

Creative Computing

Introduction to Python, turtle graphics

What have we learned?

- Anyone get the 15-min puzzle solution? :)



- Named after Monty Python
- Simple, general-purpose language
- Modern
- Web-enabled
- Used by many companies
 - Google

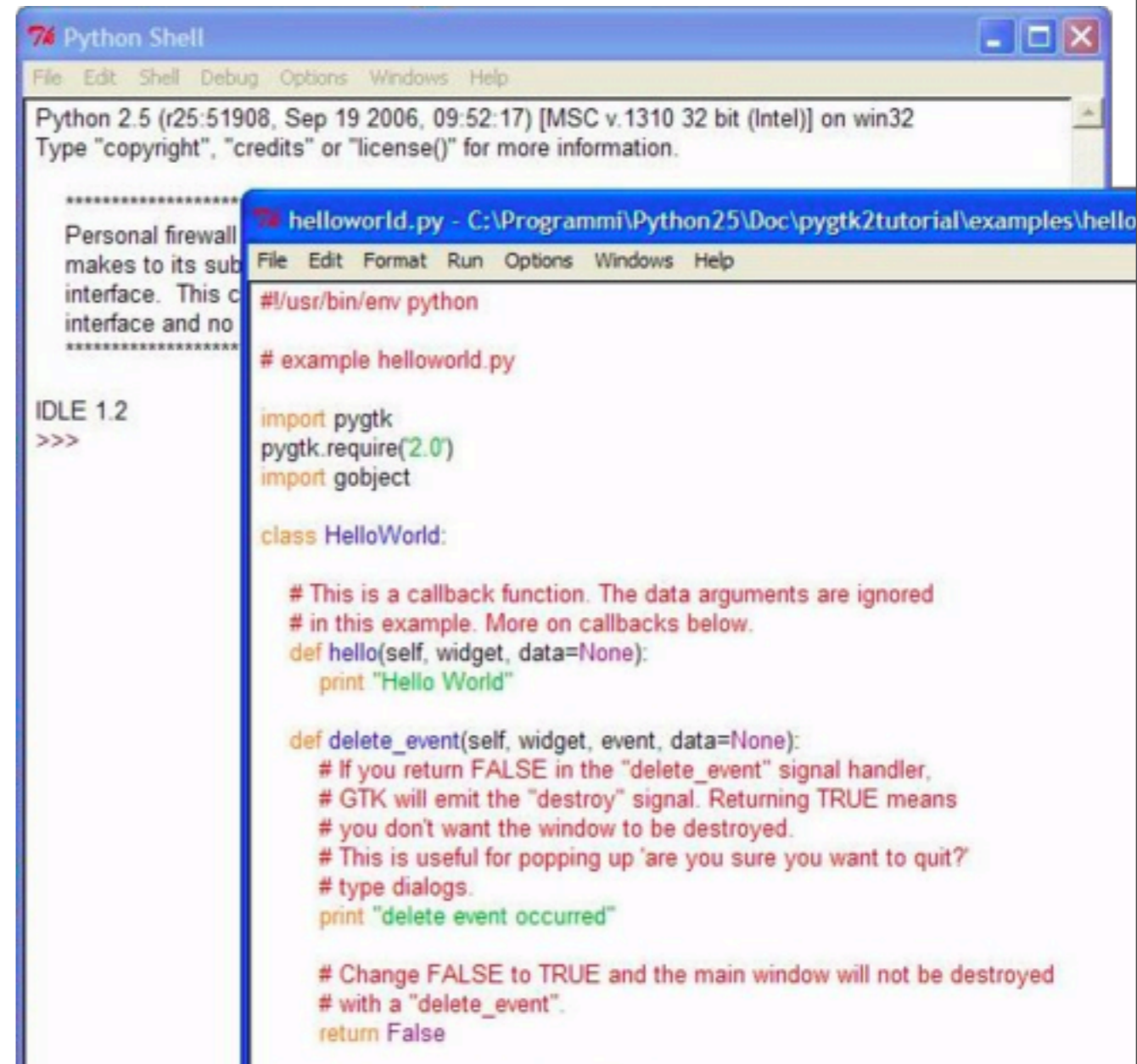


Interpreted

- Means we can type code and get feedback
- Many languages like C, Java must be compiled
- Advantages: great for quick mockups
- Disadvantages: slow

IDLE

- Integrated development environment
- Syntax highlighting
- Calls interpreter



The screenshot shows two overlapping windows from the Python IDLE environment. The background window is the 'Python Shell' with a menu bar (File, Edit, Shell, Debug, Options, Windows, Help) and a text area containing the Python 2.5 version information and a prompt '>>>'. The foreground window is an editor titled 'helloworld.py' with a menu bar (File, Edit, Format, Run, Options, Windows, Help) and a text area containing Python code for a 'Hello World' application. The code includes imports for 'pygtk' and 'gobject', a class definition for 'HelloWorld', and two methods: 'hello' and 'delete_event'. The code is syntax-highlighted with colors: red for comments, blue for class names, green for strings, and black for other code elements.

```
#!/usr/bin/env python
# example helloworld.py

import pygtk
pygtk.require(2.0)
import gobject

class HelloWorld:

    # This is a callback function. The data arguments are ignored
    # in this example. More on callbacks below.
    def hello(self, widget, data=None):
        print "Hello World"

    def delete_event(self, widget, event, data=None):
        # If you return FALSE in the "delete_event" signal handler,
        # GTK will emit the "destroy" signal. Returning TRUE means
        # you don't want the window to be destroyed.
        # This is useful for popping up 'are you sure you want to quit?'
        # type dialogs.
        print "delete event occurred"

        # Change FALSE to TRUE and the main window will not be destroyed
        # with a "delete_event".
        return False
```

Turtle graphics

- A library for drawing simple 2-d graphics
- A good way to start playing with programming concepts
- The top of our programs will have to say

```
from turtle import *
```

Functions

- Functions are pre-defined commands
- They have a name
- We have to call them to use them
- Sometimes they take parameters

Some turtle graphics functions

- `fd(<distance>)` or `forward(<distance>)`
- `bk(<distance>)` or `backward(<distance>)`
- `lt(<angle>)` or `left(<angle>)`
- `rt(<distance>)` or `right(<angle>)`
- `color(<color>)`
- `up()` and `down()`
- `pensize(<size>)`

Your own functions

- Add commands to the Python language
- Statements part of the function must be indented!
- Must call function

```
def <name>():  
    statements
```

Your turn

- Draw a house using turtle graphics
- Divide it into functions (ex: roof(), door())
- Try to embellish it: windows, etc.