Plan

Three Lectures on Python

1. Input, output and variables.
2. Loops and Conditional Statements
3. Turtle graphics
Paper and Pencil

These lectures (especially tomorrow) will include a few longer exercises that you will only really be able to engage in if you have pencil and paper.
Today’s lecture

Introduction
What is a programming language?
What is Python?
Using IDLE to experiment with Python.

Output
i.e. displaying text on the screen using `print()`

Comments
annotations to a program that help make it easier to understand

Variables
Values that can change in a program -- a central concept in programming.

Input
Getting input from keyboard using `input()`
What is a programming language?

A formal language that specifies how to perform a computational task

Many programming languages exist:

- Basic
- C and C++
- C#
- Java
- Python

Python is a relatively modern language, created in 1989 by Guido Van Rossum in The Netherlands
Statements

Most programs consists of a series of commands called **statements**.

They are generally executed (ie. run) in the order they appear (top to bottom)

Kind of like a recipe or set of instructions.

The statements must be written correctly otherwise you will get a **syntax error**.

Python programs are saved in files with the `.py` extension

---

How to boil water
Pick up kettle.
Open kettle lid.
Put kettle under tap.
Turn on cold water tap.
Wait for 5 seconds.
Put kettle on stand.
Turn on kettle.
What happens behind the scenes?

The “human-readable” statements in our programs are translated into “machine code” instructions that are run on the CPU.

Two ways this is done, either…

...a **compiler** translates the entire program file at once, or

an **interpreter** translates one line of the program and runs it, and then it translates the next line and runs that, etc.

Python is usually interpreted, but there are also compilers available for Python.
Integrated Development Environments

An IDE is an application that is used by programmers

- Write and edit program code
- Check for errors
- Run the program

We use a Python IDE called IDLE.

It provides

- a python interactive shell, and
- an editor for writing and running scripts
IDLE Interactive Shell

In the interactive shell, you can type a single statement, and when you hit [enter], that statement will be executed and you can see the result immediately.

This is especially helpful for experimentation and learning.

"Does this work or produce an error?"

DEMO

1. (different kinds of errors)
2. the cursor
IDLE Editor

Create a new program by clicking
File → New File

Type your statements in the file, then click on
Run → Run Module…

My advice
I recommend using the editor most of the time.
It saves having to rewrite one more more lines
when you have a typo.

But don't write an entire program and only try
running it when you are done! Write a line, run
it. Write another line, run it! This makes it
easier to see where you have gone wrong.

F5 is your friend!
print()

Traditional first program is displaying “Hello World” on screen

To display text on screen you use the `print()` function

DEMO
Comments

When writing a program, it is helpful to leave comments in the code.

You can write a comment in Python by typing a '#' in front of the line.

The compiler will ignore all text after the '#'.

It can be useful to "comment-out" code.

DEMO
Variables

A variable can be thought of as a ‘container’ in the computer’s memory in which you can store data.

Variables have a name and a value.

You use the name to refer to the variable. The name does not ever change.

The value is “what is in the container.” A variable’s value can and often does change when the program runs.

To assign a value to a variable i.e. to "put something in the container (removing whatever was in it before)" you use a single equals sign. e.g:  \( x = 10 \)
Assigning a value to a variable:

```r
## assigning values to four variables
age = 21
name = "Matthew"
height = 1.55
course_name = "Comp sci111/111G"
```
Exercise

What does this program produce as output?

DEMO

```python
x = 10
print(x)
x = 20
print(x)
x = x + 5
print(x)
```
Program

```
x = 10
print(x)
x = 20
print(x)
x = x + 5
print(x)
```

Output

```
10
20
25
```

Answer
Variables

The names of variables

- **should** reflect what is stored in the variable
- **can** include numbers, upper- or lower-case letters, and underscore, but
- **MUST** begin with a letter or underscore (eg. ‘_’)

In Python, variables are usually all lowercase and separated using an underscore.

**Good variable names**
- age
- num_students
- search_criteria

**Illegal variable names**
- 1st_name
- num-students

**Legal but poorly(?) chosen var. names**
- thing
- numberofpeoplethataregoingtotheparty
- x
- numStudents
- NumStudents
<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Strings</strong></td>
<td>Sequence of characters</td>
<td>e.g. &quot;Hello&quot;, &quot;Goodbye&quot;</td>
</tr>
<tr>
<td></td>
<td>Plain text (ASCII or Unicode)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enclosed in quotation marks</td>
<td></td>
</tr>
<tr>
<td><strong>Integers</strong></td>
<td>Whole numbers (i.e. without a decimal point)</td>
<td>e.g. -100, 0, 45</td>
</tr>
<tr>
<td><strong>Floating Point</strong></td>
<td>Numbers with a decimal point</td>
<td>e.g. 5.2, -1.002, 0.0</td>
</tr>
</tbody>
</table>
Exercise

What is the difference between these programs?

How does their output differ?

Why?
Why?

When you wrap something in quotation marks (") you are describing a string.

Otherwise it is python code. In this case, a variable name.

Don't wrap variable names in quotation marks!
Assigning a value to a variable:

```python
## assigning values to four variables
age = 21
name = "Matthew"
height = 1.55
course_name = "Compsci111/111G"
```

Exercise:

What type are all of these variables?
To remind you, the options are:

- floating point
- integer
- string
Changing variable values

```python
### incrementing age
age = 21
age = age + 1
print(age)
```

```output
22
```

```python
### changing a string
course_name = "Compsci111"
course_name = course_name + "/111G"
print(course_name)
```

```output
Compsci111/111G
```

Rhetorical Q: What is going on here?
Changing variable values

## incrementing age
```python
age = 21
age = age + 1
print(age)
```

## changing a string
```python
course_name = "Compsci111"
course_name = course_name + "/111G"
print(course_name)
```

Rhetorical Q: What is going on here?

**DEMO**

**Answer**
The `+` sign is "overloaded" -- it does different things depending on the type of its arguments (the things that it is joining together).

- **numbers**, it adds together.
- **strings**, it concatenates
## 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>“a” * 3 = “aaa”</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>“a” + “b” = “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

What is the output of this program?
Exercise

What is the output of this program?
Getting input

Primary source of input for our programs will be the keyboard.

The `input()` function:

- prints a prompt for the user to read
- captures the user’s keystrokes
- when the user presses ‘Enter’, stores the string in a variable
Getting input (changing type?)

Sometimes you need a numerical value rather than a string. For instance a program where you enter a number of seconds, and it responds by telling you the equivalent number of minutes.

```python
# this program produces an error, because `seconds` is a string
seconds = input("How many seconds? ")
print(seconds/60.0)

# this program works as desired
seconds = input("How many seconds? ")
seconds = float(seconds)
print(seconds/60.0)
```
Changing the **data type** of a variable

## x is a string

```python
x = "3"
print(x+x)
```

## int(x) **tries** to convert x into an integer

```python
x = "3"
x = int(x)
print(x+x)
```

## this produces an error. "a" cannot be converted into an integer

```python
x = int("a")
```
Exercise

Write a Python program that converts feet to metres. The conversion formula is:

\[ 1 \text{ foot} = 0.3048 \text{ meters} \]

Your program’s output should look like this (user input is in bold):

Enter feet: 34
34 feet is equal to 10.3632 metres

You will need to use:

- variables
- operators
- `input()` and `print()`
# get input from user
feet = int(input("Enter feet: "))
feet_to_metres = 0.3048
metres = feet * feet_to_metres
print(str(feet)+" feet is equal to "+str(metres)+" metres.")
Summary

Python programs consist of statements that are translated by an interpreter or compiler into instructions that the CPU can execute.

We’ve discussed the Python programming language and a few of its features:

- `print()`
- Data types: string, int, float
- Operators: +, *, **, /, //, %
- Variables and variable naming convention
- `input()` for getting a string from the user of your program
- `int()`, `float()`, `str()` for transforming between data-types