

Dynamic Languages

As you know programming languages classified in two way; Dynamic Languages and Static Languages.

In some cases Dynamic Languages has many advantages rather than Static Languages. Python/IronPython and Ruby are Dynamic Languages; C#,Visual Basic are Static Languages.

What makes a language “Dynamic”?

Dynamic Languages that execute at runtime many common behaviors that other languages might perform during compilation, if at all. These behaviors could include extension of the program, by adding new code, by extending objects and definitions, or by modifying the type system, all during program execution. These behaviors can be emulated in nearly any language of sufficient complexity, but dynamic languages provide direct tools to make use of them.

Especially execution of clients at runtime is pretty good for web sites.

A Variable is a named location that stores a value. We can use the variables via the name we give them. These are the rules for naming a variable.

IronPython is dynamically typed, no pre-declaration of a variable or its type is necessary. Variables are created when they are first assigned values. Dynamic Languages are strongly typed languages, not weakly typed.

So if you create a integer variable, you won't verified up it ;)

Will Dynamic Languages provide us any conveniences?

The biggest problem for programmers is variable data type declaration. We can use this feature flexibly. Anyway we will initialize when we assign the declared data type. And also one variable for only that data type. So why do we declare that data types?



“Now! *That* should clear up a few things around here!”

Like this comic character, pre-declaration of data types is like writing names on the existing wellknown objects.

So IronPython does it for us :)



| İbrahim KIVANÇ
| ik@ibrahimkivanc.com
| www.ibrahimkivanc.com


```
def fact(x): return (1 if x==0 else x * fact(x-1))
```

And now, what can we do with this flexibility of IronPython?

```
>>> x = "ibrahimkivanc"
>>> print x      # we assigned x as a string value
ibrahimkivanc

>>> x = 10
>>> print x      #now x is an integer value
10

>>> x = 10.53
>>> print x      #x is a float value
10.53

>>> x = True
>>> print x      #x is a boolean value
True

>>> x=["ipy_array","first","second"]
>>> print x      #now x is an array
['ipy_array ', 'first', 'second']
```

```
>>> x="ibrahimkivanc"
>>> print x
ibrahimkivanc
>>> x=10
>>> print x
10
>>> x=10.2
>>> print x
10.2
>>> x = 10.23
>>> print x
10.23
>>> x = True
>>> print x
True
>>> x = ["ipy_array","first","second"]
>>> print x
['ipy_array', 'first', 'second']
>>> _
```

Here is the flexibility! first we assigned x variable to an integer. As you see we assign 5 different types of data value; string, integer, float, bool and array. We can give many sample like this.

With there is no type validation, especially at XNA game development, it will really help us. Maybe in the future there will be a IronPython entegration for XNA Game development.

If you have any questions or discover any errors / typos please let me know ik@ibrahimkivanc.com

All The Best!



| İbrahim KIVANÇ
| ik@ibrahimkivanc.com
| www.ibrahimkivanc.com