since. The objective of the game is to control a snake and eat as many food items as possible without crashing into the walls or the snake's own body. The game gets progressively harder as the snake gets longer and faster. To develop this game, we will use the Pygame library, which is a set of Python modules designed for game development. Pygame provides tools and functions for handling graphics, sound, and user input, which are essential for game development. The Snake Game will have a graphical user

interface that displays the game board and the snake. The game will start with a single block snake on the board and randomly placed food items. The player will control the snake using the arrow keys to move in four directions: up, down, left, and right. The snake will grow in length as it eats the food items, and the player will score points for each food item eaten. The game will end when the snake hits the walls or its own body. In addition to the game logic, we will also implement other features such as a scoreboard to keep track of the player's score and a game-over screen that displays the player's final score. Overall, the mini project of developing a Snake Game using Python is an excellent way to learn game development using a popular and easy-to-learn programming language. By the end of the project, the developer will have gained hands-on experience in working with Pygame and developing a complete game from scratch. The Snake Game can also be expanded upon by adding new features and functionality, making it a great starting point for further game development projects.

PREREQUISITES

To make a game using Python, the following prerequisites are recommended:

1. Basic programming knowledge: Familiarity with programming concepts such as variables, loops, and functions is essential for understanding how to write a game in Python.
2. Familiarity with the Python programming language: A good understanding of Python's syntax and basic constructs and knowledge of the standard library and popular third-party libraries is necessary for creating a game.
3. Understanding of game development concepts: Knowledge of game development concepts such as game mechanics, physics, and collision detection is crucial for designing and implementing a game.
4. Familiarity with a game engine or library: Game engines like Pygame, Pyglet, and PyOpenGL, or libraries like PygameZero,

Arcade, and PyEngine, can be used to simplify the process of creating a game, and knowledge of their APIs is necessary.

5. Understanding of 2D and 3D graphics: Knowledge of creating 2D and 3D graphics, such as sprites, animations, and 3D models, is needed to create the game's visual elements.
6. Experience with version control systems (VCS): Using VCS like git can help maintain and track changes to the game's codebase, as well as for collaborating with others.
7. Optional: Understanding testing and debugging tools and experience with them is a good thing to help you identify and fix bugs in your code.

It's worth noting that Making a game could be a big and complex process, and there are a lot of different skills and areas of knowledge to learn and master, but by starting with these prerequisites, you will be well on your way to creating your games using Python.

Some of the important prerequisites for making this FLAPPYMAN game in python

1. PYTHON
2. ANY IDE (VS CODE RECOMMENDED)
3. PYGAME MODULE
4. SYS MODULE
5. RANDOM MODULE



Figure 2

Figure 1



Figure 3

PYTHON

Python is a high-level, interpreted programming language created in 1989 by Guido van Rossum. It is a versatile language that can be used for a wide range of tasks, such as web development, data analysis, machine learning, and scientific computing. Some of its key features include:



Figure 4

Easy to read and write: Python code is often described as "readable" or "elegant," making it a popular choice for beginners and experienced programmers alike.

Large standard library: Python comes with a large number of modules and libraries, including ones

for common tasks like connecting to web servers, reading and writing files, and working with data.

Dynamically-typed: Python is a dynamically-typed language, which means that variables do not need to be declared with a specific type, and the type of a variable can change at runtime.

Interpreted: Python code is interpreted, not compiled, which can make it slower than compiled languages like C or Java. However, this also means running and debugging Python code is easier.

Multi-purpose: Python is used for a wide range of tasks, such as web development, data analysis, artificial intelligence, and more. This makes it a versatile choice for many types of projects.

Python has a large user community and is supported by many libraries and frameworks, such as NumPy, Pandas, Matplotlib, TensorFlow, and more. It is widely used in industry and

academia and has a reputation for being a beginner-friendly programming language

INSTALLATION

Installing and using Python on Windows 10 is very simple. The installation procedure involves just three steps:

1. Download the binaries
2. Run the Executable installer
3. Add Python to PATH environmental variables

To install Python, you need to download the official Python executable installer. Next, you need to run this installer and complete the installation steps. Finally, you can configure the PATH variable to use python from the command line. You can choose the version of Python you wish to install. It is recommended to install the latest version of Python, which is 3.7.3 at the time of writing this article.

Step 1: Download the Python Installer binaries

Open the official Python website in your web browser. Navigate to the Downloads tab for Windows.

Choose the latest Python 3 release. In our example, we choose the latest Python 3.7.3 version.

Click on the link to download the Windows x86 executable installer if you are using a 32-bit installer. In case your Windows installation is a 64-bit system, then download Windows x86-64 executable installer.

Python Releases for Windows

- [Latest Python 3 Release - Python 3.7.3](#)
- [Latest Python 2 Release - Python 2.7.16](#)

Stable Releases

- [Python 3.7.3 - March 25, 2019](#)

Note that Python 3.7.3 cannot be used on Windows XP or earlier.

- Download [Windows help file](#)
- Download [Windows x86-64 embeddable zip file](#)
- Download [Windows x86-64 executable installer](#)
- Download [Windows x86-64 web-based installer](#)
- Download [Windows x86 embeddable zip file](#)
- Download [Windows x86 executable installer](#)
- Download [Windows x86 web-based installer](#)

Figure 5

Step 2: Run the Executable Installer

1. Once the installer is downloaded, run the Python installer.
2. Check the Install launcher for all user's checkboxes. Further, you may check the Add Python 3.7 to path check box to include the interpreter in the execution path.
- 3.



Figure 6

Select Customize installation. Choose the optional features by checking the following checkboxes:

4.Documentation

5.pip

6.tcl/tk and IDLE (to install tkinter and IDLE)

7.Python test suite (to install the standard library test suite of Python)

8.Install the global launcher for `.py` files. This makes it easier to start Python

9.Install for all users.

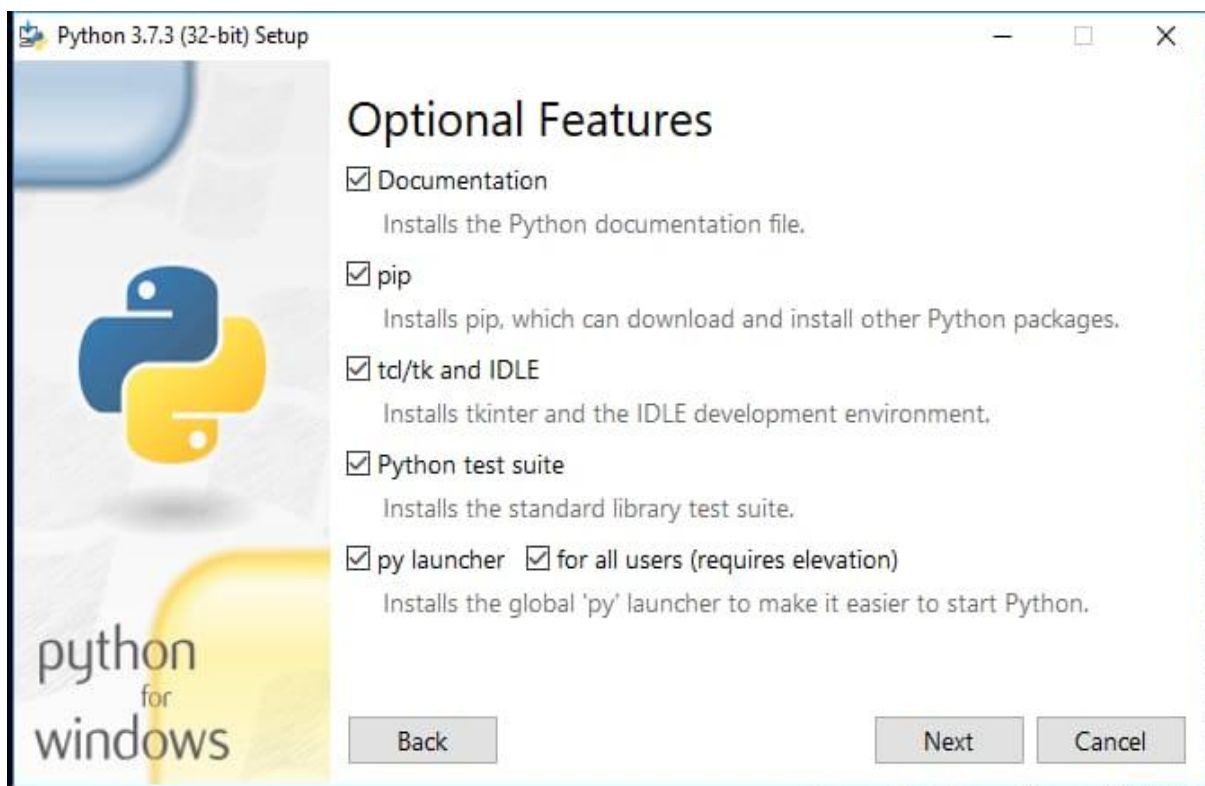


Figure 7

Click Next.9. This takes you to Advanced Options available while installing Python. Here, select the Install for all users and Add Python to environment variables check boxes. Optionally, you can select the Associate files with Python, Create shortcuts for installed applications, and other advanced options. Make note of the python installation directory displayed in this step. You would need it for the next step. After selecting the Advanced options, click Install to start the

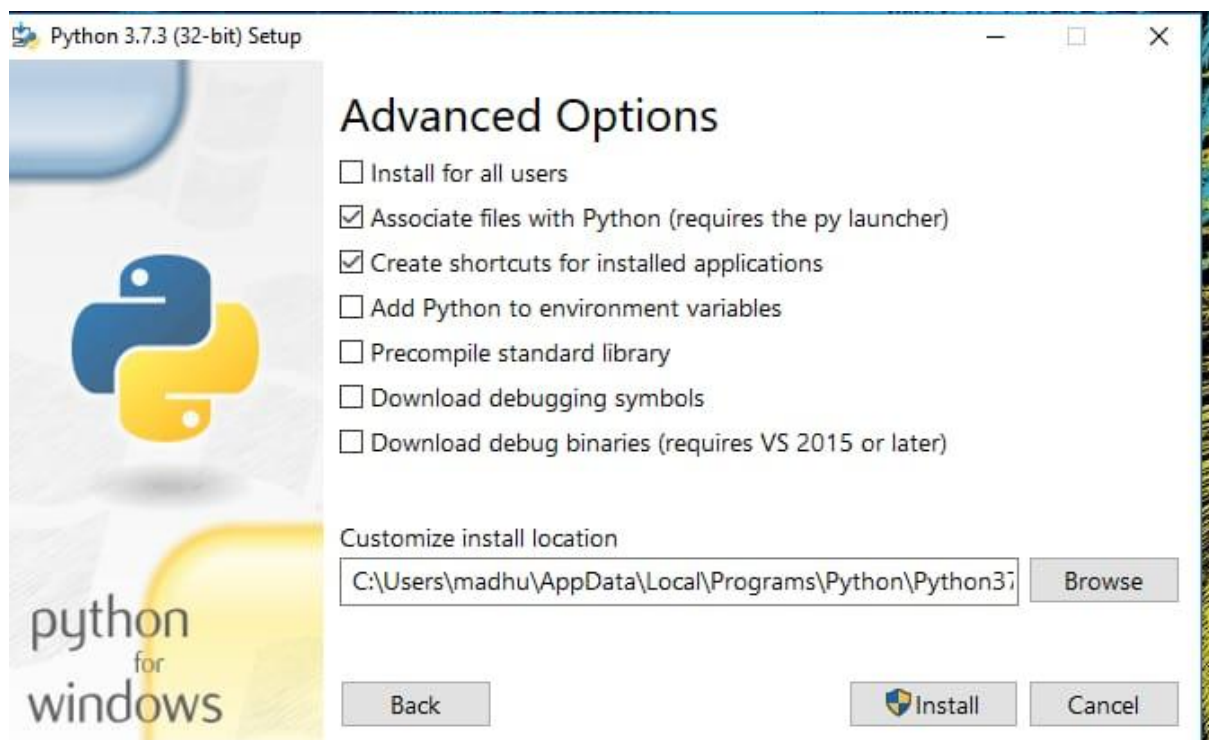


Figure 8

installation.

10. Once the installation is over, you will see a Python Setup Successful window.



Figure 9

Step 3: Add Python to environmental variables

The last (optional) step in the installation process is to add Python Path to the System Environment variables. This step is done to access Python through the command line. In case you have added Python to environment variables while setting the Advanced options during the installation procedure, you can avoid this step.

Else, this step is done manually as follows. In the Start menu, search for “advanced system settings”. Select “View advanced system settings”. In the “System Properties” window, click on the “Advanced” tab and then click on the “Environment Variables” button. Locate the Python installation directory on your system. If you followed the steps exactly as above, python will be installed in the below locations:

- C:\Program Files (x86)\Python37-32: for 32-bit installation
- C:\Program Files\Python37-32: for 64-bit installation

The folder name may be different from “Python 37-32” if you installed a different version. Look for a folder whose name starts with Python. Append the following entries to the PATH variable as shown below:

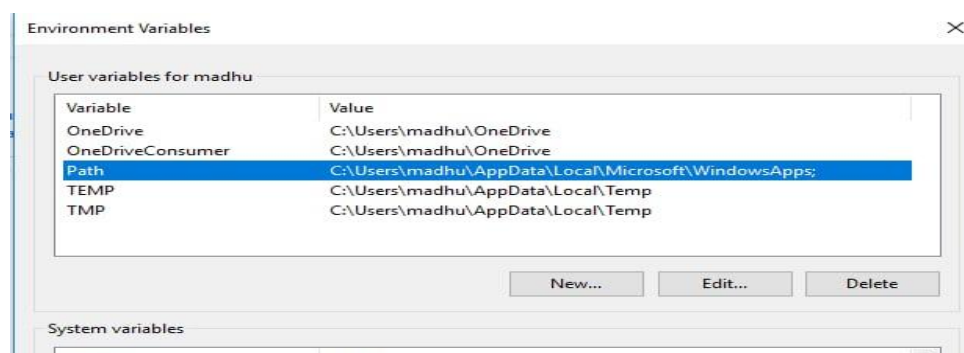


Figure 10

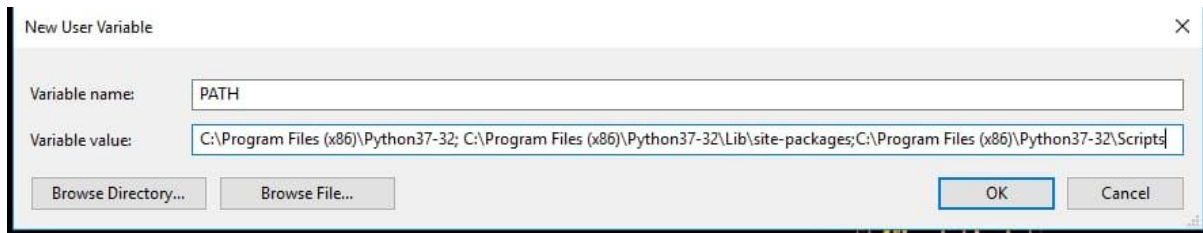


Figure 11

Step 4: Verify the Python Installation

You have now successfully installed Python 3.7.3 on Windows 10. You can verify if the Python installation is successful either through the command line or through the IDLE app that gets installed along with the installation. Search for the command prompt and type “python”. You can see that Python 3.7.3 is successfully installed.

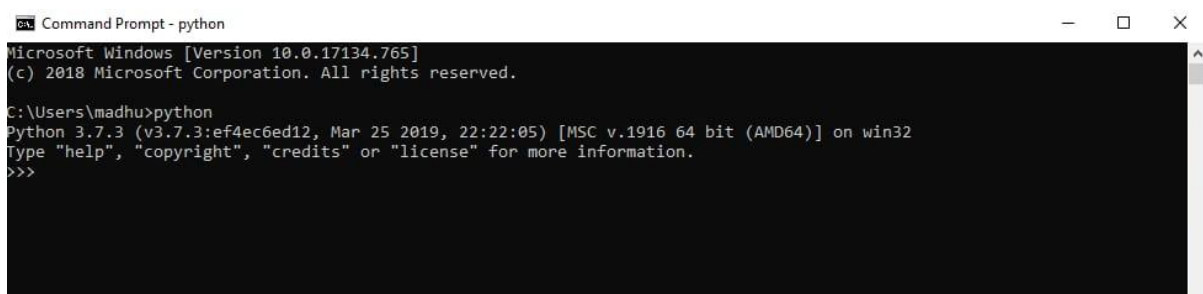


Figure12

An alternate way to reach python is to search for “Python” in the start menu and click on IDLE (Python 3.7 64-bit). You can start coding in Python

using the Integrated Development Environment(IDLE).

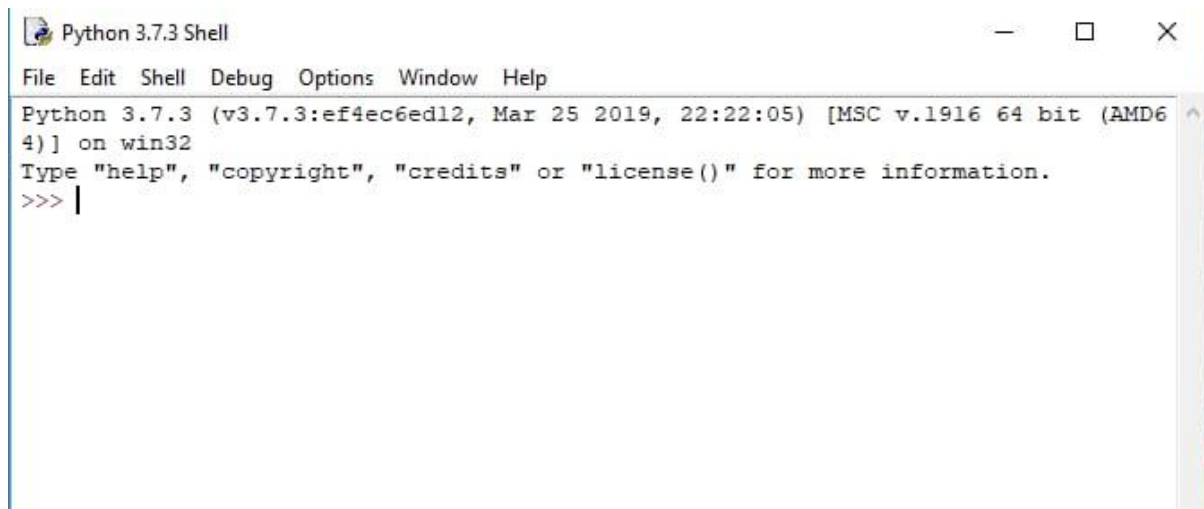


Figure 13

MODULES

Python Modules

In this article, we will cover all about Python modules, such as How to create our simple module, Import Python modules, From statements in Python, how we can use the alias to rename the module, etc.

What is Python Module?

A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized.

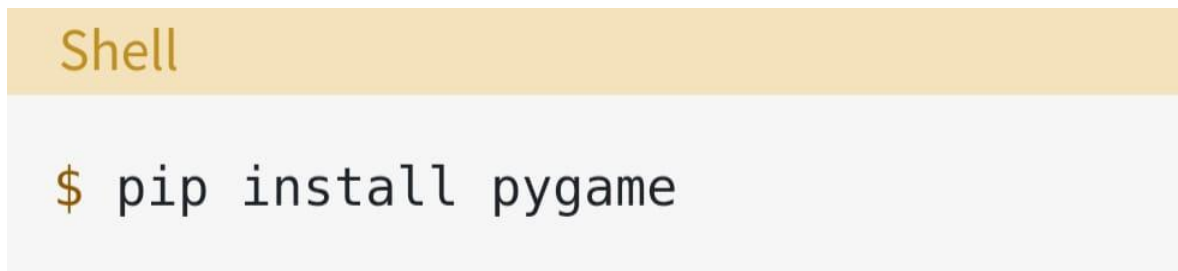
Some of the important modules are listed below

1. Pygame
2. Sys
3. Random

1. Pygame

pygame is a Python wrapper for the SDL library, which stands for Simple DirectMedia Layer. SDL provides cross-platform access to your system's underlying multimedia hardware components, such as sound, video, mouse, keyboard, and joystick. pygame started life as a replacement for the stalled PySDL project. The cross-platform nature of both SDL and pygame means you can write games and rich multimedia Python programs for every platform that supports them!

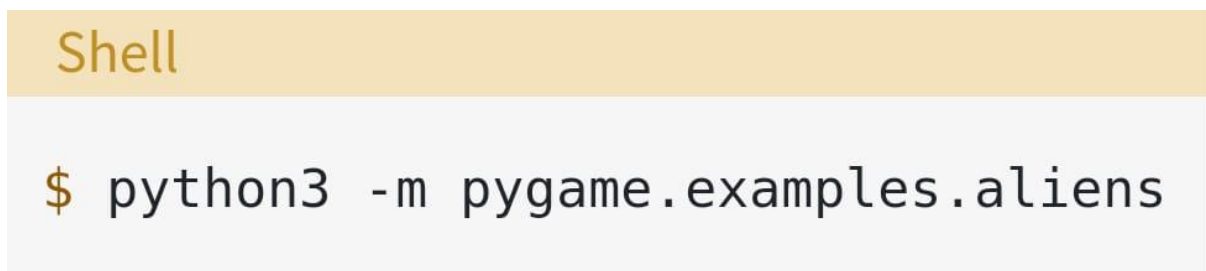
To install pygame on your platform, use the appropriate pip command:

A terminal window with a yellow header bar labeled "Shell". Below the header, the command `$ pip install pygame` is entered in a monospaced font.

```
Shell
$ pip install pygame
```

Figure 14

You can verify the installation by loading one of the examples that come with the library:

A terminal window with a yellow header bar labeled "Shell". Below the header, the command `$ python3 -m pygame.examples.aliens` is entered in a monospaced font.

```
Shell
$ python3 -m pygame.examples.aliens
```

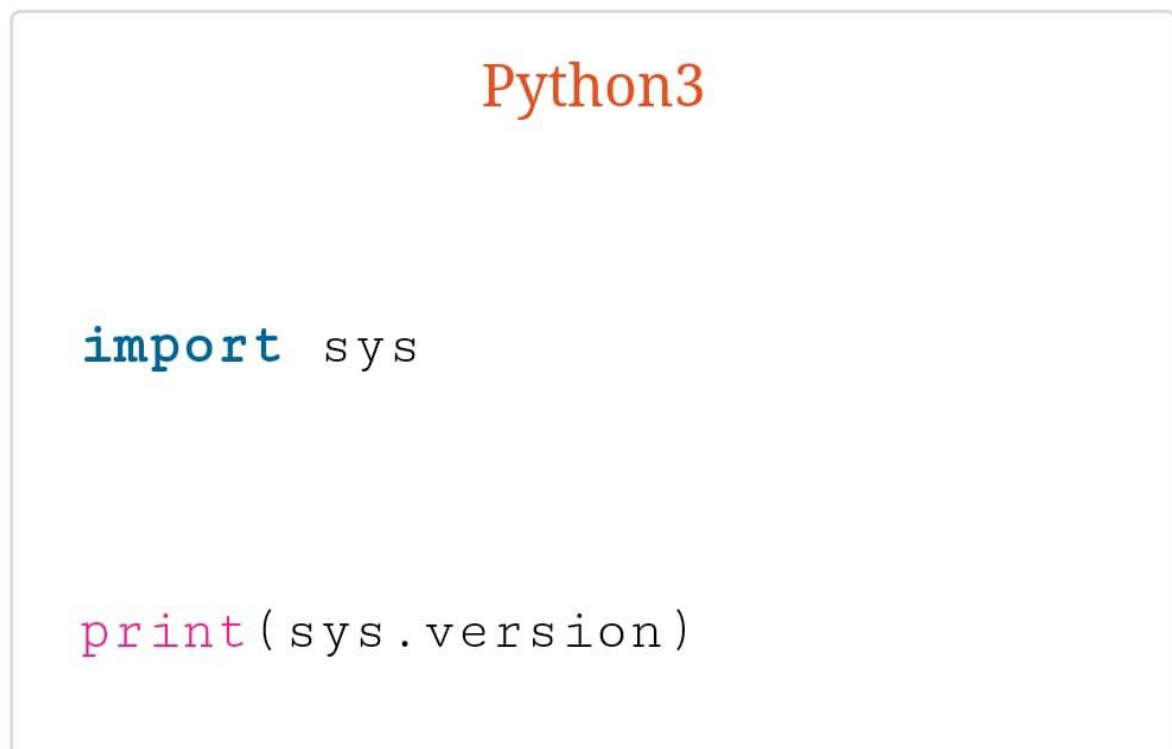
Figure 15

If a game window appears, then the pygame is installed properly! If you run into problems, then the Getting Started guide outlines some known issues and caveats for all platforms.

Python sys Module

The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter. Let's consider the below example.

Example:-

A screenshot of a Python3 terminal window. The title bar at the top reads "Python3" in orange text. The terminal has a light gray background. The first line of code is "import sys" in blue text. The second line is "print(sys.version)" in pink text. The output of the command is "3.8.5" in blue text, appearing on the line following the print statement.

```
Python3

import sys

print(sys.version)

3.8.5
```

Figure 16

Output:-

```
3.6.9 (default, Oct  8 2020, 12:12:24)  
[GCC 8.4.0]
```

Figure 17

In the above example, the **sys.version** is used which returns a string containing the version of Python Interpreter with some additional information. This shows how the sys module interacts with the interpreter. Let us dive into the article to get more information about the sys module.

3. Python Random Module

Python Random module is an in-built module of Python that is used to generate random numbers. These are pseudo-random numbers means they are not truly random. This module can perform random actions such as generating random numbers, printing a value for a list or string, etc.

Example: Printing a random value from a list

```
# import random
import random

# prints a random value from the list
list1 = [1, 2, 3, 4, 5, 6]
print(random.choice(list1))
```

Figure 18

Output:-

2

As stated above random module creates pseudo-random numbers. Random numbers depend on the seeding value. For example, if the seeding

value is 5, the output of the below program will always be the same.

Example: Creating random numbers with seeding value

```
import random

random.seed(5)

print(random.random())
print(random.random())
```

Figure 19

Output:-

0.6254875485256985

0.7254856852953568

PACKAGES

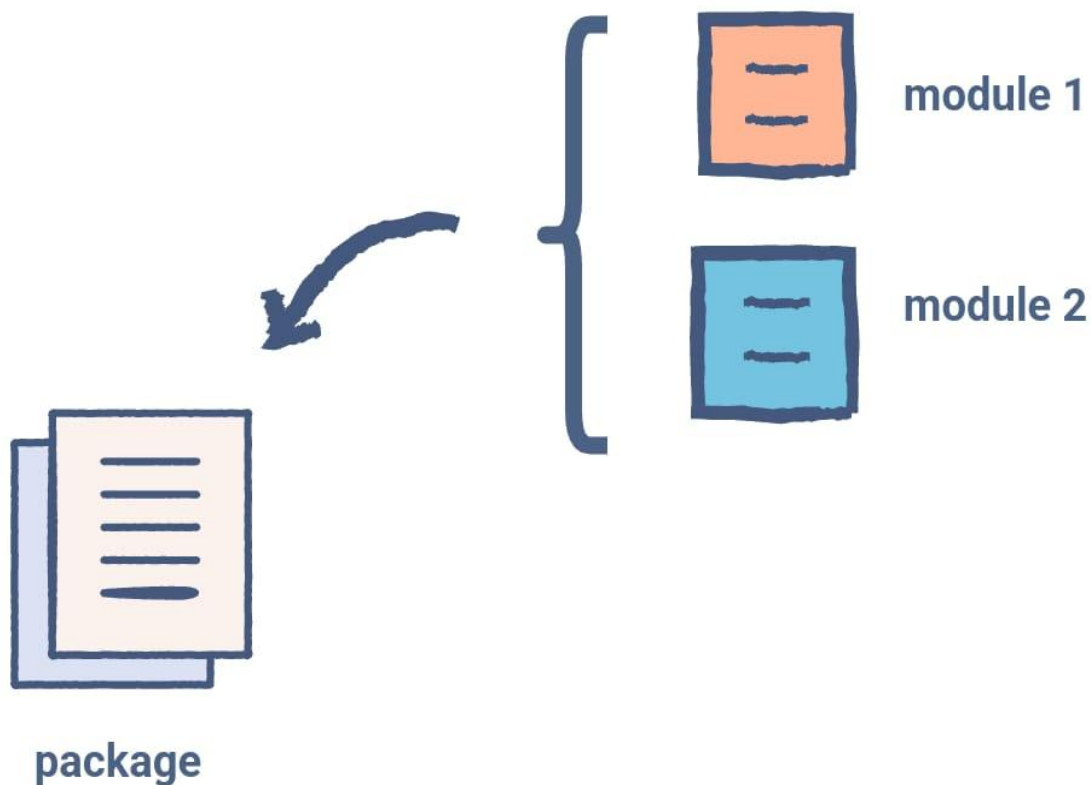
A python package is a collection of modules. Modules that are related to each other are mainly put in the same package. When a module from an external package is required in a program, that package can be imported and its modules can be

Any Python file, whose name is the module's name property without the **.py** extension, is a **module**.

put to use.

A package is a directory of Python modules that contains an additional **`_init.py`** file, which distinguishes a package from a directory that is supposed to contain multiple Python scripts. Packages can be nested to multiple depths if each

corresponding directory contains its `__init__.py` file.



When you import a module or a package, the object created by Python is always a typical

When you import a package, only the methods and the classes in the `__init__.py` file of that package are directly visible.

module.

SOME OF THE THE MOST IMPORTANT PYTHON PACKAGES

1. **NumPy** - NumPy is the primary tool for scientific computing in Python. It combines the flexibility and simplicity of Python with the speed of languages like C and Fortran
2. **Pandas** - pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.
3. **Matplotlib** - Matplotlib is the most common data exploration and visualization library. You can use it to create basic graphs like line plots, histograms, scatter plots, bar charts, and pie charts. You can also create animated and interactive visualizations with

this library. Matplotlib is the foundation of every other visualization library.