

Part 21 - Documentation

Part 21 - Documentation

 A communicable material used to explain some attributes of an object, system or procedure.

I've saved the most important for last, as documentation is itself, just as important as the code which it describes.

When documenting your code, be sure to remember:

1. All your documents should be in English.
2. Use full sentences.
3. Avoid spelling/grammar mistakes.
4. Use present tense.

Documentation is placed in tripled double-quoted strings right below what you are documenting.

Documentation with Summary

```
def Hello():  
    """Says "hello" to the world."""  
    print "Hello, World!"  
  
Hello()
```

That "docstring" is the least you can do to document your code. It gave a simple summary.

If your docstring spans more than one line, then the quotes should go on their own lines.

You may have noticed that 'Says "hello" to the world.' is not a full sentence. For the first sentence in a summary, you can imply "This member".

Parameters should also be documented.

Parameters

```
def Hello(name as string):  
    """  
    Say "hello" to the given name.  
    Param name: The name to say hello to.  
    """  
    print "Hello, $name!"  
  
Hello("Harry")
```

To read it to yourself, it goes as such: 'Say "hello" to the given name. Parameter name is defined as the name to say hello to.'

This keeps in line with using full sentences.

If describing the parameter takes more than one line, you should move it all to a new line and indent.

Long Parameter

```
def Hello(name as string):  
    """  
    Say "hello" to the given name.  
    Param name:  
        The name to say hello to.  
        It might do other things as well.  
    """  
    print "Hello, $name!"
```

The same goes with any block.

Here is a list of all the tags that can be used

Tag	Description
-----	-------------

No tag	A summary of the member.
Param <i><name></i> : <i><description></i>	This specifies the parameter <i><name></i> of the method.
Returns: <i><description></i>	This describes what the method returns.
Remarks: <i><text></i>	This provides descriptive text about the member.
Raises <i><exception></i> : <i><description></i>	Gives a reason why an <code>Exception</code> is raised.
Example: <i><short_description></i> : <i><code_block></i>	Provides an example.
Include <i><filename></i> : <i><tagpath></i> [<i>@<name>="<id>"</i>]	Includes an excerpt from another file.
Permission <i><permission></i> : <i><description></i>	Describe a required Permission.
See Also: <i><reference></i>	Lets you specify the reference that you might want to appear in a See Also section.

And a list of inline tags

Tag	Description
* <i><item></i> * <i><item></i> * <i><item></i>	Bullet list
# <i><item></i> # <i><item></i> # <i><item></i>	Numbered List
<i><<reference>></i>	Provides an inline link to a reference. e.g. <i><int></i> or <i><string></i> would link.
[<i><param_reference></i>]	References to a parameter of the method.

Here's some examples of proper documentation:

Documentation example

```
import System

class MyClass:
```

```

"""Performs specific duties."""
    def constructor():
        """Initializes an instance of
<MyClass>"""
        _rand = Random()

    def Commit():
        """Commits an action."""
        pass

    def CalculateDouble(i as int) as int:
        """
Returns double the value of [i].
Parameter i: An <int> to be doubled.
Returns: Double the value of [i].
        """
        return i * 2

    def CauseError():
        """
Causes an error.
Remarks: This method has not been
implemented.
        Raises NotImplementedException: This has
not been implemented yet.
        """
        return
NotImplementedException("CauseError() is not
implemented")

    def DoSomething() as int:

```

```
    """
Returns a number.
Example: Here is a short example:
    print DoSomething()
Returns: An <int>.
See Also: MakeChaos()
    """

    return 0

def MakeChaos():
    """
Creates Chaos.
Include file.xml: Foo/Bar[@id="entropy"]
Permission Security.PermissionSet:
Everyone can access this method.
    """

    print "I am making chaos:
$( _rand.Next(100))"

def Execute():
    """
Executes the protocol.
Does one of two things,
# Gets a sunbath.
# Doesn't get a sunbath.
    """

    if _rand.Next(2) == 0:
        print "I sunbathe."
    else:
        print "I decide not to
sunbathe."
```

```

def Calculate():
    """
    Does these things, in no particular
order,
    * Says "Hello"
    * Looks at you
    * Says "Goodbye"
    """
    thingsToDo = ["I look at you.", 'I
say "Hello."', 'I say "Goodbye."']
    while len(thingsToDo) > 0:
        num =
_rand.Next(len(thingsToDo))
        print thingsToDo[num]
        thingsToDo.RemoveAt(num)

[Property(Name)]
_name as string
    """A name""" // documents the property,
not the field

Age as int:
    """An age"""
    get:
        return _rand.Next(8) + 18

```

```
_age as int
```

```
_rand as Random
```

This should give you a good view on how to document your code.

I think Dick Brandon said it best:

Quote: Dick Brandon

Documentation is like sex: when it is good, it is very, very good; and when it is bad, it is better than nothing.

Go on to [Part 22 - Useful Links](#)