Input and Output

```python
user_input_as_str = input('Prompt: ')  
print('Hello, you entered: ' + user_input_as_str)
```

Comments

```python
# everything to the right of the # is ignored
print('hi')  # this is ignored
```

Data types (and conversion functions)

- Strings: `str()`
- Integers: `int()`
- Floats: `float()`

Assigning values to variables and

```python
x = x * 2    # <-- this puts the number 4 in x
y = 'y' * 2  # <-- this puts the string 'yy' in y
```

If statements

```python
if x >= 0:
    print('x is zero or positive')
else:
    print('x is negative')
```

While loops

```python
x = 0
while x < 10:
    print(x)
    x = x + 1
```

Today's lecture

- The Turtle graphics package
- Brief history
- Basic commands
- Drawing shapes on screen

Logo and Turtle graphics

In 1967, Seymour Papert and Wally Feurzeig created an interpretive programming language called Logo.

Papert added commands to Logo so that he could control a robot called a turtle which drew shapes on paper, from his computer.

"Turtle graphics" in Python

Using the Turtle involves instructing a virtual turtle to move on the screen and draw lines to create shapes and patterns.

Video of Turtle Robots

Start @2:49
- Turtle robots in action
- Kids using computers to instruct the turtle (and each other :D)
  *forward 5 steps!*
  *turn right!*
  *turn left 40 degrees!*

Programming

Programs are a sequence of instructions.

Turtle graphics let us see a little bit more of how you can put together these instructions (along with conditional statements and loops) to produce a wide variety of programs.

Import

Some functions are part of Python’s core libraries -- they are ‘built-in’.

For example:

- \texttt{print()}
- \texttt{input()}
- \texttt{float()} 

Other functions need to be imported into your Python program.

The turtle module must be imported at the start of any Python program that uses it.

To do so, just write this at the beginning of the program:

\texttt{import \textit{turtle}}
Turtle commands

turtle.forward(x)
Moves turtle forward in direction it is facing by x steps

import turtle
turtle.forward(200)

Turtle Functions

There are four basic turtle commands

turtle.forward(x)
Moves turtle forward in direction it is facing by x steps

turtle.back(x)
Moves turtle backward from its facing direction by x steps

turtle.left(x)
Turns the turtle x degrees counterclockwise

turtle.right(x)
Turns the turtle x degrees clockwise

Plus some others that you might find useful.

turtle.undo()
Undo whatever your last command did.

turtle.penup()
Make it so that when the turtle moves, it DOES NOT draw anything (until turtle.pendown() is called)

turtle.pendown()
Make it so that when the turtle moves, it DOES draw a line as it goes (until turtle.penup() is called)
(by default, the pen is down.)

Turtle example

Using the Python interpreter in IDLE, let’s use turtle graphics to draw a square.
First, import the turtle package...

DEMO

Turtle example

We are going to draw a right-angled triangle
Turtle example

- Note how the turtle is now facing upward after being turned 90 degrees left

```python
>>> import turtle

>>> turtle.forward(200)
>>> turtle.left(90)
```

Turtle example

- Working out the length of the longest side using the Pythagoras’ formula

```python
>>> import turtle

>>> turtle.forward(200)
>>> turtle.left(90)
>>> turtle.forward(200)
>>> turtle.left(135)
>>> c = ((200**2)+(200**2))**0.5  # around 283 steps
```
Turtle example

- The finished image

Exercise

Write a Python program that draws a rectangle.
The long sides must be 300 steps long and the short sides must be 150 steps long.

```python
import turtle

turtle.forward(300)
turtle.left(90)
turtle.forward(150)
turtle.left(90)
turtle.forward(300)
turtle.left(90)
turtle.forward(150)
```

Turtle example

We can use loops when drawing shapes using Turtle graphics.

A program that will draw a square using a loop:
We can use loops when drawing shapes using Turtle graphics.

A program that will draw a square using a loop:

```python
import turtle
count = 0
while count < 4:
    turtle.forward(200)
turtle.left(90)
count = count + 1
```

What does this program do?

```python
angle = 0
while angle < 360:
    turtle.forward(1)
turtle.right(1)
angle = angle + 1
```

What does this program do?

```python
angle = 0
while angle < 360:
    turtle.forward(2)
turtle.right(1)
angle = angle + 1
```
Turtle example 2

What does this program do?

```python
angle = 0
while angle < 360 :
    turtle.forward(2)
    turtle.right(1)
    angle = angle + 1
```

Exercise

Draw the shape that is produced by the following Python program:

```python
import turtle
count = 0
while(count < 180):
    turtle.forward(2)
    turtle.right(1)
    count = count + 1
turtle.right(45)
turtle.forward(300)
turtle.left(90)
turtle.back(150)
turtle.right(45)
turtle.back(250)
```

Exercise

Draw the shape that is produced by the following Python program:

```python
import turtle
count = 0
while(count < 180):
    turtle.forward(2)
    turtle.right(1)
    count = count + 1
turtle.right(45)
turtle.forward(300)
turtle.left(90)
turtle.back(150)
turtle.right(45)
turtle.back(250)
```

Exercise

```python
import turtle
big_line = 100
little_line = 50
angle = 90
turtle.left(angle)
turtle.forward(big_line)
count = 0
while count < 4:
    turtle.right(angle//2)
    if count != 3:
        turtle.forward(little_line)
    else:
        turtle.forward(big_line)
count = count + 1
turtle.right(90)
turtle.forward(130)
```

What does this program draw?
Operators

<table>
<thead>
<tr>
<th>Operation</th>
<th>Symbol</th>
<th>Applied to integers</th>
<th>Applied to floating point numbers</th>
<th>Applied to strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponent</td>
<td>**</td>
<td>2 ** 3 = 8</td>
<td>2.0 ** 3.0 = 8.0</td>
<td>N/A (ERROR)</td>
</tr>
<tr>
<td>Multiply</td>
<td>*</td>
<td>2 * 2 = 4</td>
<td>2.0 * 2.0 = 4.0</td>
<td>“aab”</td>
</tr>
<tr>
<td>Divide</td>
<td>/</td>
<td>10/3 = 3.333</td>
<td>10.0/3.0 = 3.333</td>
<td>N/A (ERROR)</td>
</tr>
<tr>
<td>Divide (integer)</td>
<td>//</td>
<td>10 // 3 = 3</td>
<td>10.0//3.0 = 3.0</td>
<td>N/A (ERROR)</td>
</tr>
<tr>
<td>Remainder</td>
<td>%</td>
<td>10 % 3 = 1</td>
<td>10.0 % 3.0 = 1.0</td>
<td>N/A (ERROR)</td>
</tr>
<tr>
<td>Add</td>
<td>+</td>
<td>8 + 9 = 17</td>
<td>8.0 + 9.0 = 17.0</td>
<td>“ab”</td>
</tr>
<tr>
<td>Subtract</td>
<td>-</td>
<td>9 - 7 = 2</td>
<td>9.0 - 7.0 = 2.0</td>
<td>N/A (ERROR)</td>
</tr>
</tbody>
</table>

Exercise

```python
import turtle
big_line = 100
little_line = 50
angle = 90

turtle.left(angle)
turtle.forward(big_line)
count = 0
while count < 4:
    turtle.right(angle//2)
    if count != 3:
        turtle.forward(little_line)
    else:
        turtle.forward(big_line)
count = count + 1
turtle.right(90)
turtle.forward(130)
```

Summary

(Python) programs are sequences of instructions ("statements")...
...where you can assign values to variables...
...and use "control logic," such as conditional statements and loops, to influence the order in which instructions are executed.

With just this (plus perhaps some input and output) it is possible to make a huge variety of programs -- every piece of software that you've ever used!!