

Wing IDE (Python) Tutorial

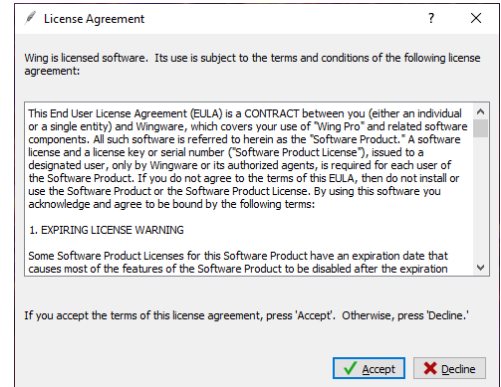
Wing is an IDE (Integrated Development Environment) and is packed full of features, but it does take some initial setting up.

Start Wing in the usual way.

(Find it near the bottom of the Windows 10 designed-for-tablets interface

Or Search for 'Wing')

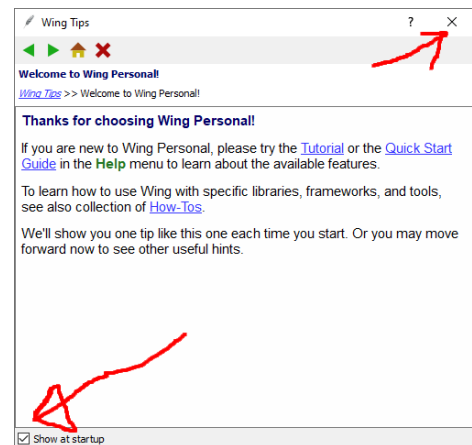
Click Accept on the License Agreement



Wing will open with this dialog

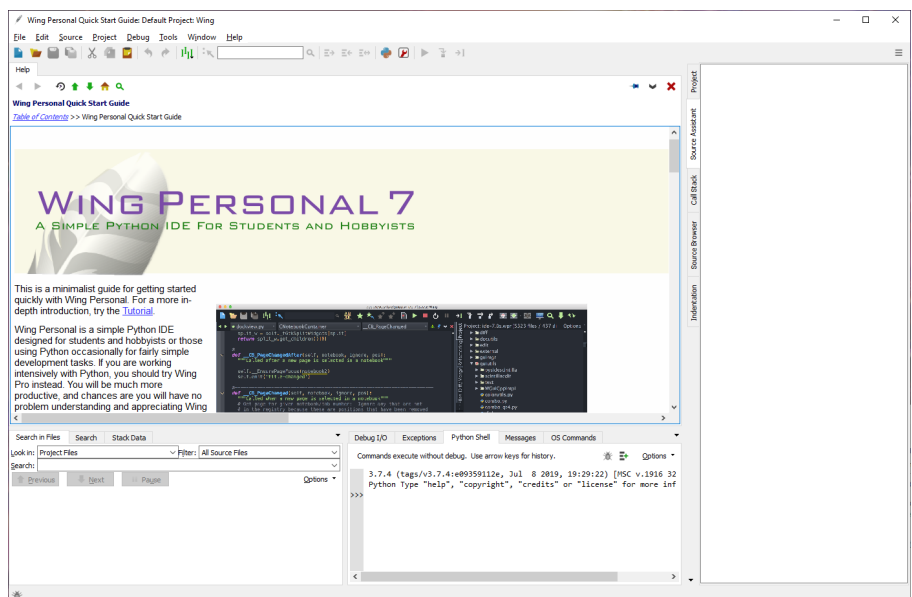
Un-tick 'Show at startup.'

Close the dialog.



Initial startup window

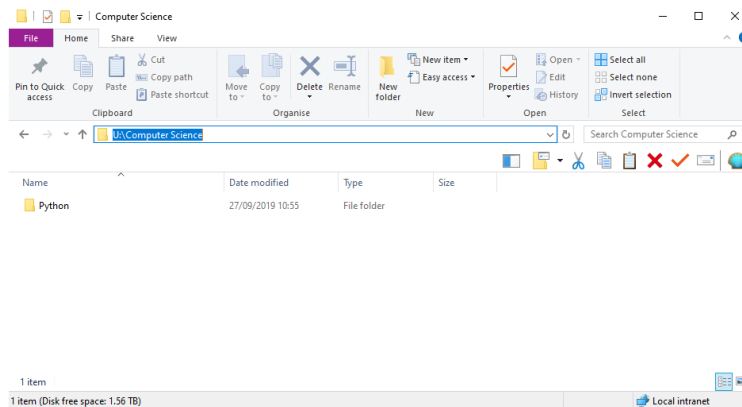
You can close the help page shown here by clicking the red cross.
(2/3 across the window, under the toolbar)



You will be using Wing to open your entire Python folder, not just individual files. If you have been subjected to the horrible 'Idle' experience, this will be a foreign concept to you.

If you have not already done so, create a Python folder somewhere in your user space. (You can minimise Wing to give you more desktop space to open Windows explorer and create your folder.)

This screenshot is **not** from a school PC, so has been created to represent your 'U:' drive, a sub-folder called 'Computer Science' and a folder called 'Python' using Windows FileSystem Explorer

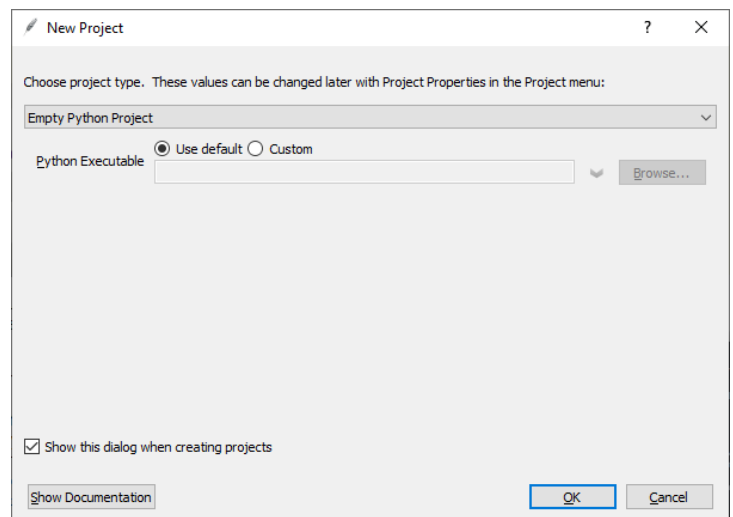


From the Wing window choose:

Menu:Project → New Project

Nothing needs changing here.

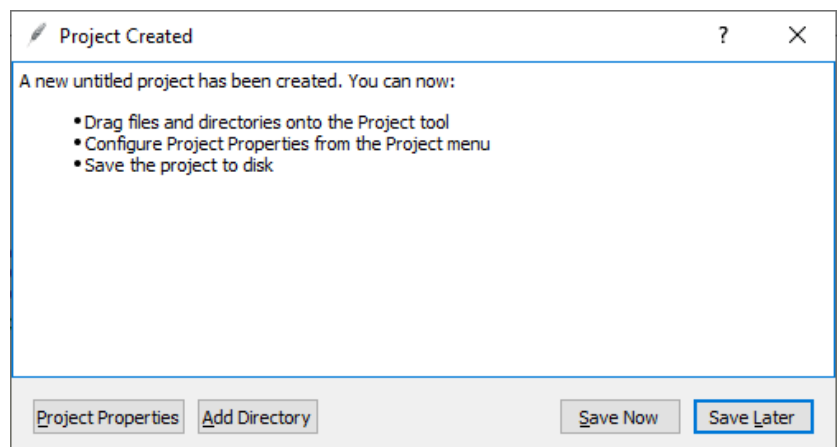
Click OK



This dialog appears

Click the 'Save Now' button

If you get warnings about 'System', or 'Permissions' etc, etc with red warning boxes, just click OK.

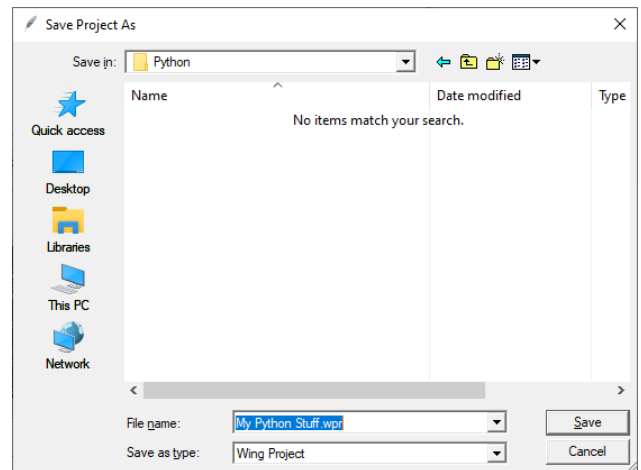


The file browser dialog will open:

Find your 'Python' folder.

Give the project a name:
(Probably NOT 'My Python Stuff')

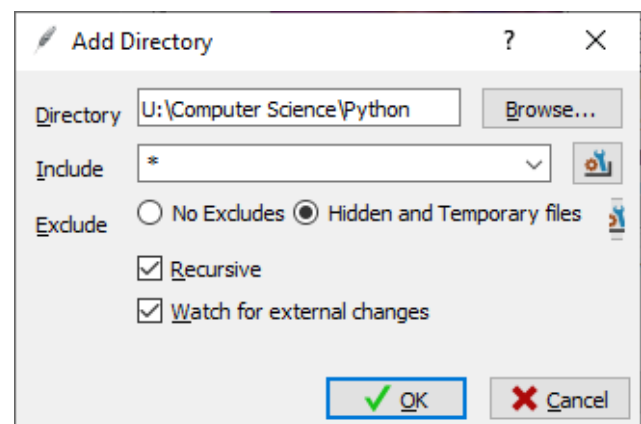
Click 'Save'



Menu: Project → Add Existing Directory

It should already be set to your Python folder
This will add any existing files or sub-folders
to your project

Click 'OK'

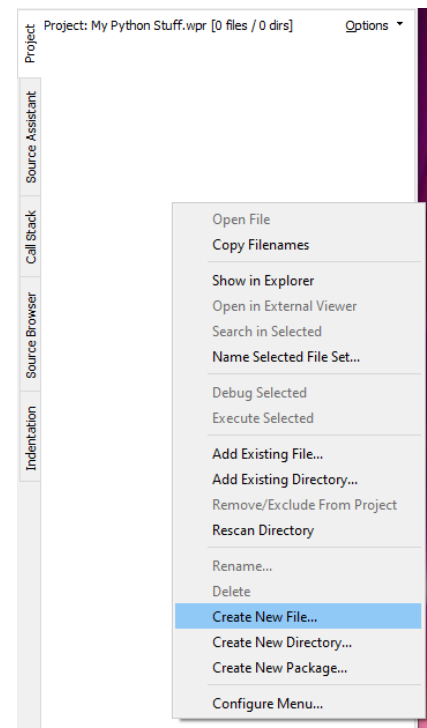


If you already have python files in that folder
they will show up on the right side in the Project tab.

(This one is empty.)

To create a new file right-click in the project tab
and select 'Create New File'

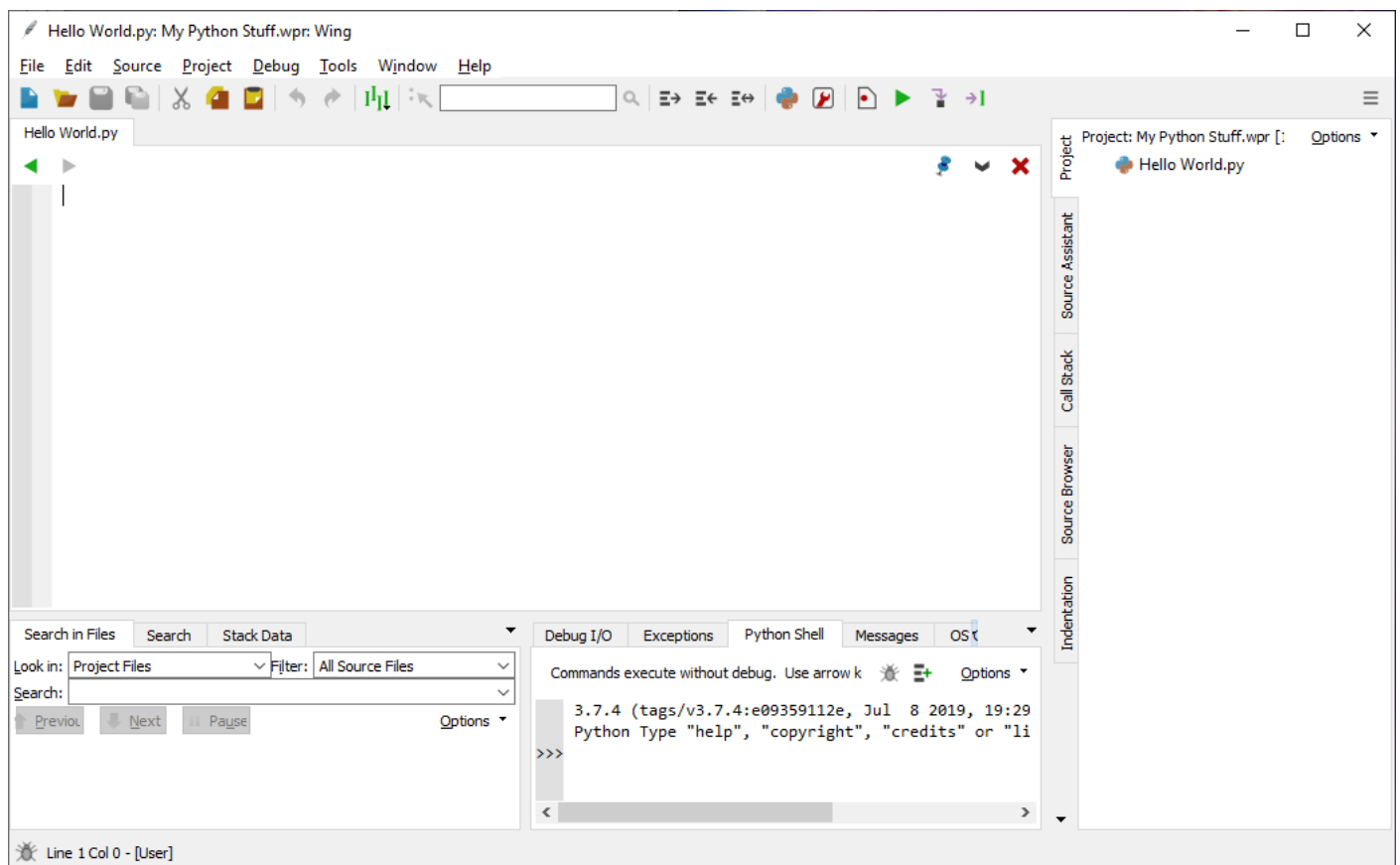
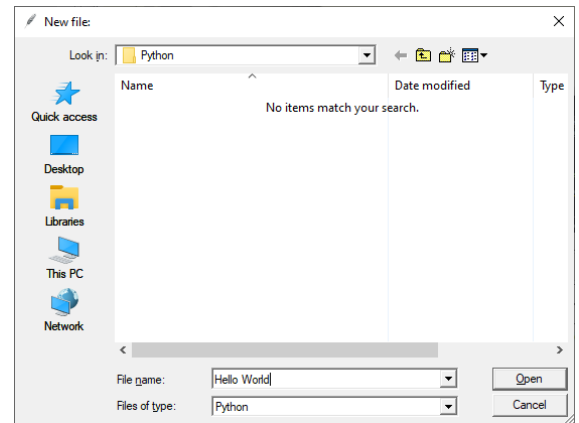
The File browser will appear



Type the name of your file:

(‘Hello World’ in this screenshot)

Click ‘Open’



Note the file showing in the Project tab on the right (Hello World.py)

Note the same file is open for editing with the cursor waiting on the left side.

You can add existing or new files / folders by right- clicking in the Project Tab.

Before starting to type your code, Change your Preferences:

1. Line numbers: **Menu → Edit → Show Line Numbers**

2. Indentation: This is the curse of Python in general, and Idle in particular:

Menu → Edit → Preferences...

Select Editor

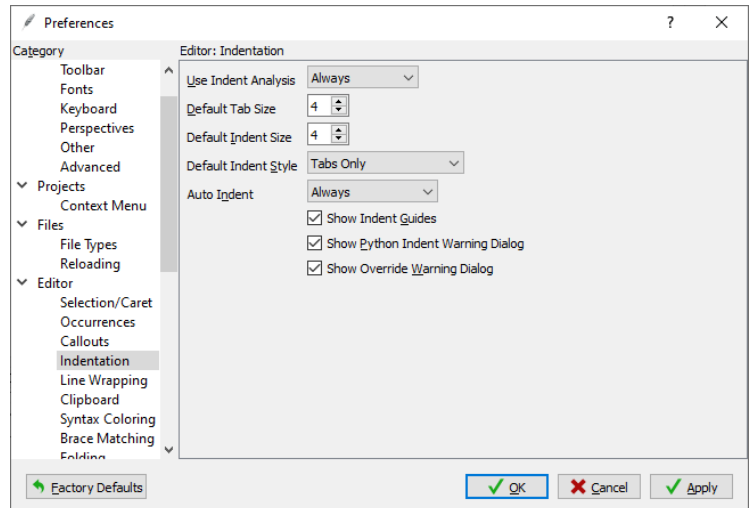
Select Indentation

Default tab size 4

Default Indent size 4

Default indent Style Tabs Only

Click: Show Indent Guides



This will give you the easiest options for indents, and give the least amount of grief.

When indenting your code always use the Tab key, not spaces

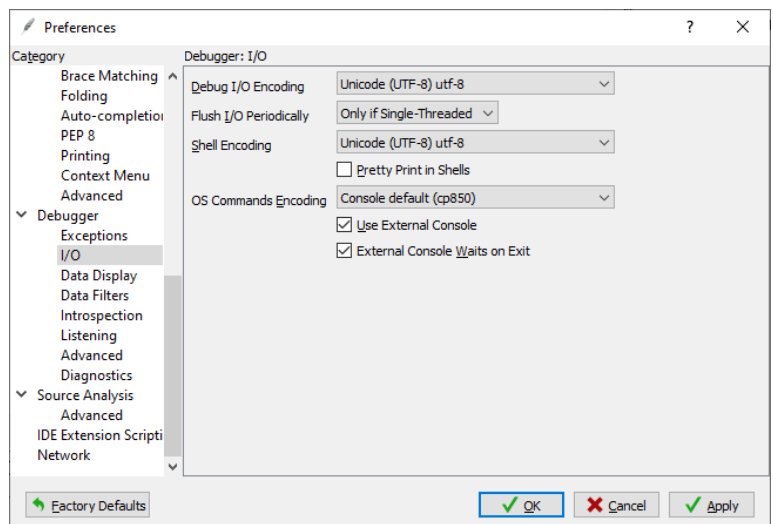
While preferences is open:

Select Debugger

Select IO

Click: Use External Console

Click: External Console Waits on Exit.



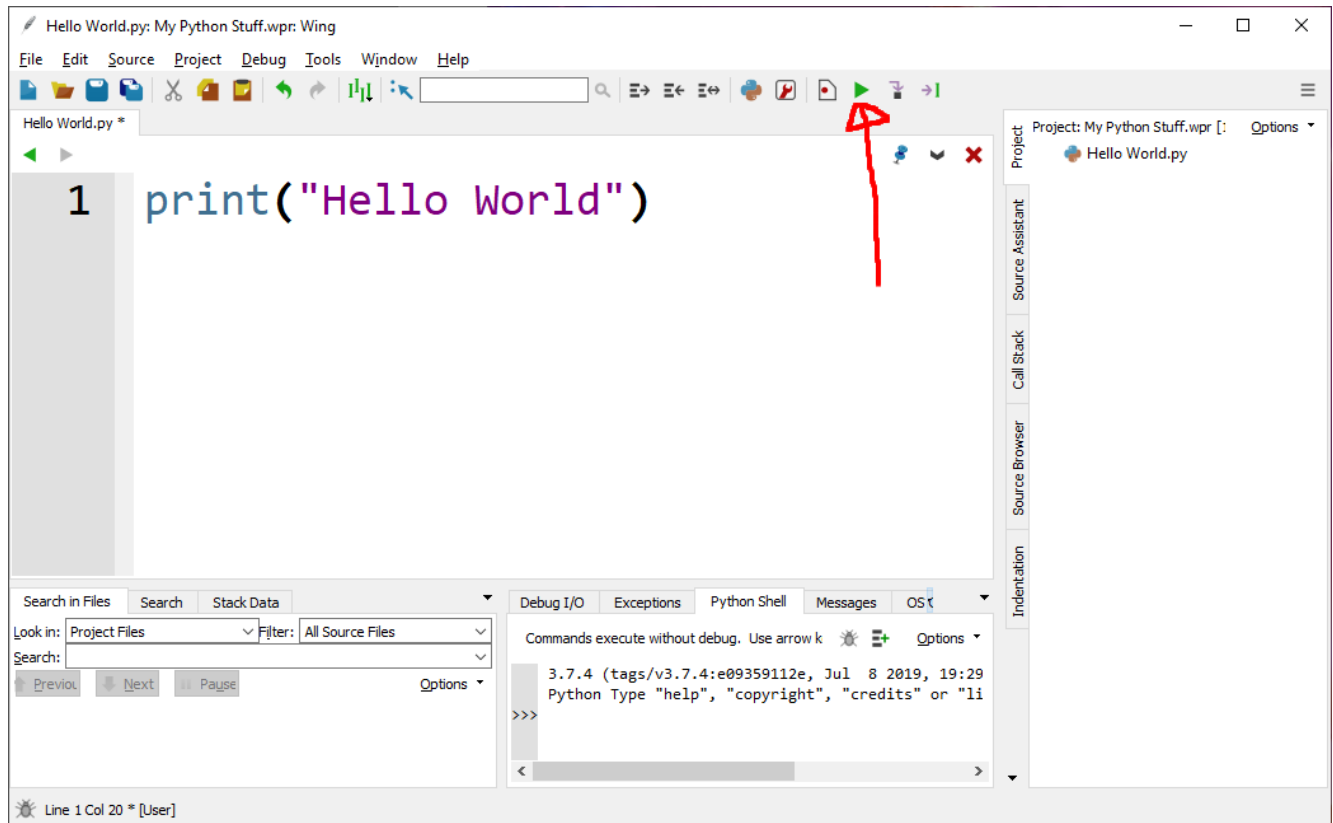
Click OK

You can now edit and run your Python scripts.

Use L-Ctrl key along with + or - keys to change the font size in the editor if required.

Write your epic script and **click the green triangle** to run it:

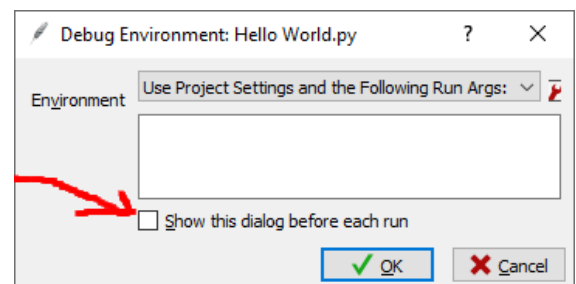
(The example shown here is obligatory when learning a new programming language)



This dialog is a nuisance.

Un-tick 'Show this dialog before each run'

It will not appear any more (but only for this file)



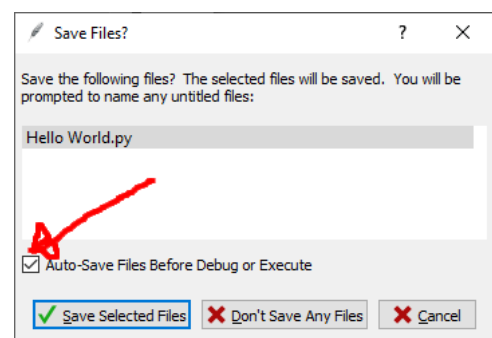
Click OK

This appears every time you run after an edit.
It is also a nuisance.

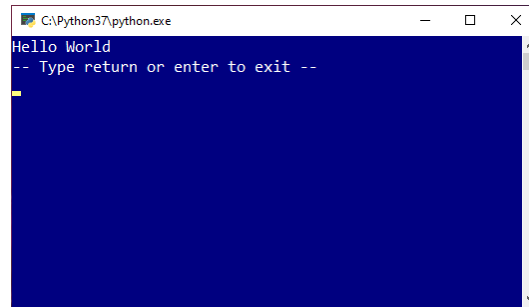
Tick the box Autosave Files

It will not appear any more (but only for this file)

Click 'Save selected Files'



Success

A screenshot of a Windows command prompt window titled 'C:\Python37\python.exe'. The window has a black background and white text. It displays 'Hello World' on the first line, followed by '-- Type return or enter to exit --' on the second line. A yellow cursor is visible on the third line.

The problem is these settings are saved to:

`C:\Users\<username>\AppData\Roaming\Wing Personal 7`

which is a local folder on the computer you are currently using. If you fight with your classmates next visit and get to use the same one again, all your settings are preserved.

There is a better way.

Why not transfer your settings to your User space and then copy them back to the next PC you work on?

There is a Python script here:

`V:\Student Space\Computing Club\Wing Launcher and Settings copier\CopySettingsToU.py`

Copy this file to your Python folder and run it inside Wing when you have completed setting up your preferences.

It will copy all your preferences to your user space .

There is a second script in this folder, which you can copy to your user space:

`V:\Student Space\Computing Club\Wing Launcher and Settings copier\LaunchWing.py`

When you use another computer run `LaunchWing.py` by double-clicking it.

Do NOT run it from inside Wing, as it is trying to over-write the settings file already open in Wing and it will error.

Alternative: Open it in Idle and run it from there.

This will copy your settings onto local AppData, then start Wing, which will use the newly copied settings. As a bonus, It may be faster than trying to find 'Wing' in the Windows 10 interface!

The contents of these two files are shown at the end of this tutorial. They use some advanced Python techniques, and you are not expected to understand them yet!

So why go to all this trouble? What is there to be gained from using Wing?

1. Multiple files can be edited, allowing copy/paste between them, and when you start using multi-file programs, you can switch between the files quickly and easily.
2. Instant indentation error fixing.
3. Auto-complete
4. Excellent debugging

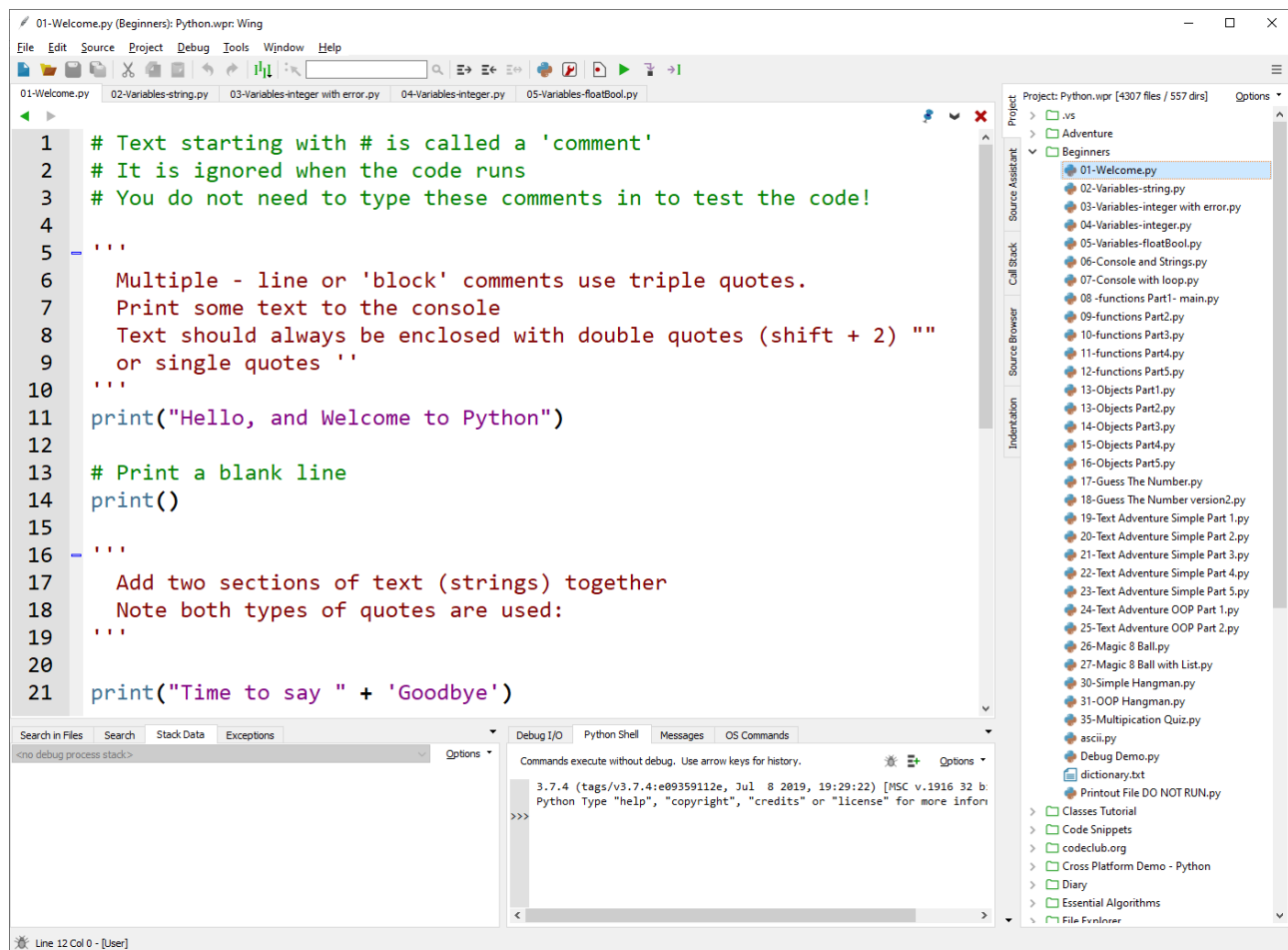
1. Multiple file editing

This screenshot shows multiple folders and files in the Project Tab

5 files have been opened by double-clicking them in the Project Tab

The first file '01-Welcome.py' has been selected for editing.

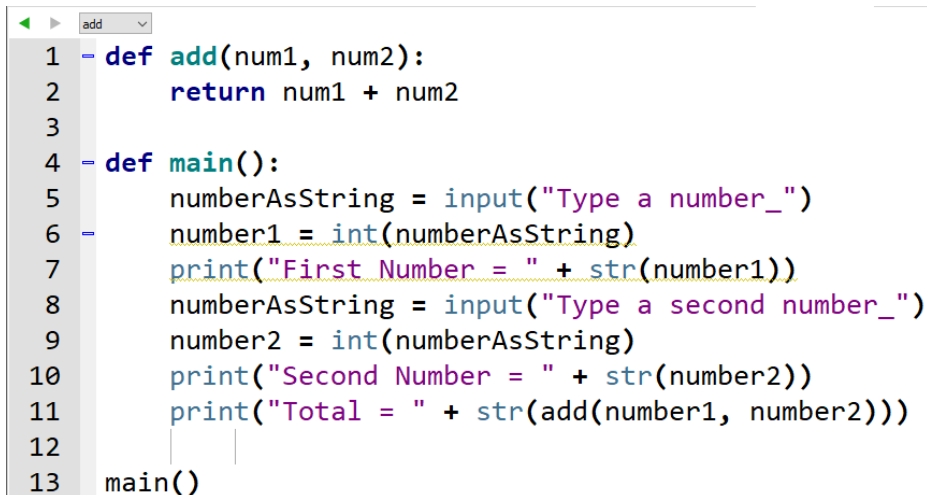
You can copy lines from any file and paste in another



2. Instant indentation error fixing

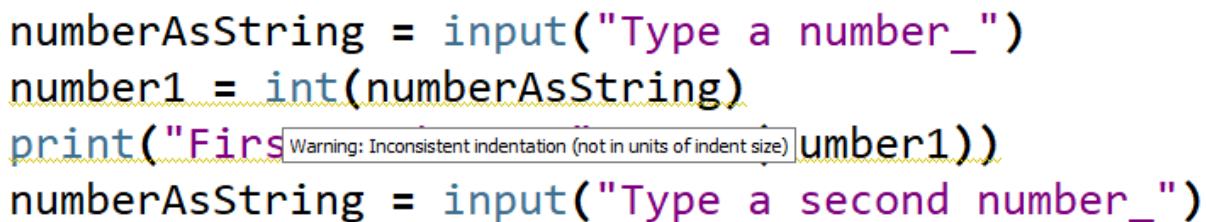
If you look carefully at the screenshot below, you can make out faint yellow lines under the text on lines 6 and 7

This is because the indentation is a mixture of 4 space characters and one tab character.



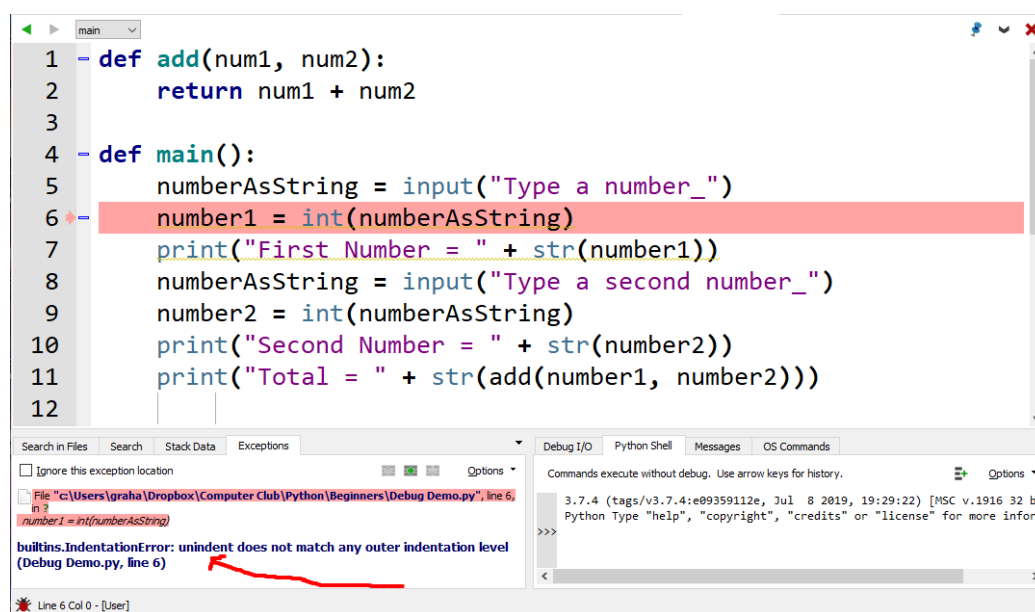
```
1 - def add(num1, num2):
2     return num1 + num2
3
4 - def main():
5     numberAsString = input("Type a number_")
6     number1 = int(numberAsString)
7     print("First Number = " + str(number1))
8     numberAsString = input("Type a second number_")
9     number2 = int(numberAsString)
10    print("Second Number = " + str(number2))
11    print("Total = " + str(add(number1, number2)))
12
13    main()
```

When you hover the mouse over the yellow line you get a tooltip:



```
numberAsString = input("Type a number_")
number1 = int(numberAsString)
print("First Number = " + str(number1))
numberAsString = input("Type a second number_")
```

Finally when you try to run the code you get the usual error Idle throws at you all the time:



```
1 - def add(num1, num2):
2     return num1 + num2
3
4 - def main():
5     numberAsString = input("Type a number_")
6     number1 = int(numberAsString)
7     print("First Number = " + str(number1))
8     numberAsString = input("Type a second number_")
9     number2 = int(numberAsString)
10    print("Second Number = " + str(number2))
11    print("Total = " + str(add(number1, number2)))
12
```

Search in Files Search Stack Data Exceptions Options

File "c:\Users\graha\Dropbox\Computer Club\Python\Beginners\Debug Demo.py", line 6, in number1 = int(numberAsString)

builtins.IndentationError: unindent does not match any outer indentation level (Debug Demo.py, line 6)

Debug I/O Python Shell Messages OS Commands

Commands execute without debug. Use arrow keys for history.

3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 b Python Type "help", "copyright", "credits" or "license" for more information]

>>>

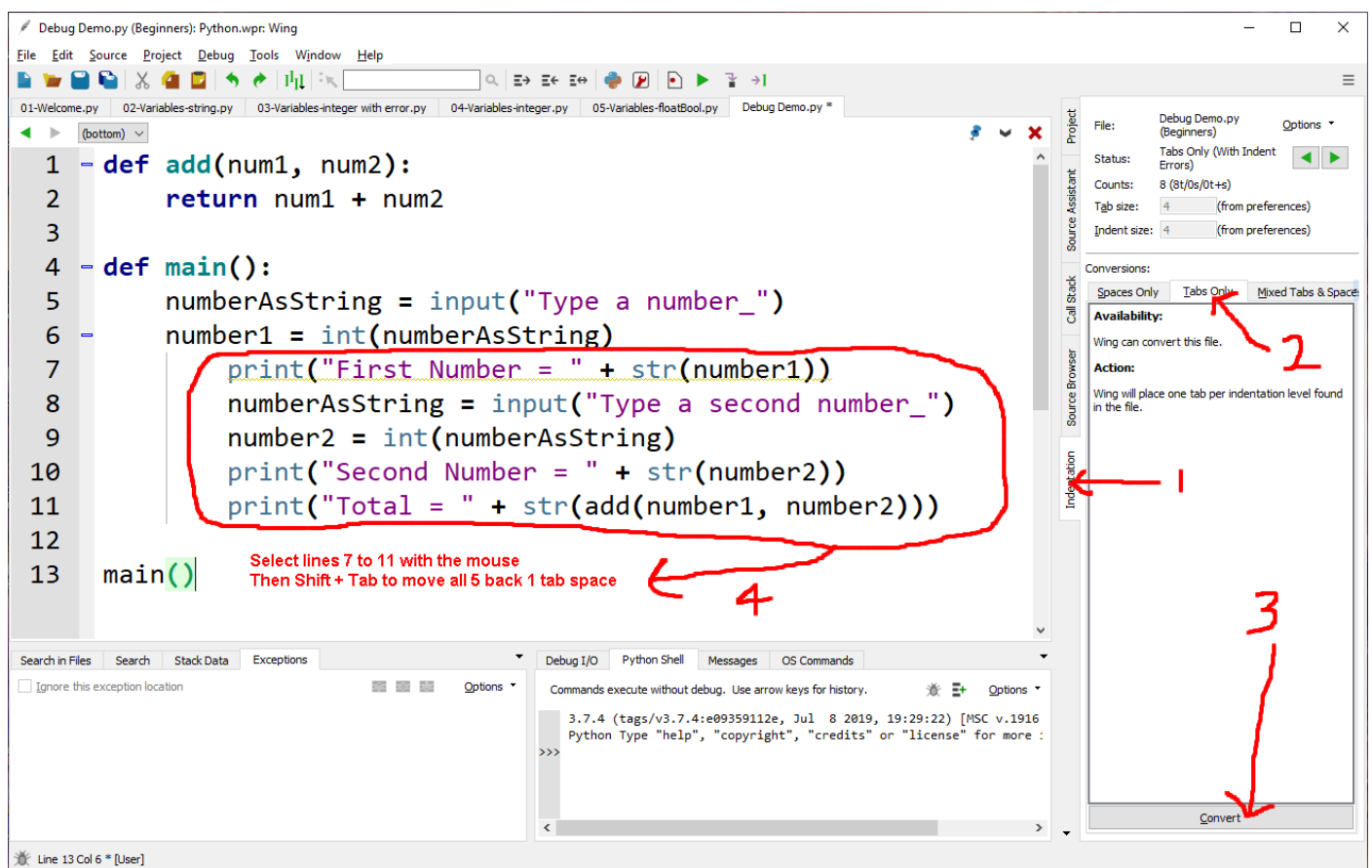
Line 6 Col 0 - [User]

Fixing the error:

1. Select the Indentation Tab
2. Select the Tabs Only Tab
3. Click 'Convert'
4. Depending how messed-up the original file was, some manual correcting may be needed, as here.
Select all the lines out of place with the mouse.

To move them to the right, hit the Tab key

To move them to the left hit Shift + Tab

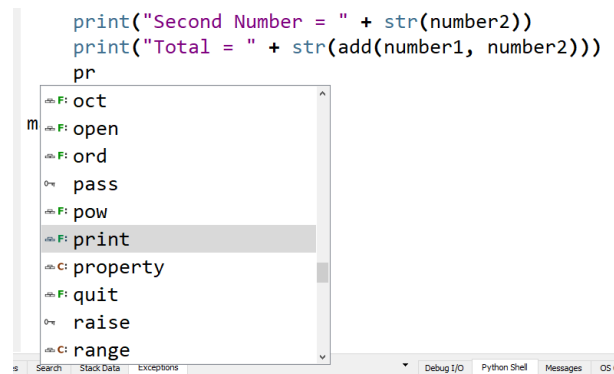


3. Auto-Complete (Intellisense)

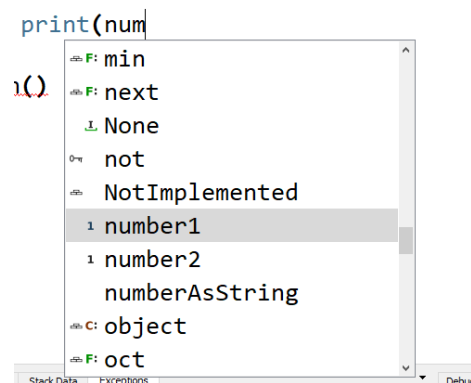
Using the above example, start a new line and type 'p' followed by 'r':

The intellisense has chosen 'print' as the most likely term required. Hit the Tab key to auto-complete

If it was not the correct word, use mouse or arrow keys to select the correct entry



The variable `numberAsString` has already been defined, so to complete the print statement, use an open left bracket, followed by 'num'. The first 3 matches are returned:



Down arrow twice, Tab key and auto-complete is done.

Debugging

To stop the script running at a selected place, click in the grey margin next to a line number to insert a red dot (a break point).

Select the tab 'Stack Data' in the lower left pane

Select the 'Call Stack' tab on the right pane

If you start the program with F5 or the green triangle it will run until it reaches the break point.

For this example start by hitting F7 (or Menu → Debug) so it goes to the beginning (Line 1) and waits there:

```
1 - def add(num1, num2):
2     return num1 + num2
3
```

Line 1 (as expected) Hit F7 again

```
1 - def add(num1, num2):
2     return num1 + num2
3
4 - def main():
5     numberAsString = input("Type a number_")
```

Line 4 NOT Line 2. The interpreter saw a function definition, noted it's presence, and moved on. F7

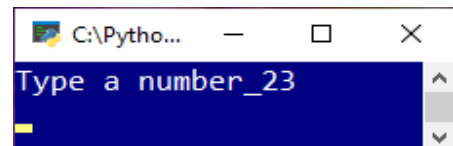
```
12
13 - main()
14
```

Line 13, The end of the script. Not line 5. As before a definition has been noted and ignored. Hit f7

```
4 - def main():
5     numberAsString = input("Type a number_")
```

Finally a line of code on line 13 'called' the function main() and was sent to the first line of code inside the function. Hit F7

As this will run the code containing input(), you will need to interact with the console, type a number, eg 23 and hit return.



```
5     numberAsString = input("Type a number_")
6     number1 = int(numberAsString)
```

Line 6. The variable numberAsString has been given a value of 23. Look in the 'Stack Data' pane:

Hit F7 again to move to line 7. This will execute line 6

Line 6 converts the input string '23' into a number 23

Unfortunately the Stack Data does not distinguish between strings and numbers.

Hovering the mouse does:

Search in Files		Search	Stack Data	Exceptions
★ main(): Debug Demo.py, line 6				
Variable		Value		
▼ locals		<locals dict; len=1>		
numberAsString		23		

Search in Files		Search	Stack Data	Exceptions
★ main(): Debug Demo.py, line 7				
Variable		Value		
▼ locals		<locals dict; len=2>		
number1		23		
numberAsString		23		

Note the Stack Data output:

The variables `numberAsString` and `number1` are both listed as 23, but the first is a string, the second a number

If you hover the mouse over `numberAsString` you get '23' : (quotes round the characters 23 therefore a string)

```
numberAsString = input("Type a number_")
number1'23': int(numberAsString)
```

If you hover over `number1` you get 23: (no quotes therefore a real number)

```
number1 = int(numberAsString)
print23("First Number = " + str(number1))
```

This exercise showed how the interpreter reads your script, and follows the program flow between functions (and at a later date: classes).

This ability to trace variables step by step is invaluable when de-bugging your program.

CopySettingsToU.py

```
import os
import shutil
import errno

def copy(src, dest):
    retValue = True
    try:
        shutil.copypath(src, dest)
    except OSError as e:
        # If the error was caused because the source wasn't a directory
        if e.errno == errno.ENOTDIR:
            shutil.copy(src, dest)
        else:
            print('Directory not copied. Error: %s' % e)
            retValue = False

    return retValue

def main():
    # Check if Wing has been used on this PC by the current user
    sourceDir = os.path.join(os.getenv('APPDATA'), "Wing Personal 7")
    print("Checking Source " + sourceDir + "...")
    if os.path.isdir(sourceDir): # Settings found
        print("Source found")
        destDir = "U:\\My Settings\\Wing Personal 7"
        print("Checking Destination " + destDir)
        if os.path.isdir(destDir):
            print("Destination found, over-writing...")
            shutil.rmtree(destDir)
        else:
            print("Destination not found, Creating...")

        if copy(sourceDir, destDir):
            print("Your Wing settings have been copied from this computer to your user space 'My Settings' folder :)")
            print("\nWhen starting Wing in future, use WingLauncher.py by double-clicking it")
            print("\nThis will copy your settings to this computer and start Wing")
            print("\nIf you need to start again, delete the folder My Settings\\Wing Personal 7 from your account")
            print("Start Wing normally, re-do all the settings, then run this file again")
        else:
            print("Errors prevented your settings from being copied :(")
    else:
        print("No Wing settings found on this computer. Start Wing, change preferences, then run this file again")

main()

input("Press Enter") #Needed if started by double-clicking the file and running it outside of an IDE
```

LaunchWing.py

```
import os
import shutil
import errno
import subprocess

# NOTE This will error if you have Wing already running!!!!
def copy(src, dest):
    retValue = True
    try:
        shutil.copytree(src, dest)
    except OSError as e:
        # If the error was caused because the source wasn't a directory
        if e.errno == errno.ENOTDIR:
            shutil.copy(src, dest)
        else:
            print('Directory not copied. Error: %s' % e)
            retValue = False

    return retValue

def main():
    sourceDir = "U:\\My Settings\\Wing Personal 7"
    destDir = os.path.join(os.getenv('APPDATA'), "Wing Personal 7")

    print("Checking Source " + sourceDir + "...")
    if os.path.isdir(sourceDir): # settings saved to user account
        print("Saved settings found in 'My Settings'")
    print("Checking Destination " + destDir + "...")
    if os.path.isdir(destDir):
        print("Existing Wing settings found")
        if input("Do you want to over-write existing Wing Settings on this computer?").lower() == "y":
            print("Over-writing Wing settings...")
            shutil.rmtree(destDir)
            if copy(sourceDir, destDir):
                print("Your Wing settings have been copied from your user space 'My Settings' to\nthis computer :)")
            else:
                print("Errors prevented your settings from being copied :(")
        else:
            print("You do not have Wing settings saved on your user space")

    print("\nLaunching Wing Personal 7...")
    subprocess.Popen(['C:\\Program Files (x86)\\Wing Personal 7.1\\bin\\wing-personal.exe'])
    sys.exit(0)

main()
```