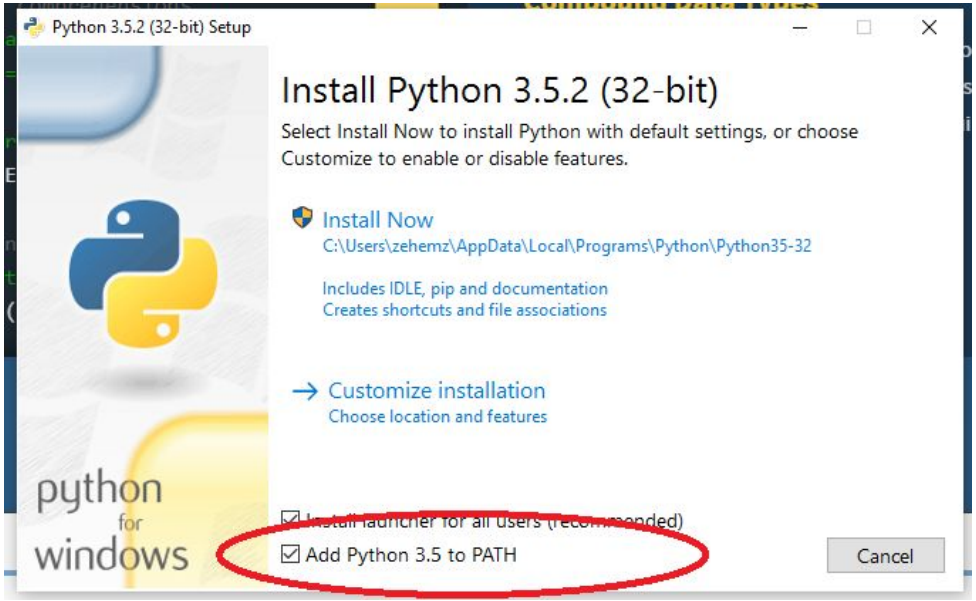
Python 101

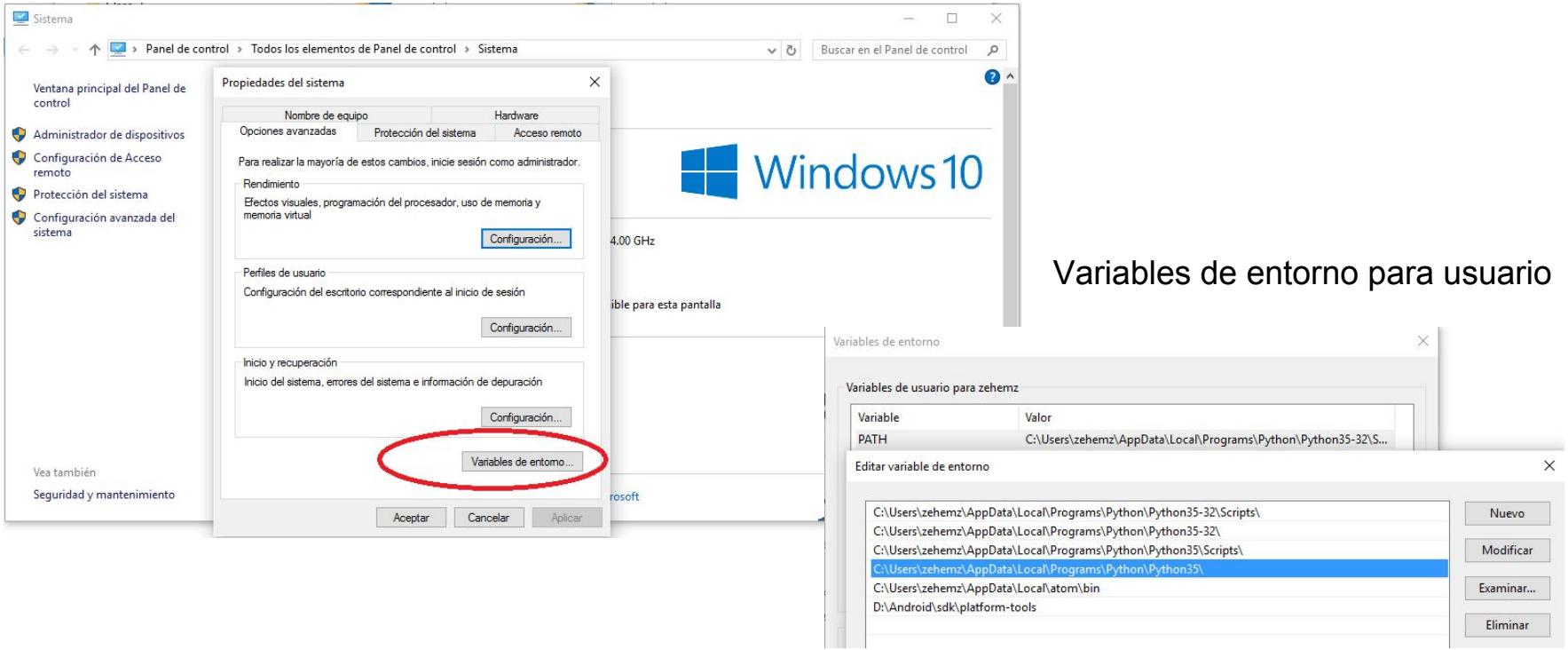
+ Entorno de desarrollo, Tipos de datos, __main__, funciones

Instalar python en windows

Ingresar a python.org -> https://www.python.org/



Variables de entorno



Variables de entorno para usuario

Ide - Eclipse

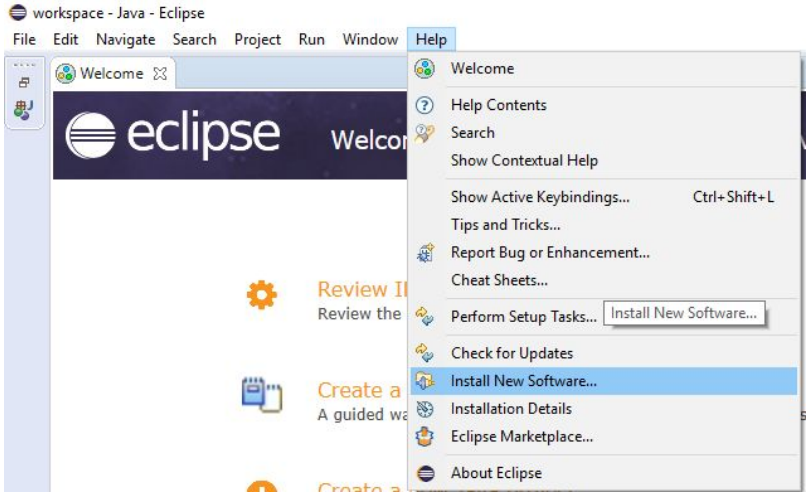
Requiere Java client (eclipse utiliza motor java para funcionar), pueden bajar el JDK de java aquí: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

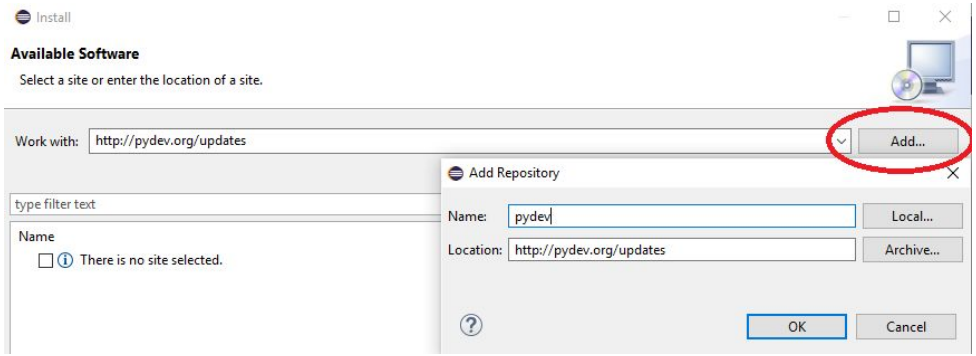
Bajar eclipse installer de: https://eclipse.org/downloads/ (eclipse neon) y seleccionar “Eclipse IDE for Java Developers”

PyDev Plugin para Eclipse

Pydev: http://www.pydev.org/download.html

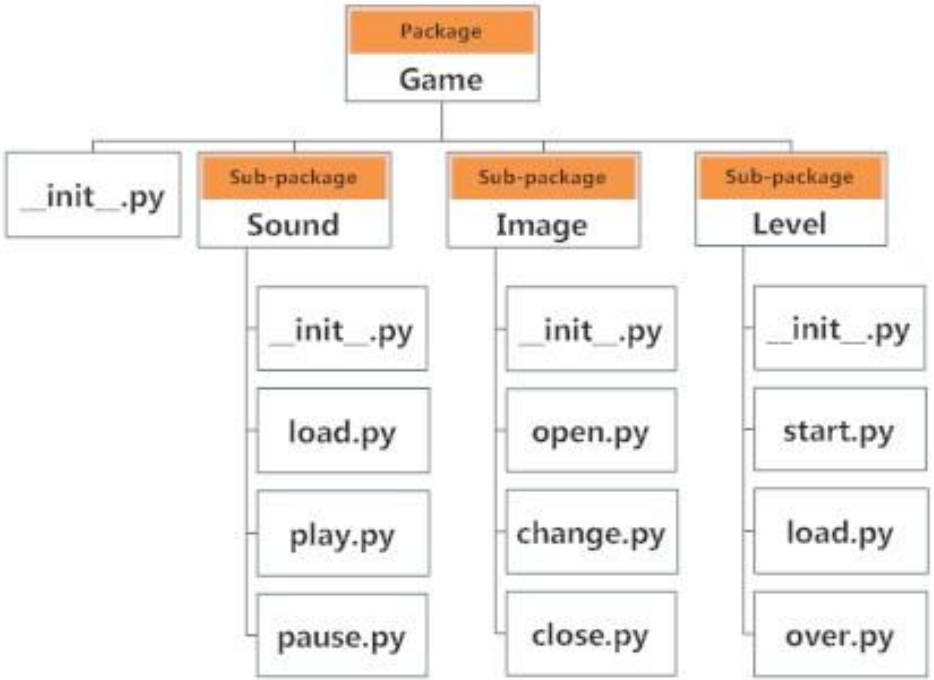
Pydev url para eclipse plug-in: http://pydev.org/updates





Luego de instalar: window->perspective->other->pydev

Módulos - from e import



Concepto de __main__

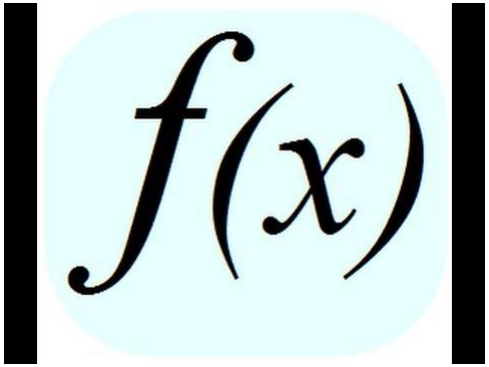
El main es un espacio de escritura en el cual cada archivo .py tiene la posibilidad de ejecutar instrucciones al ser ejecutado, previo a cualquier línea anterior.

```
if __name__ == "__main__":

    … #Código
```

Vamos a la consola!

Funciones



def nombre(param1=”def”, param2, *argL, **argD):

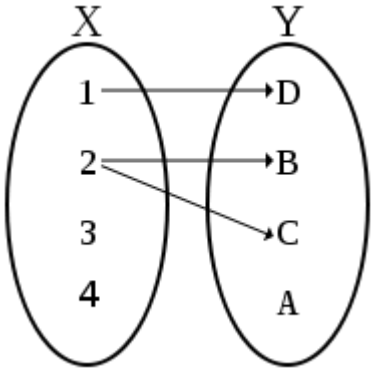
….code

….code

(return - yield?)

nombre(param2=10)

¿cómo se llaman las funciones que no retornan valores? p...



Listas

```
>>> cuadrados = [1, 4, 9, 16, 25]

>>> cuadrados

[1, 4, 9, 16, 25]
```

Diccionarios

```
>>> tel = {'jack': 4098, 'sape': 4139}

>>> tel['guido'] = 4127

>>> tel

{'sape': 4139, 'jack': 4098, 'guido': 4127}

>>> tel['jack']

4098

>>> del tel['sape']
```

Funciones como variables

```
nuevo = nombre
nuevo() -> es la función nombre.
```

Funciones de orden superior

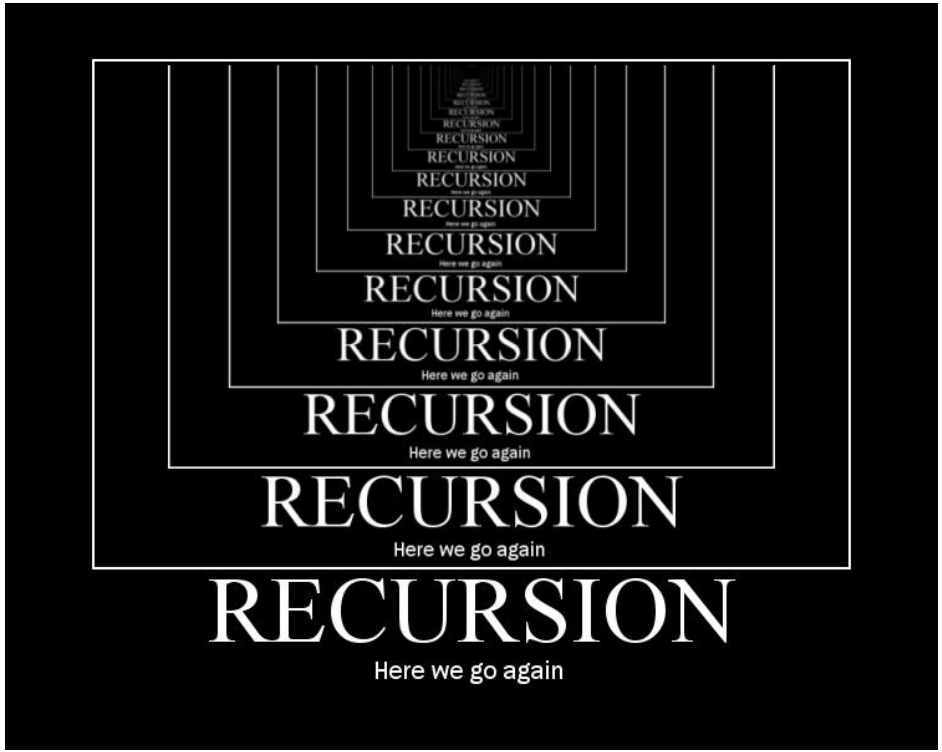
```
def saludar(lang):
     def saludar_es():
          print (“hola”)
     def saludar_en():
          print (“Hi”)
     def saludar_fr():
          print (“Salut”)

     lang_func = {“es”: saludar_es, “en”: saludar_en, “fr”: saludar_fr}
     return lang_func[lang]
```


ARCTIC OCEAN
Queen Elizabeth Islands
GREENLAND (Kalaallit Nunaat) (Denmark)
CANADA
RUSSIAN FEDERATION
UNITED STATES OF AMERICA
NORTH ATLANTIC OCEAN
SOUTH ATLANTIC OCEAN
INDIAN OCEAN
SOUTH PACIFIC OCEAN
BRAZIL
ARGENTINA
THE WORLD
MERCATOR PROJECTION
scale at the Equator

Recursividad

¡Vamos al pizarrón!



Listas por comprensión

Con map:

```
squares = list(map(lambda x: x**2, range(10)))
```

Listas por comprensión:

```
squares = [x**2 for x in range(10)]
[(x, y) for x in [1,2,3] for y in [3,1,4] if x != y]
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```

```
>>> squares = []
... for x in range(10):
...     squares.append(x**2)
```

Generators

Generators are iterators, but **you can only iterate over them once**. It's because they do not store all the values in memory, **they generate the values on the fly**:

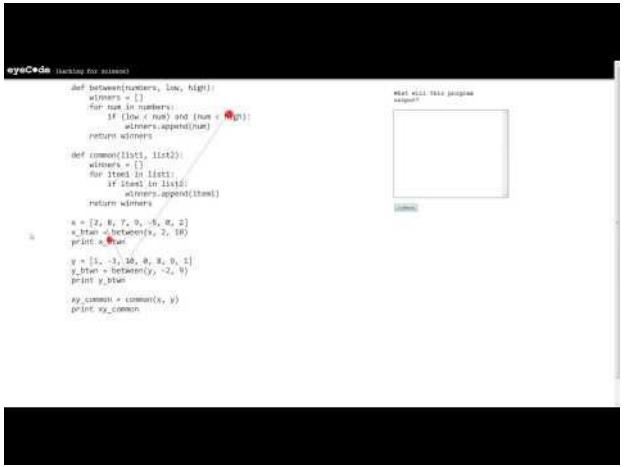
```
>>> mygenerator = (x*x for x in range(3))

>>> for i in mygenerator:
... print(i) 0 1 4
```

Generators

```
>>> def createGenerator():
... mylist = range(3)
... for i in mylist:
    ... yield i*i
>>> mygenerator = createGenerator()
# create a generator
>>> print(mygenerator)
# mygenerator is an object! <generator object createGenerator at
0xb7555c34>
>>> for i in mygenerator:
... print(i) 0 1 4
```

Python style rules: https://www.python.org/dev/peps/pep-0008/



Utf-8 en python2: # -*- coding: utf-8 -*-

Entorno de interprete (para ejecutable, unix): #!/usr/bin/env python

Archivos y OS

¡Vamos a la consola!


That's all Folks!