****

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**

**KHOA ĐIỆN TỬ II**

**\_\_\_\_\_\_\_\_\_\_\_\_\_**

**BÁO CÁO   
ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH: CÔNG NGHỆ KỸ THUẬT ĐIỆN- ĐIỆN TỬ  
HỆ: CHÍNH QUY  
NIÊN KHÓA: 2014-2019**

**Đề tài: Thiết kế game 2D chạy trên Android/IOS**

**Sinh viên thực hiện: Võ An Khang  
 MSSV: N14DCDT138  
 Lớp: D14CQKD01-N  
 Giáo viên hướng dẫn: Ths. Phạm Thế Duy**

Thành phố Hồ Chí Minh – 12/2018

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**

**KHOA ĐIỆN TỬ II**

**\_\_\_\_\_\_\_\_\_\_\_\_\_**

**BÁO CÁO   
ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH: CÔNG NGHỆ KỸ THUẬT ĐIỆN- ĐIỆN TỬ  
HỆ: CHÍNH QUY  
NIÊN KHÓA: 2014-2019**

**Đề tài: Thiết kế game 2D chạy trên Android/IOS**

**Sinh viên thực hiện: Võ An Khang  
 MSSV: N14DCDT138  
 Lớp: D14CQKD01-N  
 Giáo viên hướng dẫn: Ths. Phạm Thế Duy**

Thành phố Hồ Chí Minh – 12/2018

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

**MỤC LỤC**

**DANH MỤC HÌNH VẼ VÀ ĐỒ THỊ**

**GIỚI THIỆU CHUNG**

# **CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI.**

Trong thời đại điện tử và công nghệ thông tin bùng nổ như hiện nay, sản phẩm công nghệ ngày càng thu hút và nhận được sự quan tâm rất lớn từ phía người dùng, đặc biệt là về sản phẩm Game được rất được ưa chuộng không chỉ các thanh thiếu niên mà cả những người trung niên, lớn tuổi vì nhu cầu giải trí cao. Ngành công nghiệp Game hiện nay có thể nói là bùng nổ, với tốc độ phát triển chóng mặt, rất nhiều Game hay và hấp dẫn đã ra đời trong thời gian qua. Phía sau những Game phát triển, nổi tiếng như vậy đều có một ngôn ngữ lập trình và công cụ hỗ trợ giúp cho nhà phát triển Game một cách nhanh chóng, đơn giản hơn. Thế nhưng mỗi hệ điều hành khác nhau có những công cụ phát triển Game khác nhau. Từ những xu hướng và bất cập trên, đồ án này sẽ tập trung nghiên cứu tìm hiểu về Kivy, một thư viện Python mã nguồn mở được tạo ra để phát triển ứng dụng chạy trên đa nền tảng phổ biến hiện nay như Windows, Linux, OS X, Android, iOS. Chuẩn bị kiến thức và kỹ năng cho định hướng nghề nghiệp (phát triển game/ứng dụng) sau này của chúng em, góp phần vào sự phát triển của ngành công nghiệp Game/Ứng dụng cho nước nhà.

## **Mục tiêu đề tài.**

* Áp dụng các kiến thức đã học vào việc nghiên cứu và phát triển game đơn giản trên thiết bị di động.
* Tạo tiền đề để phát triển và ứng dụng vào thực tế.

## **Nhiệm vụ của đề tài.**

* Tìm hiểu và nghiên cứu:

⬩ Ngôn ngữ lập trình Python.

⬩ Thư viện lập trình Kivy.

* Tạo ra một mô hình Game đơn giản.
* Xây dựng một Game đơn giản chạy trên Raspberry, Windows, Linux và Android/iOS phục vụ cho việc giải trí.

## **Đối tượng nghiên cứu của đề tài.**

* Ngôn ngữ Python.
* Thư viện lập trình: Kivy.

## **Giới hạn của đề tài.**

Trong đề tài này sẽ tạo ra một Game đơn giản chạy trên Windows, Linux, Raspbian, Android/iOS.

* Không đồ họa 3D.
* Không hình ảnh trang trí, thiết kế bắt mắt.
* Không âm thanh đính kèm với Game.

## **Ý nghĩa đề tài.**

* Nghiên cứu hướng đi mới cho ngành công nghiệp Game/Ứng dụng trên các thiết bị di động.
* Tạo cơ sở định hướng nghề nghiệp phát triển game/ứng dụng mang tính giải trí trong tương lai, góp phần phát triển ngành công nghiệp game và ứng dụng cho nước nhà.

## **Bố cục đề tài.**

Nội dung đề tài được trình bày với bố cục như sau:

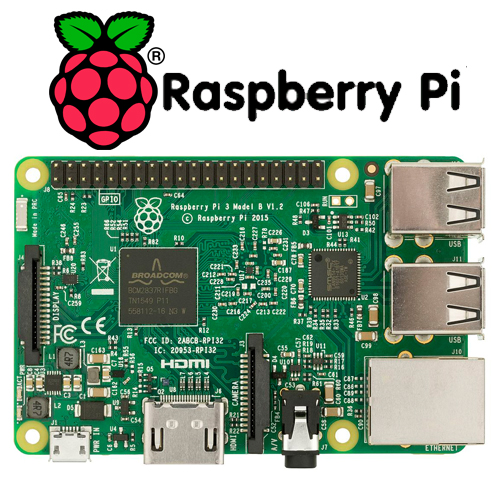
* Chương 1: Giới thiệu tổng quan về đề tài.
* Chương 2: Các cơ sở lý thuyết liên quan.
* Chương 3: Xây dựng Game 2D chạy trên thiết bị di động.
* Chương 4: Tổng kết, đánh giá và hướng phát triển đề tài.

# **CHƯƠNG 2. CƠ SỞ LÝ THUYẾT**

## **Tổng quan về máy tính nhúng Raspberry pi 3 và hệ điều hành Raspbian.**

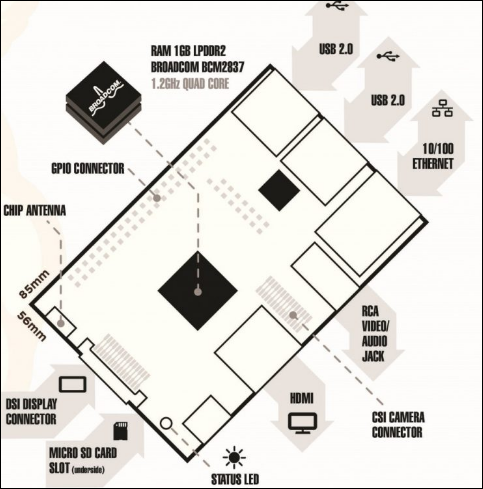
### **Giới thiệu về Raspberry pi 3.**

Raspberry Pi là từ để chỉ các máy tính chỉ có một board mạch (hay còn gọi là máy tính nhúng) kích thước chỉ bẳng một thẻ tín dụng, được phát triển tại Anh bởi Raspberry Pi Foundation với mục đích ban đầu là thúc đẩy việc giảng dạy về khoa học máy tính cơ bản trong các trường học và các nước đang phát triển.



Hình 2.1.1: Bo mạch raspberry pi 3

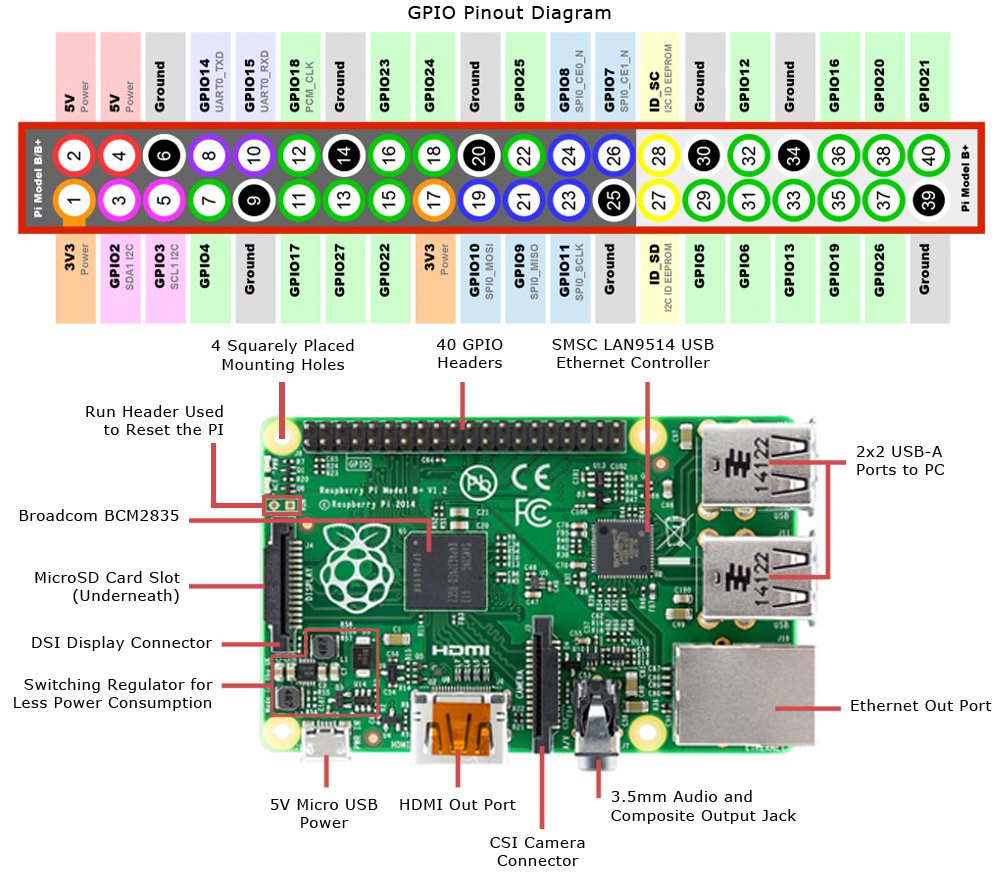
### **Phần cứng Raspberry pi 3.**



Hình 2.1.2: Sơ đồ cấu tạo Raspberry Pi

Raspberry Pi có hai phiên bản, **Model A** có giá 25$ và **Model B** có giá 35$ . Model B như hình trên thông dụng hơn cả. Model B bao gồm những phần cứng và những cổng giao diện:

* Bộ nhớ 1 GB RAM
* 1 cổng HDMI cho đầu ra âm thanh / video số
* 1 cổng video RCA cho đầu ra video Analog
* Jack Headphone Stereo 3.5mm cho đầu ra âm thanh Analog
* 04 cổng USB
* 01 đầu đọc thẻ nhớ SD để tải hệ điều hành
* 01 cổng Ethernet LAN.
* 01 giao diện GPIO (General Purpose Input/Output)
* Giao tiếp camera qua CSI
* Hỗ trợ hiển thị DSI
* Vi xử lý hình ảnh VideoCore IV 3D

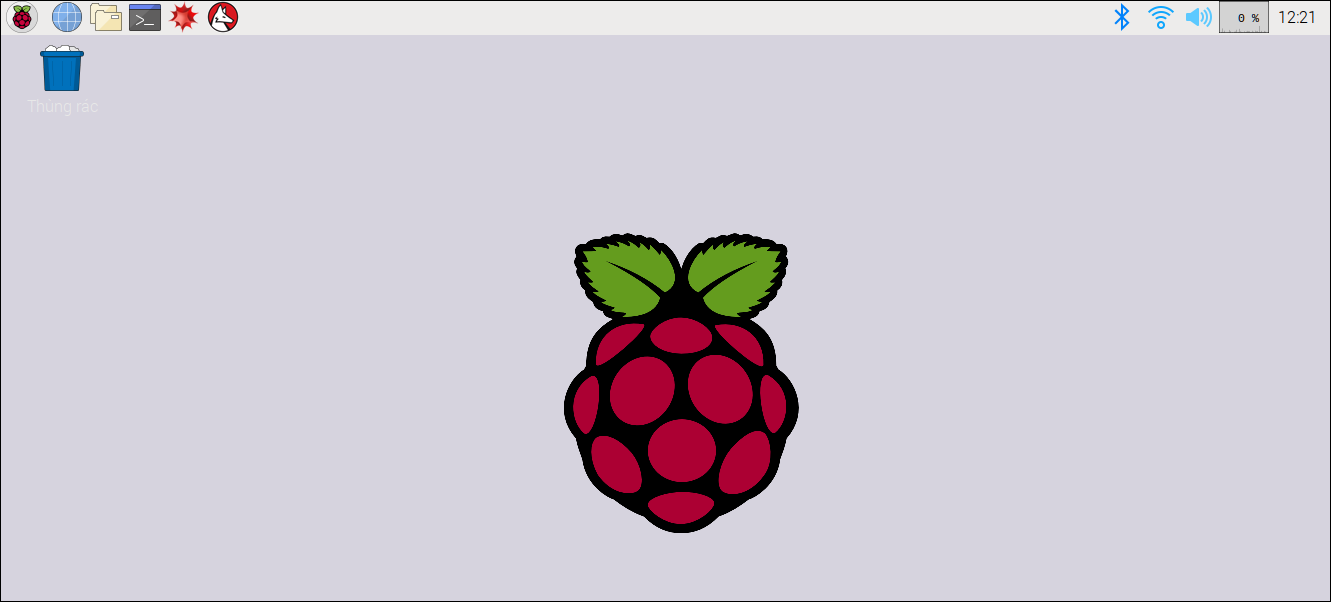


Hình 2.1.3: Sơ đồ chân của Raspberry

### **Hệ điều hành Raspbian.**

Đây là bản build Linux dựa trên nên Debian (Gần giống ubuntu) với giao diện LXDE (thay vì GNOME). Có đầy đủ web browser, media player, tools, etc … Nói chung HĐH này dành cho những người muốn dùng Raspberry Pi như một cái PC. Raspbian có dung lượng sau khi giải nén là khoảng gần 4GB, bạn cần tối thiểu 1 cái thẻ 4GB để có thể sử dụng Raspbian. Tuy nhiên, chúng tôi nghĩ bạn nên sử dụng thẻ tối thiểu 8GB vì bạn cần cài thêm các ứng dụng khác nữa. Raspbian được hướng đến người dùng có mục đích:

* Sử dụng Raspberry Pi như máy tính văn phòng để lướt web, soạn văn bản, check mail và thi thoảng nghe nhạc/xem phim.
* Nghiên cứu phát triển các thiết bị điều khiển tự động.
* Sử dụng như một máy chủ cung cấp các dịch vụ như web, file server, printer server.



Hình 2.1.4: Giao diện raspbian

## **Tổng quan về ngôn ngữ lập trình Python.**

**Python** là một ngôn ngữ lập trình kịch bản (scripting language) do **Guido van Rossum**tạo ra năm 1990. Đến nay thì cộng đồng người sử dụng ngôn ngữ này rất đông, nếu so sánh từ bảng xếp hạng các ngôn ngữ lập trình năm 2017 thì Python đứng trong top 10 ngôn ngữ phổ biến nhất.



Hình 2.2.1: Bảng xếp hạng top 10 ngôn ngữ lập trình phổ biến nhất thế giới năm 2017.

**Python** là 1 ngôn ngữ lập trình mạnh và dễ học. Python có tính hiệu quả rất cao với các cấu trúc dữ liệu và đơn giản nhưng lại rất hữu dụng với ngôn ngữ lập trình hướng đối tượng. Cú pháp Python gọn gàng và có tính năng gõ động. Cùng với tính diễn dịch tự nhiên, làm Python là 1 ngôn ngữ lý tưởng dành cho viết kịch bản và phát triển ứng dụng nhanh chóng trong nhiều lĩnh vực trên nhiều nền tảng khác nhau.

### **Đặc điểm ngôn ngữ lập trình Python.**

Triết lý căn bản của ngôn ngữ python như sau:

* **Đơn giản:** Python là 1 ngôn ngữ đơn giản và tối giản. Đọc 1 chương trình Python có cảm giác như đọc tiếng Anh, mặc dù ở dạng rút gọn. Tính tự nhiên của mã giả trong Python là 1 trong các điểm mạnh nhất của ngôn ngữ này. Điều này giúp cho lập trình viên tập trung vào giải pháp giải quyết vấn đề hơn là việc tập trung vào ngôn ngữ.
* **Dễ học:** Python dễ học vì có cú pháp cực kỳ đơn giản.
* **Miễn phí và mã nguồn mở:** Python là 1 ví dụ của FLOSS (Free/Libré and Open Source Software). Vì vậy, bạn có thể tự do phân phối bản sao chép của phần mềm, cũng như mã nguồn, thay đổi hay sử dụng các thành phần phần mềm trong các chương trình mới. Một trong những lý do Python là ngôn ngữ mạnh vì nó được cộng đồng thường xuyên phát triển và nâng cấp.
* **Ngôn ngữ bậc cao:** Khi sử dụng Python, bạn sẽ không bao giờ phải để ý đến các chi tiết mức thấp như quản lý bộ nhớ cho chương trình của bạn, và vân vân.
* **Khả năng bỏ túi:** Do tính tự nhiên mã mở của Python, Python cũng xây dựng chạy trên nhiều nền tảng khác nhau. Bạn có thể sử dụng Python trên GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE và PocketPC. Bạn còn có thể dùng một nền tảng như Kivy để tạo các trò chơi trên máy tính dành cho iPhone, iPad, và Android.
* **Diễn dịch:** Khi 1 chương trình được viết bằng ngôn ngữ biên dịch (như C hoặc C++) thì nó được chuyển đổi từ mã ngôn ngữ (C/C++) thành ngôn ngữ mà máy tính có hiểu được bằng cách dùng 1 trình biên dịch với các chức năng và các cờ khác nhau. Trái lại, Python không cần biên dịch ra nhị phân. Chương trình viết bằng Python chạy trực tiếp từ mã nguồn. Cụ thể, Python sẽ chuyển mã nguồn thành 1 dạng trung gian gọi là bytecode, sau đó dịch dạng trung gian thành ngôn ngữ mà máy tính có thể hiểu được.
* **Hướng đối tượng:** Python là ngôn ngữ hỗ trợ cho lập trình hướng đối tượng lẫn cả lập trình thủ tục. Nếu so sánh với C++ hoặc Java, Python rất mạnh nhưng lại cực kỳ đơn giản để thực hiện lập trình hướng đối tượng.
* **Tính mở rộng:** Nếu bạn cần 1 đoạn mã chạy nhanh hoặc 1 vài thuật toán đóng, bạn có thể lập trình ở C/C++ và sau đó sử dụng nó cho chương trình Python. Python cho phép tích hợp các chương trình ở các ngôn ngữ khác.
* **Khả năng nhúng:** Bạn có thể nhúng Python bên trong chương trình C/C++ để mang lại các khả năng kịch bản cho người dùng chương trình.
* **Thư viện mở rộng:** Thư viện tiêu chuẩn Python thì rất lớn. Thư viện giúp bạn làm nhiều thứ khác nhau liên quan đến biểu thực chính quy, gieo tài liệu, tiến trình/tiểu trình, database, trình duyệt web, CGI, FTP, email, XML, XML-RPC, HTML, tập tin WAV, mã hóa, GUI, và các phần khác. Tất cả thứ này đều sẵn có khi cài đặt Python.

### **Cài đặt Python.**

Python hỗ trợ hầu hết các hệ điều hành phổ biến hiện nay, ta có thể tải về phiên bản python dễ dàng trên trang chủ python (<https://www.python.org>). với phiên bản mới nhất hiện tại là phiên bản 3.7.0.



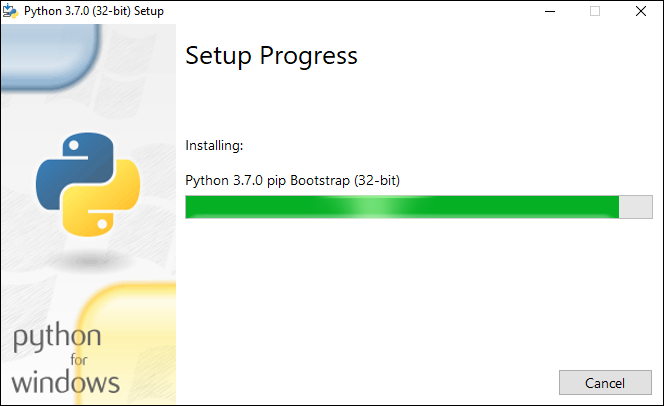
Hình 2.2.2: Giao diện trang chủ python

Tiếp đó ta chạy file vừa tải về và cài đặt như hình 2.2.4:

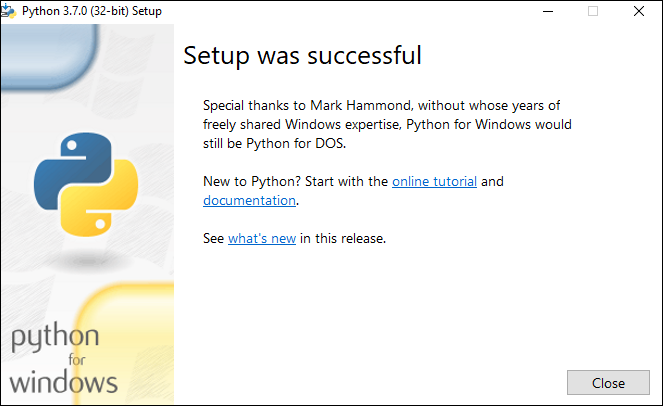


Hình 2.2.3: Cài đặt ngôn ngữ python trên windown

Chọn **“Add Python 3.7 to PATH”**, sau đó chọn **“Install Now”** để bắt đầu cài đặt.

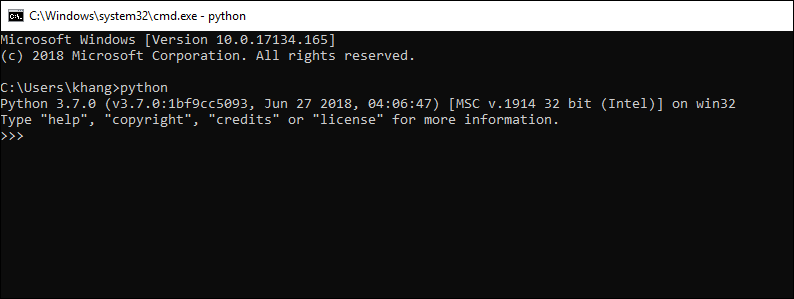


Hình 2.2.4: Quá trình cài đặt python



Hình 2.2.5: Màn hình báo cài đặt thành công

Để kiểm tra mình đã cài đặt thành công, ta vào **“cmd”** và gõ **“python”** như hình 2.2.6:

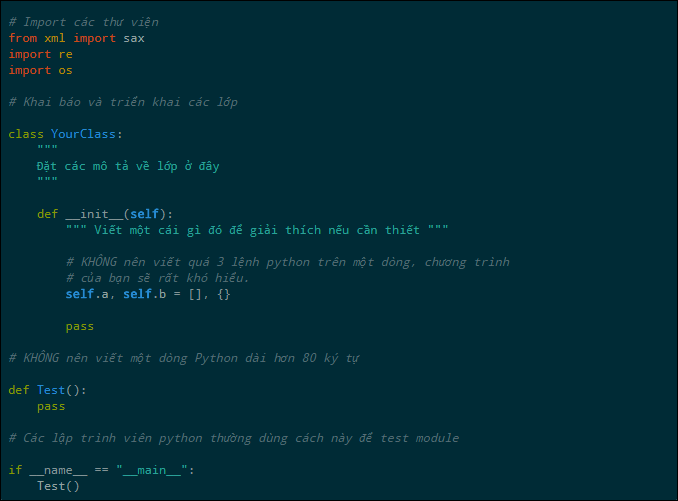


Hình 2.2.6: Thao tác kiểm tra phiên bản python đã dược cài đặt

Như vậy ta đã cài đặt thành công python. Và có thể bắt đầu code với python.

### **Cấu trúc đầy đủ của một chương trình Python.**

Một ví dụ đầy đủ của chương trình trong python như hình 2.2.7



Hình 2.2.7: Cấu trúc đầy đủ của một chương trình trong python

Như hình trên ta thấy, một cấu trúc đầy đủ trên python bao gồm:

* Import các thư viện sẽ sử dụng (nếu dùng những lệnh đơn giản thì không cần phải khai báo, python vẫn hiểu được)
* Khai báo và triển khai các lớp, các hàm sẽ sử dụng.
* Cuối cùng là các lệnh chạy chính được thực hiện từ trên xuống dưới.

Khác với những ngôn ngữ khác, code lệnh nào nếu đúng python sẽ thực hiện lệnh đấy. Nếu sai sẽ báo lỗi, nhưng không báo lỗi toàn bộ chương trình như những ngôn ngữ khác.

**Các loại biến trong Python**

Có 5 kiểu dữ liệu chuẩn là:

* Kiểu Number
* Kiểu String
* Kiểu List
* Kiểu Tuple
* Kiểu Dictionary

Để gán giá trị cho biến trong Python, chúng ta không cần khai báo biến một cách tường minh như những ngôn ngữ khác, khi ta gán giá trị cho một biến bất kì, biến đó sẽ tự động được khai báo.

Ví dụ: a = 12 # a được gán kiểu số nguyên

b = 1.0 # b được gán kiểu số thực

c = “ toi la python” # c được gán kiểu chuỗi

Ngoài ra, còn có các loại toán tử, các lệnh điều hiển luồng và vòng lặp, các loại hàm, module I/O, xử lý ngoại lệ và Python nâng cao. Để tìm hiểu chi tiết, có thể tham khảo thêm tại trang web <https://vietjack.com/python/module_trong_python.jsp>.

Sau khi tìm hiểu sơ lược về ngôn ngữ lập trình Python, ta sẽ sang chương tiếp theo để tìm hiểu về Raspberry pi 3 và lập trình ứng dụng cho nó.

## **Tổng quan về thư viện Kivy.**

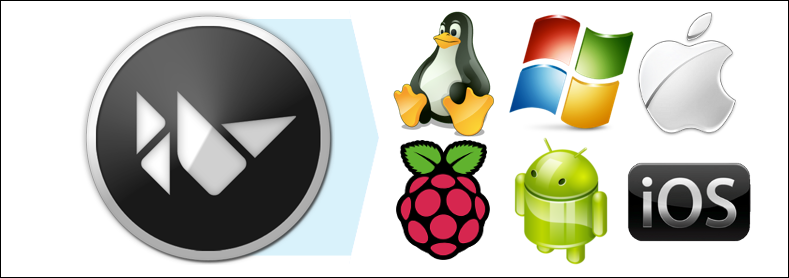
2.3.1 Giới thiệu Kivy

Kivy là một thư viện mã nguồn mở viết bằng Python để phát triển ứng dụng, đặc biệt là giao diện người dùng.

Kivy có thể chạy hầu hết các hệ điều hành hiện nay như Windows, Linux, Mac OS và thậm chí là cả trên Raspberry Pi.

Kivy có tính đa nền tảng, những ứng dụng viết bằng Kivy có thể chạy trên Windows, Linux, OS X, và Android hay iOS.

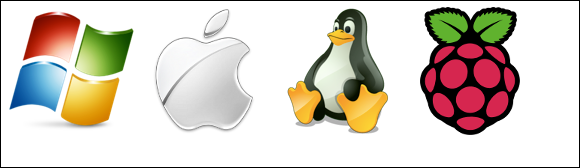
Kivy hoàn toàn miễn phí, có thể sử dụng tạo ra các sản phẩm phục vụ thương mại tùy ý.



Hình 2.3.1: Biểu tượng của Kivy với các nền tảng khác.

2.3.2 Cài đặt Kivy

Với Kivy, có thể sử dụng môi trường phát triển yêu thích của mình để bắt đầu mã hóa ứng dụng. Để bắt đầu, ta cần tải xuống phiên bản Kivy mới nhất: <http://kivy.org/#download>



Hình 2.3.2: Các nền tảng có thể chạy Kivy

2.3.3 Chức năng một số hàm quan trọng.

2.3.3.1 Module kivy.app

Các lớp App là cơ sở cho việc tạo ra ứng dụng Kivy. Nó giống như là Main chính trong vòng lặp của Kivy. Ví dụ ta gọi một Class và tạo ứng dụng, các câu lệnh hay các chức năng bên trong class đó. Sau đó, khi sẵn sàng bắt đầu chạy ứng dụng, ta sẽ gọi lệnh App.run() để chạy chương trình.



Hình 2.3.3: Ví dụ đơn giản về module kivy.app

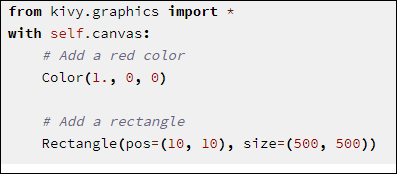
Kết quả trả về như hình 2.3.4:



Hình 2.3.4: Ví dụ tạo một ứng dụng đơn giản

2.3.3.2 Module kivy.graphics

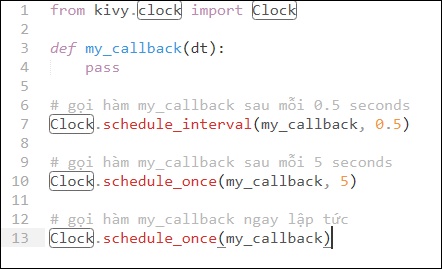
Module tập hợp nhiều hàm được sử dụng để vẽ. Để vẽ trên màn hình bạn xem ví dụ như hình



Hình 2.3.5: Vẽ một hình chữ nhật màu đỏ lên màn hình

2.3.3.3 Module kivy.clock

Đối tượng Clock cho phép lên lịch một cuộc gọi hàm trong tương lai; một lần hoặc nhiều lần trong khoảng thời gian được chỉ định.

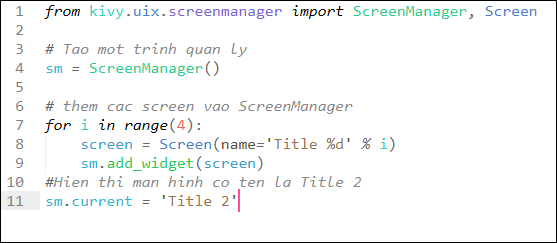


Hình 2.3.6: Ví dụ gọi lại hàm trong khoảng thời gian

Trong hình 2.3.5 là ví dụ cho việc gọi lại hàm thông qua đối số *dt.* Như hình, hàm sẽ được gọi trở lại sau 0.5s, 5s, và luôn luôn gọi lại ngay khi có thể.

2.3.3.4 kivy.uix.screenmanager

Trình quản lý màn hình là một tiện ích dành riêng cho việc quản lý nhiều màn hình cho ứng dụng. Màn hình mặc định chỉ hiển thị từng cái một và sử dụng một TransitionBase để chuyển từ màn hình này sang Màn hình khác. Nhiều chuyển tiếp được hỗ trợ dựa trên việc thay đổi tọa độ / tỷ lệ màn hình hoặc thậm chí thực hiện hoạt ảnh ưa thích bằng cách sử dụng trình tạo bóng tùy chỉnh.



Hình 2.3.7: Ví dụ tạo màn hình cơ bản trong kivy

Theo ví dụ ở trên, mặc định màn hình đầu tiên sẽ được thêm vào ScreenManager để hiển thị. Sau đó lần lượt chuyển sang màn hình khác.

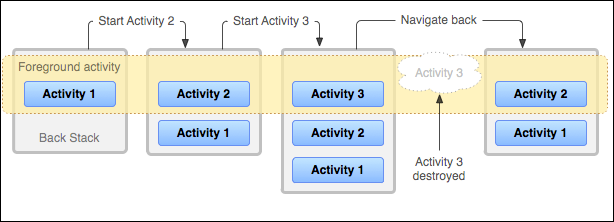
Ngoài ra còn rất nhiều hàm thực hiện các chức năng khác phục vụ nhu cầu tạo một ứng dụng phức tạp. Tham khảo tại trang chủ: <https://kivy.org/doc/stable/api-kivy.html>.

# **CHƯƠNG 3: Xây dựng game 2D chạy trên thiết bị di động.**

## **Vòng đời một Activity chạy trên một thiết bị di động.**

### **Activity trong Android là gì ?**

Activity trong Android được xem là một màn hình tương tác trong ứng dụng Android. Activity chịu trách nhiệm chuyển giao sự kiện cho các view trong nó và quản lý vòng đời (LifeCycle) của nó. Một ứng dụng Android có thể có một hoặc nhiều Activity. Và những Activity này có vòng đời độc lập với nhau và được duy trì trong Activity BackStack.



Hình: Quá trình chuyển đổi qua lại giữa các activity trên android

* Giả sử ta đang ở Activity 1 và từ Activity này bạn chuyển sang Activity 2. Khi đó Activty 1 sẽ được đẩy xuống dưới stack và Activity 2 nằm trên cùng của Stack.
* Từ Activity 2 bạn lại chuyển sang Activity 3 thì tiếp tục Activty 2 lại được đẩy xuống, Activity 3 nằm ở trên stack.
* Ở Activty 3 bạn nhất nút back thì activity 3 được destroy và đẩy ra khỏi stack. Khi đó Activity thứ 2 sẽ được đưa lên top của stack và tương tác với người dùng.

**Vòng đời của Activity**

Một Activity gồm 4 trạng thái chính:

* Nếu activity ở phía trên của màn hình (hay ở trên cùng của ngăn xếp), thì nó đang ở trạng thái **active** (hoạt động) / **running** (đang chạy). Ví dụ khi ta cần gọi điện thì activity bấm số đó đang ở trạng thái active.
* Nếu activity không thể tương tác nhưng vẫn nhìn thấy ( khi mà bị che bởi 1 activity khác nhưng người dùng vẫn có thể nhìn thấy nó ở phía sau ) thì activity này đang ở trạng thái **paused** (tạm dừng). Khi ở trạng thái này activity có thể bị xóa bỏ bởi hệ thống khi thiết bị thiếu bộ nhớ. Ví dụ khi có 1 activity khác dạng dialog hiện lên chỉ che đi 1 phần của activity hiện tại thì activity vào trạng thái paused.
* Nếu activity hoàn toàn bị che khuất bởi activity khác thì nó đang ở trạng thái **stopped** (đã dừng). Activity này vẫn giữ được tất cả trạng thái và thông tin, nhưng không còn hiển thị với người dùng và thường xuyên bị xóa bỏ bởi hệ thống khi thiếu bộ nhớ. Ví dụ khi ta tắt màn hình thì khi đó activity vào trạng thái stopped.
* Nếu activity ở trạng thái **paused** (tạm dừng) hay **stopped** (đã dừng), hệ thống có thể xóa bỏ activity đó khỏi bộ nhớ bằng cách yêu cầu nó tự kết thúc hoặc xóa bỏ tiến trình của nó. Khi acitivty đó hiển thị lại với người dùng thì sẽ được khởi tạo lại và khôi phục lại trạng thái trước đó.

Sơ đồ sau đây minh họa cho vòng đời của 1 Activity cùng với các trạng thái của nó:

Hình : Sơ đồ vòng đời một Activity chạy trên một thiết bị di động.

User điều hướng đến activity

**Game process killed**

OnDestroy()

**Activity shut down**

**Activity running**

OnRestart()

Activity launched

OnCreate()

OnStart()

OnResume()

Một hoạt động khác đến

Activity chấm dứt hoặc bị kết thúc bởi hệ thống

User quay trở lại Activity

OnPause()

OnStop()

Activity không còn hiển thị nữa

Khi game tiêu tốn quá nhiều vùng nhớ của hệ thống

User muốn chuyển hướng đến Activity

Trong sơ đồ trên có 3 vòng lặp chính:

* **Entire lifetime**: xảy ra giữa onCreate() và onDestroy(). 1 activity sẽ cài đặt các trạng thái trong onCreate() và giải phóng toàn bộ tài nguyên trong onDestroy(). Ví dụ có 1 luồng chạy ngầm tải dữ liệu từ trên mạng, ta có thể tạo luồng tại onCreate() và kết thúc luồng tại onDestroy().
* **Visible lifetime**: xảy ra giữa onStart() và onStop(). Trong giai đoạn này, người dùng có thể nhìn thấy activity trên màn hình, tuy nhiên nó không ở trên đầu ngăn xếp và không thể tương tác với người dùng. Giữa 2 phương thức này người dùng có thể lưu trữ được tài nguyên cần thiết hiển thị lên activity cho người dùng. Ví dụ ta cần quan sát thay đổi ảnh hưởng tới giao diện, thì ta có thể đặt Broadcast Receiver tại onStart() và loại bỏ tại onStop().
* **Foreground lifetime**: xảy ra giữa onResume() và onPause(). Trong giai đoạn này activity ở trên đầu ngăn xếp và có thể tương tác với người dùng.

## **Xây dựng mô hình Game với Kivy và Python.**

## **Game 2D Snake - DK.**

Kết Quả

# **CHƯƠNG 4: TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN.**

## **Tổng kết.**

### **Đánh giá tiêu chí cơ bản của một game.**

## **Hướng phát triển.**

# **TÀI LIỆU THAM KHẢO**