CSC108H Lab 1

Introduction

Welcome to the KDE desktop! You can find Wing IDE in the taskbar, it's the rightmost icon there (the one with the feather).

Please don't write on these handouts and return them to the lab TA after class in order to save some trees. Lab handouts will be posted at the end of each week for future reference.

Driver and Navigator

You will be working in pairs. Throughout the term, we will use the terms **driver** and **navigator**. Here are the definitions of the two roles:

Driver: The person typing at the keyboard.

Navigator: The person watching for mistakes, and thinking ahead.

Here is the most important rule for this and all future labs:

The navigator must not touch the keyboard or mouse. The driver's role is to work with the computer, and the navigator's is to think about language issues and upcoming issues related to the problem being solved. If the navigator interferes with the driver, the group loses the view of what is coming up and may make the problem harder to solve, making it harder for the driver to learn the material.

In every lab handout, we'll call you two s1 and s2, and s1 will be the first driver.

Part 1: Conversions

Open up Wing IDE. You will be writing short programs to perform conversions from one type of unit to another.

Write the .py files necessary to perform the following conversions and save them in your home directory:

File Name	Conversion	Formula
`pints_to_litres.py`	Pints to Litres	L = P * 0.5682
`litres_to_pints.py`	Litres to Pints	P = L * 1.76
`metres_to_inches.py`	Metres to Inches	I = M * 0.0254
`inches_to_metres.py`	Inches to Metres	M = I * 39.4

Each program should ask for a value from the user (using `raw_input()`) and print the converted value. Use it to find the following:

99 pints in litres

465 litres in pints

3.5 metres in inches

42 inches in metres

Part 2: Generalization

Switch driver and navigator (so s2 is driving now).
Write a program called `convert.py`. It should ask the user for two values - the initial value and a factor by which the initial value is multiplied. It should print a nice message that includes the original and the converted value. Try the values from Part 1 with this new program.

Part 3: Using the Linux shell to run programs

The CDF machines are running a version of Linux, which comes with a terminal tool called Konsole. Open a Konsole window by clicking on the fourth icon in the taskbar. Type in `ls` (LS) and press Enter. You should see the .py files in your home directory.
To open a Python shell type in: `python`
To close the Python shell type in: `exit()`
You can run your program by invoking the Python interpreter directly:
To run the .py files in Konsole, type in: `python` *`filename`*`.py`
What happens when you run the `.py` files from the shell? Is there any difference between the output in Wing and in the Terminal?

Part 4: The `turtle` module

Switch driver and navigator.

The **`turtle`** module is a collection of functions meant to teach Python novices about programming through the use of graphics. In **`turtle`**, you are working on a blank canvas composed of 400x400 pixels. You give sequential commands to move a pen (denoted by an arrow) to draw various shapes. It has several functions you can call to make a cursor draw lines and shapes on the canvas:

`color(`*`string`*`)` – changes the color of the pen to the color denoted in *`string`*.

`up()` – lifts the pen off the canvas. Moving the pen after an `up()` call will not draw anything on the canvas.

`down()` – touches the pen to the canvas. Moving the pen after a `down()` will cause a line to be drawn.

`goto(x,y)` – sends the pen cursor to point (x,y) (which must both be integers)

`setheading(`*`angle`*`)` – sets the heading of the pen (the direction in which the pen is pointing). Possible headings are 0 (right) to 359.

`forward(x)` – commands the pen to go forward for x pixels in its current heading (set by `setheading()`)

`fill()` – use this command before drawing a closed figure to indicate that you will want it filled with the current colour when it is drawn

`endfill()` – use this when you finish drawing a figure to get turtle out of filling mode

`write(`*`string`*`)` – writes the string on the canvas at the pen's current location

`circle(x)` – draws a circle of radius x starting at the pen's current position going counter clockwise from the pen's current heading

The **`turtle`** canvas uses a Cartesian grid with horizontal (x) and vertical (y) coordinates. The centre of the **`turtle`** canvas is point (0,0). The x coordinates grow when going right, the y coordinates grow when going up.

Download the file `christmas.py` and open it in Wing to see a sample **`turtle`** program. Pay close attention to the comments (lines beginning with #) that explain what each line of code does. Save the file in Wing, go back to Konsole and run the file in the Linux Terminal (like in Part 3) to see what it draws.

Now download the file `canvas.py`. In it, write code to draw:

- A rectangle
- A triangle
- A smiley face (Hint: you will need to use `help(circle)` to work out how to draw arcs)