Keylogger in Python

Before you can brag that you've executed a keylogger, go through these points to code in Python.

Output using the print() function

```
>>> print("RTFM 2017")
RTFM 2017
```

Assigning value to a variable

```
>>> speed_of_light = 299792458
>>> print(speed_of_light)
299792458
#As you people are seeing, there are no semicolons in Python. That's one thing less for you guys. :P
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Importing Packages

Simply, a **module** is a file consisting of **Python** code. A **module** can define functions, classes and variables. A **module** can also include runnable code. (Basically it's another Python program which you will stitch to your current file.)

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* for importing a module 'module_name.py' has the following syntax:

```
import module_name
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path.

Classes in modules can accessed using dot operator.
Example :

```
module_name.class_name
```

if Statement

Syntax

```
if expression:
   statement(s)
```

If the boolean expression evaluates to TRUE, then the block of statement(s) inside the if statement is executed. If boolean expression evaluates to FALSE, then the first set of code after the end of the if statement(s) is executed.

Example:

```
var1 = 100
if var1:
   print "1 - Got a true expression value"
   print var1
```

Indentation

Python programs get structured through indentation, i.e. code blocks are defined by their indentation. All statements with the same distance to the right belong to the same block of code.

Consider

```
 if var1:
   print "1 - Got a true expression value"
   print var1
print ”abc”
```

[print ”abc” is not within the if statement]

To say simply,instead of typing a pair of braces for a block of code, you have to simply indent your code. The hard work of typing braces has been eliminated. :P

Functions

A function is a block of code that takes in some data and, either performs some kind of transformation and returns the transformed data, or performs some task on the data, or both.

How to create a function:

In Python to create a function, you need to write it in the following manner. Please note that the body of the function is indented by **4 spaces**.

Example :

```
def add_two_numbers(num1, num2):
            #this is a simple function which returns the sum
            result = num1 + num2
return result
```

Calling the function :

Syntax

Function name (arguments)

Example

```
add_two_numbers(1, 2)
```

Classes and objects

Python is an “object-oriented programming language.(OOP)” This means that almost all the code is implemented using using a special construct called classes. Classes are used to keep related things together.

How to create a class

The simplest class can be created using the class keyword.

```
>>> class Snake:
...     name = "python"
...
...     def change_name(self, new_name):    # this is a function belonging to the class
...         self.name = new_name                # access the class attribute with the self keyword
```

Our class Snake has one attribute name and one function change_name.

We refer to the members of an object using the first argument passed to the function.
In this case it is the argument ‘self’ and we access its members using self.member_name.

Creating an object

You can create instances of a class which are called objects.
You need to create an object to use the member functions of the class.
Assume class to be a defined data type and objects are variables that we declare of that data type.
To create an object ‘obj’ of a class ‘abc’ in a package ‘A’ the syntax is:
obj=A.abc()

Adding our own function to an object:
If we need a particular object member 'member' to work as per our will, we need to create a function 'fnctn()' with the block of code to be executed and then assign it to the member of the object using 'obj.member=fnctn'

.start()
Sometimes objects of a particular class need a **kick start** to start working. There might be a group of functions in the class which need to be called in a proper sequence. The .start() function in python is such a function which calls the .run() and associated functions sequentially (you don’t need to know further about this :p).
Syntax:
obj.start()

Opening and Closing Files

The *open* Function

Before you can read or write a file, you have to open it using Python's built-in *open()* function. This function creates a **file** object, which would be utilized to call other support methods associated with it.

Syntax
file_variable = open(file_name , access_mode)

Example:
fo = open("rtfm.txt", ‘a’)

Here are parameter details:

- **file_name:** The file_name argument is a string value that contains the name of the file that you want to access.
- **access_mode:** The access_mode determines the mode in which the file has to be opened, i.e., read, write, append, etc. This is optional parameter and the default file access mode is read (r). a stands for append, w stands for write.

The ***write()*** Method

The *write()* method writes any string to an open file. The write() method does not add a newline character ('\n') to the end of the string, You need to add it separately. −

Syntax

fileObject.write(string)

Example

file_variable.write("A")

This writes the character "A" in the file.

Closing a File

Syntax

file_variable.close()

Ok, Enough with the basics already.

Lets get started by implementing your freshly gained knowledge by making a keylogger.
A keylogger is a type of surveillance software that has the
capability to record every keystroke you make to a log file.
Just explore the py-keylogger folder properly.
Now you need to create a keylogger file with an extension of '.py' using python inside this folder.
Here the actual programming starts and you'll need to use the knowledge from earlier parts of the manual.
First you have to import the pyxhook package to your file. Feel free to scroll up the manual if you forgot.
Now your pyxhook package takes care of accepting the events on keyboard but isn't smart enough to save it in a log file. This is where you come into action.
You need to create a function which would take an argument 'event', open a log file (preferably in the same folder), (give a proper thought on the access mode), write 'event.Key' to it and to keep the log file clean separate each character with a new line(Hope you know the new line character).
Now you need to stop logging the keys at some point of time. Lets keep a definite key to stop it and that would be ` (the grave key ascii=96). You need to close the file after you encounter this key.
So good job in designing the function of key logger. But wait a second. How will the pyxhook package know that it needs to use this function to record the keys. This is where you assign this function to an object created of the class HookManager in the pyxhook package. You need to assign this function to the member function called 'KeyDown'.

Now lets come to the final step.
You need to give the object a kick start (Remember this word?).
Finally you need to tell your object to stop. There's a function called '.cancel()' which can be called for the same. Now where would you call this function? That's for you to decide.