**Syntax Errors and Runtime Errors**

**Syntax Errors**

A ***syntax error*** is an error of spelling or grammar. Here are some syntax errors:

def function() # forgot the colon

PRINT ("hello") # forgot the indent, print should be lowercase

34 = x # backwards assignment is not allowed

And this is the same code fixed up:

def function():

print ("hello")

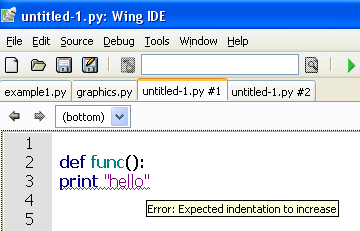
x = 34

Poor English does not trigger an exception:

print ("this spullin is redikulus") # ok

print ("where da party at ?") # still ok

A program will be checked for syntax errors by the interpreter, and will not run until they are fixed. In WingIDE the error is underlined to show the problem, and you can place your mouse pointer over the line and it will state the problem:



In this example, the print statement should have been indented.

In some cases, the error checking by the IDE is very helpful. Sometimes you simply made a spellign mistake (you see that?) and you just have to check your spelling.

You will know that your code is fine if you don’t see any red squiggly underlines.

**Runtime Errors**

Once you fix all your syntax errors, your code may still crash. This is often caused by a ***runtime*** ***error***. A runtime error is an error made when the interpreter tries to execute an impossible statement. The interpreter will indicate the line that it crashed at, and give an error message. Here is a common example of a runtime error:

a = 3

b = 4

print (a,b,c)

In this example, there is no spelling mistake. The last line tries to print out three variables, but only two of them are defined. The interpreter spits out an error message:

NameError: name 'c' is not defined

**Try... Except**

Sometimes the program crashes and it isn't your fault. Maybe the disk drive is full or your users are incompetent. For example, try this simple code:

x = int(input("Enter a number: "))

print ("Thank you! You chose a lovely number!")

This code compiles fine - no syntax errors, no runtime errors - but what if the user is a buffoon ? What if the user enters a non-integer?

Try running the code again, but enter "hello" and see what happens.

We can avoid crashing our program by using a try...except code block, like this:

try:

x = int(input("Enter a number: "))

print ("Thank you! You chose a lovely number!")

except:

print ("Fool! I said a number!")

In this case, we **try** to get the input, and if for some reason it doesn't work out, the program jumps to the **except** block with suitable output. We could even insert the try...except block inside a loop:

good = False

while good==False:

try:

x = int(input("Enter a number: "))

print ("Thank you! You chose a lovely number!")

good = True

except:

print ("Fool! I said a number!")

good = False

Since users have been making mistakes since 1066, and maybe even before that, it is a good idea to use the **try...except** block to avoid them from crashing everything.

**Keywords**: ***Syntax error, runtime error, try...except***