* **Python’s objects and classes**

(source: http://blog.lerner.co.il/pythons-objects-and-classes-a-visual-guide/)

- **Everything** in Python is an **object** which means that **everything** in python is an **instance** of **a class**. For example, **123** is an instance of **int** and **"abc"** is an instance of **str**.

- That's what we have in other object-oriented languages such as Java.

- However, in Python, our classes are also objects. In our previous example, **int** and **str** are **objects**. So they have also to be instances. They are actually **instances** of class **type**.

* Therefore, in Python **every class** is **an instance** of **type** class (including type itself).

e.g.1,

>>>class Foo(object):

>>> pass

# pass statement **does nothing** particular but can act as a **placeholder**

>>>a=Foo()

**Instantiation:**

**type** indicates from which class this class is **instantiated**.

>>> type(a)

<class '\_\_main\_\_.Foo'>

>>> type(Foo)

<class 'type'>

>>> type(object)

<class 'type'>

**Inheritance:**

But classes **inherit** from other **classes**. To know the class from which a class inherits, use the attribute **\_\_bases\_\_**.

>>> Foo.\_\_bases\_\_

<class 'object'>

>>> type.\_\_bases\_\_

<class 'object'>

>>> object.\_\_base\_\_ # nothing, no base for object.

So, **object** is the base for **every** object

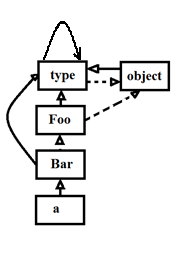
**Object** is an **instance** of **type** and **type** **inherits** from **object**

Every **class** has two pointers:

* one represents its type
* the other represents from which class (object) it inherits.

**Why having two pointers? what is their role in the objects life.**

* Because Foo is an instance of type, **type.\_\_init\_\_** determines what happens to our class when it is created.
* Because Foo inherits from object, **invoking a method on its object a** will result in, first, looking for that method on Foo. If the method doesn’t exist on Foo, then Python will look on object.

Two built in functions **isinstance** vs **issubclass:**

- isinstance(object, classinfo) asks whether an **object** is an **instance** of a **class.**

- issubclass(class, classinfo) asks whether **one class** is a **subclass** of another class - not necessarily immediate subclass.

>>> class Foo(object):

>>> pass

>>> class Bar(Foo):

>>> pass

>>> a = Bar()

>>> issubclass(Bar, Foo)

True

>>> issubclass(Foo, object)

True

>>> issubclass(Bar, object)

True

>>> issubclass(Foo, type)

False

#---------------------

>>>isinstance(Bar, Foo)

False

>>> isinstance(a, Bar)

True

>>>isinstance(a, Foo)

True

>>> isinstance(a, object)

True

>>> isinstance(Bar, type)

True

Bar is a subclass of Foo, not an instance of it.

Bar is an instance of type which is a subclass of object,

therefore the class Bar is an instance of object.

every **class** is an instance of type and every **class** is an instance of object

>>> isinstance(Bar, type)

>>>True

>>> issubclass(type, object)

>>>True

>>>isinstance(Bar, object)

>>>True