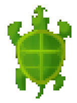
Starting Python and Turtle

Python and Turtle are free tools, created by hackers like you for hackers like you.

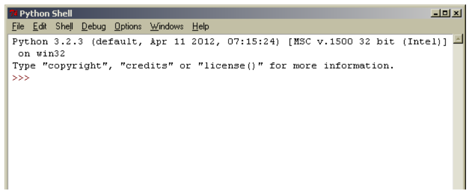
These are some notes to get you started; There is lots of experimenting and poking around you will need to do yourself!

- Meet Turtle. You'll be making Turtle move , draw lines and run around in loops using a language called Python



- Python is how you tell Turtle what to do. We're using Python here to move Turtle, but Python can do a lot more. It can write games, do your homework , make websites work, feed the dog... Actually, Python can't feed your dog, but it can do all the other stuff

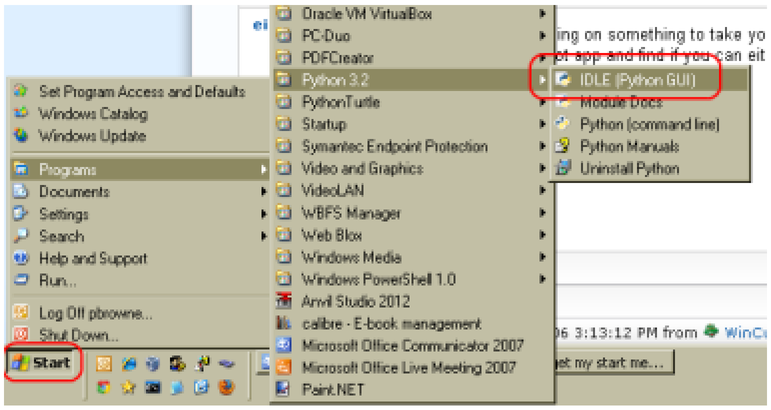
- Python IDLE is the playground where you type the Python commands, so you can do all the cool stuff.



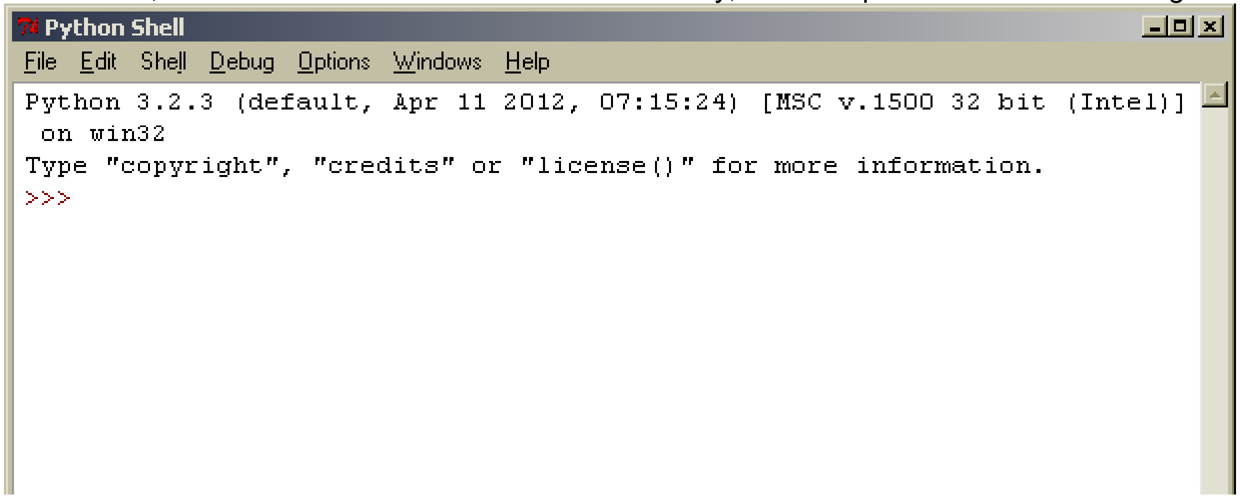
```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

1.Getting a Copy of Python and IDLE to play with

- Download Python (which includes Python IDLE) from this website - there should also be a copy on the USB key that this guide came on. http://www.python.org/download/
- Follow the usual instructions to install it.
- If everything has gone well, you should be able to run Python from the Windows Menu



2. Starting it up Python Idle
- Click on the Python IDLE icon. You should see the something like the picture.
- Yes, it's meant to be as bare as this! Don't worry, we'll fill up the screen soon enough.



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

***Try it out!** Take a look around. Try pushing buttons and click on things. Find out what they do. Tell your friends to see who has found the coolest.*

3. Tour of the Python IDLE Screen

Have a look at the top of the screen. Some important Menu items are
- File – to open, close and save programs that you write
- Edit – help you write those programs quicker. Use it to copy/ paste/ change code you've already written – no more typing again and again.
- Help – can you guess what that does?

In the main bit of the screen is the prompt **>>>>** where the commands that you type will go.

***Try it out!** Can you find out how to ….*

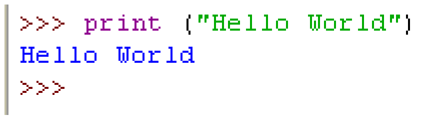
- *Type some text. Press the enter Key (the big key on the right hand side) - don't worry yet about the strange messages that you get!*
- *See if you can copy / paste this text.*
- *See if you find the guide to Turtle in the help (hint – look underHelp … Python Docs. Then click on 'Global Module Index' and search for Turtle*

4. Making Python Say Hello

It's traditional for computer hackers to write a Hello World program when they're getting started.

Type the following, then press enter (the big key on the right of the computer).

```
print("Hello World")
```



```
>>> print ("Hello World")
Hello World
>>>
```

What happens? Turtle will say 'Hello World' back to you - it's the bit in blue on the screenshot on the right,

- It's pretty simple; You type the commands, Python does them.
- Unfortunately 'clean my room' doesn't work, but a lot of other stuff does. As long as you remember to get the spelling right.

Watch out!
Python is cool, but he can be a bit silly sometimes. If you don't tell him exactly the right command he'll tell you a message like this one.

```
SyntaxError: invalid syntax
```

What Python means is; I don't understand , try again.

- Remember to check your spelling, check that you have spaces correct and check that you have round brackets **(**, and check that you have the quote **"** marks in the right place.

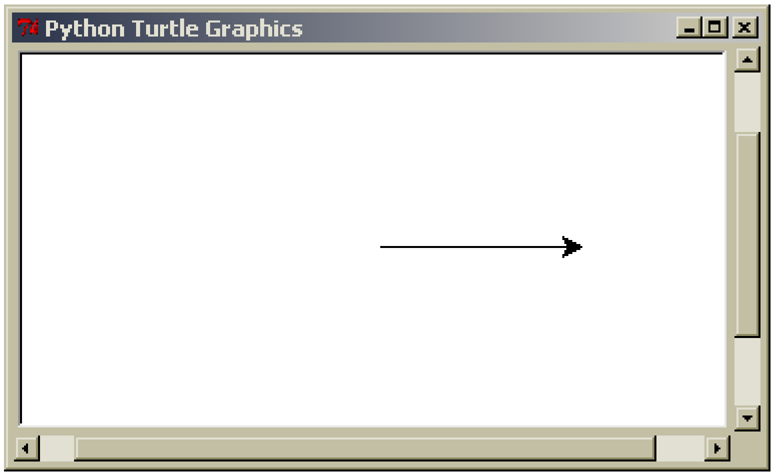
Try it out!
- *Can you make Python say your name instead?*
- *Can you make Python do your Maths homework?*
  - *Hint:* `print (2+2)`
  - *Hint: use * to multiply and / to divide*

5.Where's Turtle? Lets do some drawing

Are you looking for Turtle? Let's wake him up.

Try typing these two commands and press 'Enter' (the big key on the right of the keyboard)
What happens?

```
from turtle import *
forward (100)
```

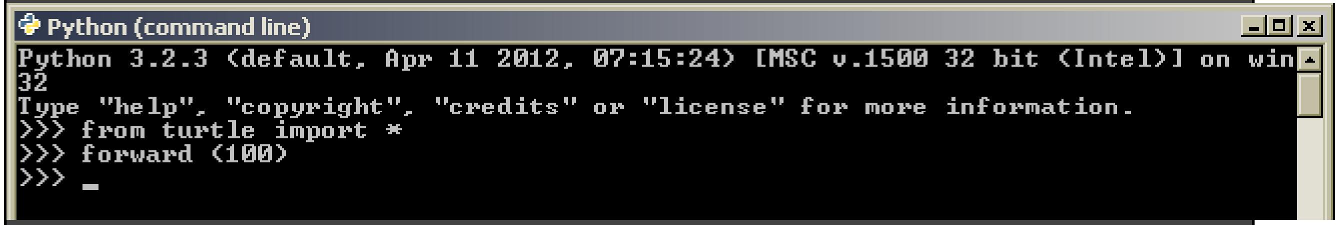


- The first line tells Python IDLE you want to use Turtle. You only need to type once every time you open Python. If you don't type it, Python won't know that you want to talk Turtle!
- Hint: Turtle can be a little bit shy, so you may need to look around on your computer to find the 'Python Turtle Graphics' Window

Getting Stuck!
Turtles are a little bit slow in cold weather, so sometimes the Turtle window will 'freeze' and not respond to what you type. Actually, this is a bug in Python that is easy to get around.

Here are two things to try

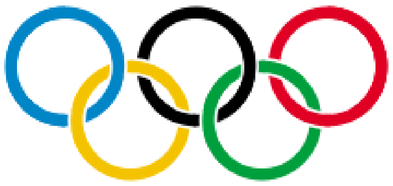
1. Type `bye()` (and press 'Enter'). This closes the turtle window, lets you start again.
2. If this doesn't work , try opening the Python (Command Line) - it should be right beside the place you started IDLE. It's ugly (picture below), but it works!



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> from turtle import *
>>> forward (100)
>>> _
```

Try it out! -
- Can you draw a square? Hint: try `left (90)` or `right (90)`.

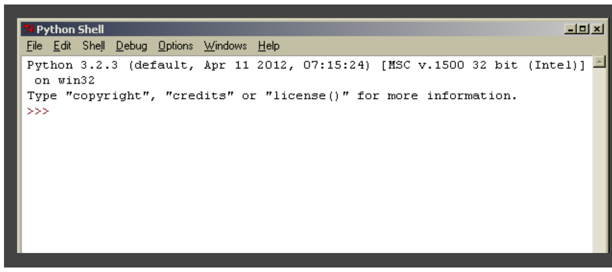
- Can you draw a triangle? Hint: use left and right, but with different numbers.
- Can you draw a square and a triangle with a clear gap between them?
  - Hint: use `clear()` or `reset()` to clear the screen, check what `penup()` and `pendown()` do.
- Draw the Olympic Rings -
  - Hint; Use `circle (20)` - try different numbers
  - Hint; use `width (10)` - try different numbers
  - Hint; use color (“red”) - try other colors, remember the “quotes” and remember to spell it the American way!



- Make a Motorway sign
  - Hint: `bgcolor(“blue”)`
  - Hint: `write(“something”)`
  - Hint: `try goto(10,10)` see if it helps, use different numbers.
  - Hint: try `setx(100)` or `sety(100)`.

5. Saving your Programs

Are your fingers tired yet? Want to avoid typing the same Turtle Commands over and over again? Here’s how you can save (and open) your Turtle Programs.



```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

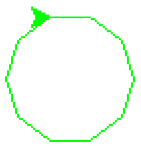
And we’re back!
If your Turtle got stuck earlier on, notice that we’re back using Python Idle (black writing on white screen).

Everything should work ok for you from here.

In the next program lines beginning # are comments - they help us understand what the program does, but Python just ignores them - say whatever you want after a # !

- In IDLE, create a new window from the Menu, select File … New Window.
- Give our new Program a name (File … Save As).Remember where you put it!
- Type the Text, similar to the picture on the right. Check your Spelling!
- Make sure to type the : and check that the next two lines start slightly further in. Hit the ‘Tab’ key on the top left of your keyboard if need be.

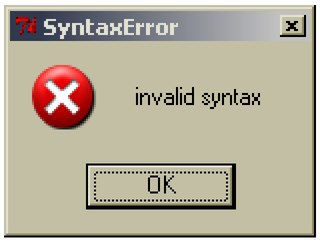
```
#Tell Python we want to use Turtle
from turtle import *

#change our pen colour to blue
color("blue")

#Loop 10 times
for i in range (10):

    #move the turtle
    forward (20)
    right (36)

done()
```



Hit F5 to run
(or Menu Run … Run Module)

If the program runs ok, you should see a picture like this one.

What if something goes wrong? Python will show you a message like this one.



invalid syntax

Invalid Syntax is Python's way of saying 'check your spelling'

Python will also turn the line of your program orange where it *thinks* the problem is. It doesn't always get it right, but it's a good place to start.

There's some more new stuff going on here

- The word `done()` at the end helps stop Turtle from freezing up.
- Loops (the 'for …' bit) .More on this later!

***Try it out!** - Can you*

- *Close the program you've just typed , then open it up again.*
- *Create a new script , use it to draw triangles; Hint: Save it as a different name.*
- *Create a new script , use it to draw squares with different widths of lines.*
- *Find turtle programs on disk or the web, try opening them, see what they do?*

6. Getting Help - STFW (Search the fantastic Web)

Ok , you're bumping into a problem or you want to know more?

- Ask the people sitting beside you. You probably already did this if you got stuck in the previous sections (such as looping)

- Ask the nice people at coder-dojo (but we won't always be there)

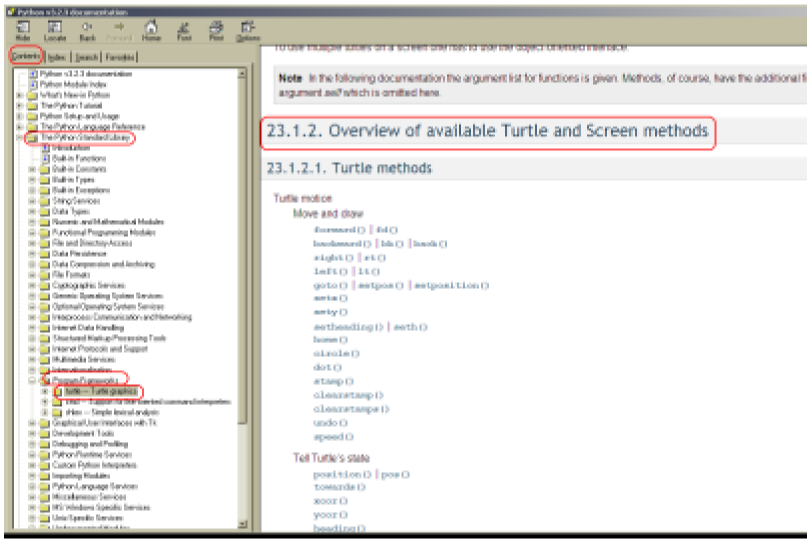
- Use the help command at the command prompt e.g.
  - `help(bgcolor)` or `help (right)` or `help (forward)`

Don't Worry be Happy!
Not sure what the stuff in the help files means? - just try typing it in and see what it does!

- Not sure of what to type in the help command? type help () empty brackets then type
  - keywords - gives the list of special keywords for python
  - modules - gives the list of extensions to python (e.g. Turtle)
  - topics - chapters in the help that might give you

- Use the suggestions that come in the IDLE editor - e.g type turtle. (remember the dot!!) and then press <ctrl> space. A list of the commands that you can use pops-up.
- Look at the Help that comes with Python.IDLE (Help … Python Docs)

- There is a *lot* of information in the Python Docs - they have *everything* that you can do with Python.
  - For Turtle, click on on the 'Content' tab in the top left
  - In the tree, click on "The Python Standard Library" then "Program Frameworks" then "Turtle - Turtle Graphics"
  - On the Turtle page, scroll down to "Turtle methods" - it will give you more information on the Turtle commands that you've been using already, and introduce a few new ones.


Python v3.2.3 documentation
Hide
Locate
Back
Forward
Home
Font
Print
Options
Contents
Index
Search
Favorites
Python v3.2.3 documentation
Python Module Index
What's New in Python
The Python Tutorial
Python Setup and Usage
The Python Language Reference
The Python Standard Library
Introduction
Built-in Functions
Built-in Constants
Built-in Types
Built-in Exceptions
String Services
Data Types
Numeric and Mathematical Modules
Functional Programming Modules
File and Directory Access
Data Persistence
Data Compression and Archiving
File Formats
Cryptographic Services
Generic Operating System Services
Optional Operating System Services
Interprocess Communication and Networking
Internet Data Handling
Structured Markup Processing Tools
Internet Protocols and Support
Multimedia Services
Internationalization
Program Frameworks
turtle — Turtle graphics
cmd — Support for line-oriented command interpreters
shlex — Simple lexical analysis
Graphical User Interfaces with Tk
Development Tools
Debugging and Profiling
Python Runtime Services
Custom Python Interpreters
Importing Modules
Python Language Services
Miscellaneous Services
MS Windows Specific Services
Unix Specific Services

Note In the following documentation the argument list for functions is given. Methods, of course, have the additional argument *self* which is omitted here.

23.1.2. Overview of available Turtle and Screen methods

23.1.2.1. Turtle methods

Turtle motion
  Move and draw
    forward() | fd()
    backward() | bk() | back()
    right() | rt()
    left() | lt()
    goto() | setpos() | setposition()
    setx()
    sety()
    setheading() | seth()
    home()
    circle()
    dot()
    stamp()
    clearstamp()
    clearstamps()
    undo()
    speed()
  Tell Turtle's state
    position() | pos()
    towards()
    xcor()
    ycor()
    heading()

What's this? - name 'turtle' is not defined
Some of the help docs talk about `turtle.clear()` , but we can get away with using `clear()` - why? Because we've already told Python we want to use turtle using

```
    from turtle import *
```

- Google for it . A phrase like 'Python Turtle Howto loop'' will bring up plenty of results. Some good places to start are;
  - http://p2pu.org/en/groups/python-programming-101/
  - Search for "Python Videos" on Youtube

- Python Tutorial on Wikibooks http://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3/Print_version
- A byte of Python : http://www.swaroopch.org/notes/Python
- Invent your own games with Python book - IYOCGwP_book1.pdf. Google it or there should be a copy of this (free) book with the guide.
- Making Games with Python and PyGame - makinggames.pdf - Google it or there should be a copy of this (free) book with the guide.
- Some more Python Tutorials http://www.dreamincode.net/forums/forum/133-python-tutorials/

***Try it out!**: Look for help to get these working*

- *Find out what* `title()` *,* `window_height()` *and* `window_width()` *do, and how to use them.*
- *Find out what* `clear()` *and* `reset()` *do, and how to use them.*
- *What do* `showturtle()` *and* `hideturtle()` *do?*
- *How to change the Turtle shape.* `shape("turtle")` *and* `getshapes()`
- *Draw shapes filled with colours - look in help for* `begin_fill()` *and* `end_fill()`

7. Do it again. And Again . And Again ... Loops

What's a loop? Anything that you want Turtle to do again, and again, and again!

- Back in a previous section we made turtle go round in a loop. We used the words
  `for i in range (10):`
- The colon : is important as it tells python to start a new block (the one that starts a few spaces in) that will be repeated again and again.
- Try typing this program - remember that Python ignores any line beginning with # - the make it clearer.

```
#Tell Python we want to use Turtle
from turtle import *

#Tell Python that we want to set
#our colours using numbers
colormode(255)

#Loop 255 times
for i in range (255):

  bgcolor(i,i,i)

#Tell Python that we're finished
done()
```

i is a just a box to store a number in - a number that changes a lot . We'll cover it more in the next section

- What does it do? Python counts from 1 to 255, each time puts that number into a box called '`i`'.
- Each time it counts / loops , Python sets the `bgcolor`() to whatever number `i` is. That makes the screen fade from black to white over the 255 steps.
- The `bgcolor`() command takes three numbers; one each for red, green and blue. The number must be between 1 and 255.

***Try it out!**:*
- *At the command line, play with `bgcolor()` - what colours can you get?*
- *Can you get Python to print out the number of `i` as it loops?*
- *In the program , can you change the `bgcolor()` line to get different colour effects. e.g. instead of `i`, put in `255-i` or `i-2 ?`*
- *Can you change the previous turtle sample (where we looped and drew a circle) to draw a square instead?*
  - *Hint; loop 4 times, `forward (100), right (90)`*
- *Can you do a loop within a loop? e.g. draw the square 20 times , turning a little bit within each one ?*
- *As you draw the 20 squares, can you draw each one in a different colour?*
  - *Hint; set `colormode(255)` to use numbers, then set the `pencolour`(red,green,blue) - where red green and blue are different numbers*

(just like `bgcolour`)

- *As you draw the 20 squares, can you make each one a little bit bigger than the last?*

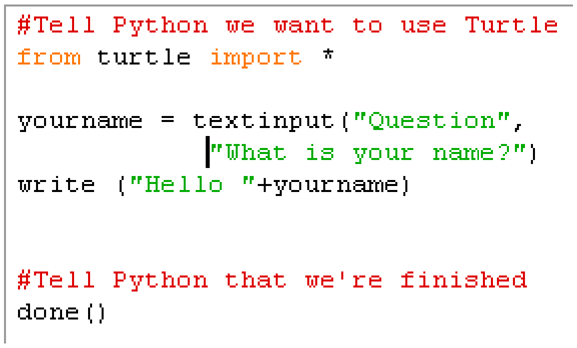
8. Give Python a memory - Variables

- In last example, our loops put a number into a box called '`i`', and then used this number again later. Another name for these boxes are variables.
- You can give variables almost any name you want - myLunchMoney, myName, carColour could all be variable names.

Variables are boxes that you can store a number, word, or pretty much anything else in.

Variables give Turtle a memory.

Lets try using variables in a different way. Try typing the text in the screenshot.



```
#Tell Python we want to use Turtle
from turtle import *

yourname = textinput("Question",
                     "What is your name?")
write ("Hello "+yourname)


#Tell Python that we're finished
done()
```

- Try running the program - what does it do?
- `TextInput` asks you a question, puts the answer into a variable / box called `yourname`.
- It then tells turtle to write the value in `yourname` to the screen. Simples!

- So far we've used variables / boxes with bits of text in them (yourname). Before that we used variables to count from 1 to 255 and changed colours using numbers.
- Be careful when using text and numbers together - type and run this program.

- We ask the user for a **number** using `numinput`.
- We convert it to an **Integer number** using `int`
- We loop the number of times the users asks using `for`
- We change the number to **text** (using `str`) and `print` it.

```
from turtle import *

# Ask the user for a number
number = numinput("question",
             "pick a number between 10 and 20",
             minval=10, maxval=20)

#Change the type of number that we have
numberAsInteger=int(number)

#Count up to this number
for i in range (numberAsInteger):

    # Change the number to text and print it
    print("counting "+str(i))

done()
```

Floats are numbers with decimals (like 12.4) Integers are full numbers (like 4 or 12)
Python will remind you if you use the wrong one!

Try it out!

- *Create new program. Get it to do the following*
    - *Tell it to use turtle*
    - *Ask the user for a colour*
    - *Set the pencolour to that colour*
    - *Ask the user for a number. Store it in a variable called shapesize*
    - *Draw a square of the size the user said.*
- *Hint: a lot these steps have been used in previous samples!*

9. IF … Then Deciding what to do

Turtle and Python would be pretty silly if they couldn't decide what to do! Here's how

```
if (sometest) :
      block of commands
```

IF allows you to make a decision. IF it is 12 O'clock I will eat Pizza ELSE I will play with Turtle

```
#Usual tine - tell pyton we want turle
from turtle import *

#Loop 100 times
for i in range (100):

    #Ask the user for a command
    userinput = textinput("command",
                "Please enter a turtle command")

    # If the user has pressed 'cancel' ...
    if (userinput == None):

        # ... end the loop
        write ("bye!")
        break

    # Print what the user has entered
    print ("user entered:"+ userinput)
```

- Here's an example
- Loop 100 Times
- Ask the User for a command
- if (the user has pressed cancel):
    - then break out of the loop
- Print out the command the user has entered

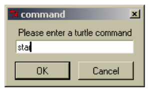
more ....

```
    # If the user has entered 'star' ...
    if (userinput == "star"):

        # ... then draw a star!
        begin_fill()
        for j in range (7):
            fd (100)
            rt (150)
            fd (100)
            lt (99)
        fd (5)
        end_fill()

#Free up the window
done()
```

- if (the user has entered ‘star’):
    - then draw a star
- End of loop / go back around to start (notice the spacing)

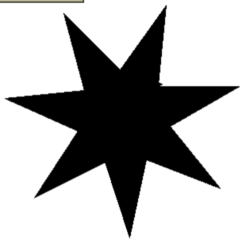
When the program runs type star (and press ok) - it draws a star and fills it!

Press cancel - it ends the program



Please enter a turtle command

sta

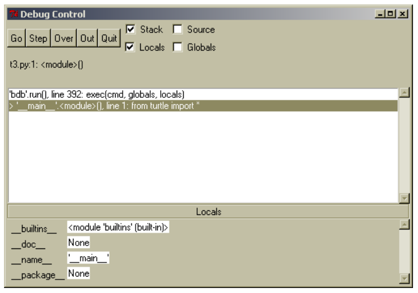


Try it out!

- *Change the star command to draw in a red line, fill it in yellow (hint read the help docs)*
- *Add a new command to draw a square*
- *Add a new command to change the colour*

10. When things to wrong - debugging.

Did you get to this stage in the guide without a single error? Me neither! What can we do when things go wrong?



[✓ Stack] Source
[✓ Locals] Globals

```
t3.py:1: <module>()

'bdb'.run(), line 392: exec(cmd, globals, locals)
> '__main__'.<module>(), line 1: from turtle import *
```

__builtins__	<module 'builtins' (built-in)>
__doc__	None
__name__	'__main__'
__package__	None

To view your program running in slow motion

In your program set a break point (the line where your program will pause)

- Right click on a line and then ‘set break point’
  The Line will become yellow

Now in the **main** window, select debug from the menu, then debugger.
In Debug menu on main screen window … debugger

Now run program as normal … program will stop at debugger.

- Step over to go through the program step by step. (Google for the other commands)
- The stack window (left) will show the values of variables as you move through the programme.

Don't forget the more simple things!

- When there is an error, read the error message , and google it if you don't understand it!
- Python highlights in orange the line where it thinks the problem is
- Check your spelling!
- You can use print() commands to see where in the program

11. Here's Some I Did earlier - Turtle Examples

There are lots of good examples for Python and Turtle. We've included a couple of samples in the TurtleDemo-Python3.x folder

To find them, click the 'File' Menu, then 'Open', then look for that folder. There are plenty of different scripts to try out.

Try it out!

- *Some of the ones that we liked are below. Which is your favourite? Mine was tdemo_paint.py*
- *Change one of the scripts to make it even better*

12. What Happens Next?

There are two free books included on the USB key / folder. The nice authors that wrote them allow you to share them as long as you keep their names attached.

- Making Games with Python and PyGame
- Invent your own Games with Python

If you don't have the USB key, Google for these files (I*YOCGwP_book1.pdf*) - they're easy enough to find!

Try it out!

- In Have a quick read of Invent your own Games with Python - esp chapter 1 to 6.
- Try out the Guess the Number, the Jokes and the Dragon Realm samples – you already have Python installed, so you're good to go!

Well done, now you have all the bits you need to program Games!.
The words may be different in other computer languages, but the ideas are the same.
What are you going to do next?