Simple Python Wrestling

Overview

- How to talk Python
- Storing Data
- Making Decisions
- Looping the loops
- Collecting data in, er, collections
- Methods and madness

Starting with Python   28-Jan-14   2

What is Python?

- Python has been around for a while
- It was invented by Guido van Rossum as a general purpose scripting language which supports a wide range of programming styles and techniques
- It is named after the TV show
  – Of course

Starting with Python   28-Jan-14   3

UNIVERSITY OF **Hull**

What is Python?

- Python is a *scripting* language that you can use to write programs
  - A *scripting* language is used to give commands to an *interpreter* that then acts on each command
- This not quite the same as some programming languages which are converted into low level instructions

Starting with Python   28-Jan-14   4

UNIVERSITY OF **Hull**

Learning Python

- Python is great to learn
  - But there are some pitfalls
- Like everything else to do with computers, you would think that everything was there for the best possible reason and works in the best possible way
  - This is not necessarily always the case though
  - We just have to deal with this

Starting with Python   28-Jan-14   5

UNIVERSITY OF **Hull**

Python Versions

- Python has now reached Version 3.n
  - But there is a large legacy of Python code out there that is based on Version 2.n
- This is not an issue for the purpose of this course, because everything we will mention runs on both versions
- But you may need to remember this when looking at existing code
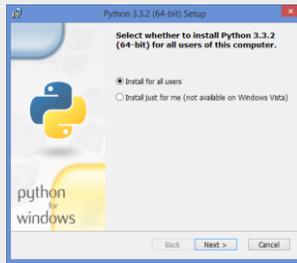 - We are going to use Version 3.3.2

Starting with Python   28-Jan-14   6

UNIVERSITY OF **Hull**

Getting Python

python™

» Download

ABOUT
NEWS
DOCUMENTATION

Download Python
The current production versions are Python 2.7.5 and Python 3.3.2.

- You can download Python from
  – http://www.python.org/download/
- You just need to run the MSI installer package

Starting with Python   28-Jan-14   7

UNIVERSITY OF **Hull**

Python Installer

- Install for all users and select the default destination directory

Python 3.3.2 (64-bit) Setup

Select whether to install Python 3.3.2 (64-bit) for all users of this computer.

● Install for all users
○ Install just for me (not available on Windows Vista)

python
for
windows

Back   Next >   Cancel

Starting with Python   28-Jan-14   8

UNIVERSITY OF **Hull**

Running Python

- When you install python you get a number of things
  – Python command line
  – IDLE Python development environment
- We are going to do everything in IDLE
- This is where we will write, run and save our programs

Starting with Python   28-Jan-14   9

Starting Idle

- Once you have installed Idle you can just start it in the usual way
- The program will open and display the "Python Shell"

Apps

IDLE (Python GUI)

Starting with Python   28-Jan-14   10

The Python Shell in Idle

- The "Python shell" is where we can type Python statements and have them obeyed instantly

Starting with Python   28-Jan-14   11

PRACTICAL BREAK 1

Data Processing with Python

Starting with Python   28-Jan-14   12

What we learned

- The Python shell lets us enter statements and have them obeyed instantly
- Python stores information (numbers and text) in named locations
- We can write expressions that change the information Python stores
- We can use input and print to communicate with the program users

---

A PYTHON PROGRAM

Making Decisions   28-Jan-14

Sums

```
# This program works out the result by adding two numbers
# Rob Miles November 2012

# Read the numbers

firstString = input('Enter first number: ')
firstNumber = int(firstString)
secondString = input('Enter second number: ')
secondNumber = int(secondString)

# work out the sum

result = firstNumber + secondNumber

# Display the result

print('The result is ', result)
```

- This program reads in two numbers, adds them together and then prints out the result

Sums

```
# This program works out the result by adding two numbers
# Rob Miles November 2012

# Read the numbers

firstString = input('Enter first number: ')
firstNumber = int(firstString)
secondString = input('Enter second number: ')
secondNumber = int(secondString)

# work out the sum

result = firstNumber + secondNumber

# Display the result

print('The result is ', result)
```

- Ask for the first number and read it in as a string

Sums

```
# This program works out the result by adding two numbers
# Rob Miles November 2012

# Read the numbers

firstString = input('Enter first number: ')
firstNumber = int(firstString)
secondString = input('Enter second number: ')
secondNumber = int(secondString)

# work out the sum

result = firstNumber + secondNumber

# Display the result

print('The result is ', result)
```

- Convert the string into an integer which we can do sums with

Sums

```
# This program works out the result by adding two numbers
# Rob Miles November 2012

# Read the numbers

firstString = input('Enter first number: ')
firstNumber = int(firstString)
secondString = input('Enter second number: ')
secondNumber = int(secondString)

# work out the sum

result = firstNumber + secondNumber

# Display the result

print('The result is ', result)
```

- Ask for the second number and convert that into a second integer

Sums

```
Adder.py - C:/Users/Rob/SkyDrive/Wrestling with Python/Week 02 Making Decisions...
File  Edit  Format  Run  Options  Windows  Help
# This program works out the result by adding two numbers
# Rob Miles November 2012

# Read the numbers

firstString = input('Enter first number: ')
firstNumber = int(firstString)
secondString = input('Enter second number: ')
secondNumber = int(secondString)

# work out the sum

result = firstNumber + secondNumber

# Display the result

print('The result is ', result)
                                                          Ln: 18 Col: 0
```

- Do the sum

Sums

```
Adder.py - C:/Users/Rob/SkyDrive/Wrestling with Python/Week 02 Making Decisions...
File  Edit  Format  Run  Options  Windows  Help
# This program works out the result by adding two numbers
# Rob Miles November 2012

# Read the numbers

firstString = input('Enter first number: ')
firstNumber = int(firstString)
secondString = input('Enter second number: ')
secondNumber = int(secondString)

# work out the sum

result = firstNumber + secondNumber

# Display the result

print('The result is ', result)
                                                          Ln: 18 Col: 0
```

- Display the result

A WORD ABOUT COMMENTS

UNIVERSITY OF **Hull**

Writing Software

- It is important when you write software that you ensure that you do it well
- A "good" program is not just one that works – although this does of course help
- For a program to be properly useful it is also important to ensure that it is well written

Making Decisions 28-Jan-14 22

UNIVERSITY OF **Hull**

Well Written Code

- Easy to read
  – All the names in the text should add meaning
- Clean and consistent layout
  – The same format for common constructions
- Well managed
  – It should be clear who wrote the code and the reasons for any changes

Making Decisions 28-Jan-14 23

UNIVERSITY OF **Hull**

Comments

- One way to add a lot of value to a program is to add comments
  – We already do this with sensible variable names, but comments allow even more detail
- A comment is something that the compiler completely ignores
  – It is only for use by the programmer

Making Decisions 28-Jan-14 24

Creating a Comment

```
7                          *Untitled*                    – □ ×
File  Edit  Format  Run  Options  Windows  Help
# This program works out the result by adding two numbers
# Rob Miles November 2014|
                                                    Ln: 2 Col: 25
```
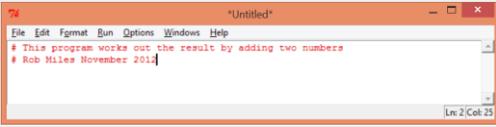
- The character # means the start of a comment

Line Comments

```
x=0 # start at the left hand edge
```

- The character sequence # starts a comment that extends to the end of the line
- You can use these to quickly explain what a statement is doing

Stupid Comments

```
count = count + 1 # add 1 to count
```

- Comments should add value
- They should not just replicate information that a programmer should know already

Comments are cool

- Make sure that you use comments
- At least put your name and the date at the top
- That way you can convince yourself that you actually wrote the code when you look at it six months later….

Making Decisions   28-Jan-14   28

IDLE PROGRAMS

Starting with Python   28-Jan-14 29

UNIVERSITY OF Hull

Programs and shells

- At the moment we have just typed Python statements into the shell and they have been obeyed
- This is fine, but what we really want is a Python program that we can just run when we need it
- This is how PC programs work

Starting with Python   28-Jan-14   30

Editing a Program

- What we want to do is write some Python statements and then have them executed for us
- We can't use the shell for this, we need an edit window where we write our code

Starting with Python   28-Jan-14   31

Opening an Edit Window

- The File menu in Idle lets us create a new Window where we can write our program

Starting with Python   28-Jan-14   32
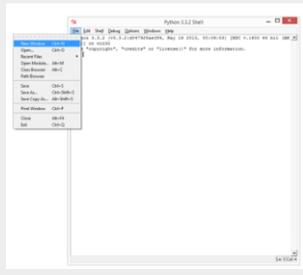
The Edit Window

- Think of the edit window as a "word processor" for program code

Starting with Python   28-Jan-14   33

Running a Program

- When we want to run a program we select the Run command from the edit window

Starting with Python   28-Jan-14   34

UNIVERSITY OF Hull

Save before Run

- Before you run a program it needs to be saved in a file
- The Python system will then read this file and run it
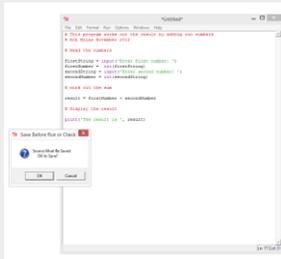
Starting with Python   28-Jan-14   35

UNIVERSITY OF Hull

Saving a Python Program

- You can store the program anywhere you like
- It will have the language extension ".py" so that the system knows it is a Python program

Starting with Python   28-Jan-14   36

Running the Program

UNIVERSITY OF **Hull**

- The program runs inside the Python shell
- If you double click the source file it will run in the shell automatically

PRACTICAL BREAK 2

Making an Adder with Python

MAKING DECISIONS

UNIVERSITY OF **Hull**
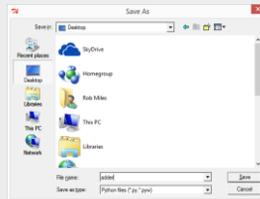
Program Flow

- At the moment every program we have written has just run through its statements in sequence
- This form of linear program flow is not always what you want
- The power of computer programs is that they can make decisions

Chapter 6.1 : Conditions   28-Jan-14   40

UNIVERSITY OF **Hull**

The Three Types of Flow

1. Straight line:
   Perform one statement after another
2. Decision:
   Choose a statement based on a given condition
3. Loop
   Repeat statements based on a given condition

Chapter 6.1 : Conditions   28-Jan-14   41

UNIVERSITY OF **Hull**

Double Glazing Program

- We are going to consider a program we are writing for a customer
  – Read in height and width of window
  – Print out area and length of glass to buy
- This might even be useful
- Before we can write the program we need to go find some *metadata*

Chapter 6.1 : Conditions   28-Jan-14   42

UNIVERSITY OF **Hull**

What is Metadata?

- Metadata is data about data
  - Limits (maximum and minimum values)
  - Units (measured in metres, gallons, years)
- It gives a proper context for what the program is doing
- You have to gather the Metadata **before** you write the program

Chapter 6.1 : Conditions 28-Jan-14 43

UNIVERSITY OF **Hull**

Where does Metadata come from?

- It must come from the customer
  - They are the only people who can tell you about their business
- Only the double glazing salesman knows that he measure his windows in meters
- If you assume that he uses feet and inches you will supply a useless program

Chapter 6.1 : Conditions 28-Jan-14 44

UNIVERSITY OF **Hull**

Getting Metadata

- You need to go out and ask the customer for this information
- They will not necessarily think to tell you
- Two assumptions that lead to disaster
  - Customer assumes you know the units
  - You assume the customer measures his windows in feet
- Result = **FAIL**

Chapter 6.1 : Conditions 28-Jan-14 45

UNIVERSITY OF **Hull**

Double Glazing Metadata

```
# Window sizes measured in meters
#  Invalid values:
#   width less than 0.5 metres
#   width greater than 5.0 metres
#   height less than 0.75 metres
#   height greater than 3.0 metres
```

- This is the metadata that drives our value inputs for the double glazing program
- I have written it as a comment
  – This is not accidental

Chapter 6.1 : Conditions   28-Jan-14   46

UNIVERSITY OF **Hull**

Conditional Execution - if

- The if statement lets a program react in a particular way to data it receives
- This allows us to use metadata in our programs to make them more effective
  – The double glazing program could reject widths and heights that are incorrect
  – This will protect us from lawsuits..

Chapter 6.1 : Conditions   28-Jan-14   47

UNIVERSITY OF **Hull**

Conditional Statement

```
if (condition):
    statement we do if condition is true
else:
    statement we do if condition is false
```

- This is the general form of the Python conditional statement
- The condition is an expression that returns a boolean result – True or False

Chapter 6.1 : Conditions   28-Jan-14   48

Relational Operators

```
2 * ( width + height ) * 3.25
```

- We have seen how a operators can be used in arithmetic expressions to produce numeric results

```
height > 3.0
```

- We can use relational operators in expressions to produce boolean results which are true or false

Chapter 6.1 : Conditions   28-Jan-14   49

Testing the height upper limit

```
if (height > 3.0):
    print('Too high')
else:
    print('OK')
```

- This test validates the upper bound of the height value
- Note that it doesn't check for heights which are to small or negative

Chapter 6.1 : Conditions   28-Jan-14   50

Missing off the else part

```
if (height > 3.0):
    print('Too high')
```

- If you don't need the else part you can leave it out
- Whether you have an else part depends on what you are trying to achieve with the code
  – Don't feel obliged to add one

Chapter 6.1 : Conditions   28-Jan-14   51

UNIVERSITY OF **Hull**

Relational Operators

- You use relational operators to perform comparisons
- A relational operator works between two numeric operands
- It returns a boolean result which is either True or False

Chapter 6.1 : Conditions   28-Jan-14   52

UNIVERSITY OF **Hull**

== operator

```
if ( age == 21 ):
    print ('Happy 21st')
```

- The == operator returns true if the two operands are equal
- Note that this is not the same as the = operator, which performs assignment

Chapter 6.1 : Conditions   28-Jan-14   53

UNIVERSITY OF **Hull**

== operator and Floating Point

```
if ( average == 1.0f ):
    print ('Average of 1')
```

- Because floating point values can't be held exactly it is very dangerous to compare them for equality
- The condition may be unreliable because of errors in calculation

Chapter 6.1 : Conditions   28-Jan-14   54

== operator and strings

```
if ( name == 'Rob' ):
    print ('Hello Rob')
```

- We can compare strings for equality
- The comparison is case sensitive
  – The string "rob" would not be recognised by the above code

Chapter 6.1 : Conditions 28-Jan-14 55

The != operator

```
if ( name != 'Rob' ):
    print ('You are not Rob')
```

- The != (not equals) operator returns true if the operands are not equal to each other
- This can be used in the same way as the == operator

Chapter 6.1 : Conditions 28-Jan-14 56

The < and > operators

```
if (width < 0.5 ):
    print ('width too low')
```

- The < and > operators test for less-than and greater-than respectively
- Note that if the operands are equal the result is not true

Chapter 6.1 : Conditions 28-Jan-14 57

UNIVERSITY OF **Hull**

The <= and >= operators

```
if (width >= 0.5 ):
    print ('not too low')
```

- These work like < and >, but also include the case where the two are equal
- To invert a < you have to use a >=
- The code above inverts the previous test

Chapter 6.1 : Conditions 28-Jan-14 58

UNIVERSITY OF **Hull**

The ! operator

```
if ( not False )
    print ('not false is true')
```

- The not operator can be used to invert a boolean value
- It works on one operand or expression in brackets

Chapter 6.1 : Conditions 28-Jan-14 59

UNIVERSITY OF **Hull**

Combining Logical Operators

- Sometimes a program needs to combine a number of logical expressions
  – If the height is too wide or the height is too high
- Python provides operators that can be used in this way:
  and for logical **and**
  or for logical **or**

Chapter 6.1 : Conditions 28-Jan-14 60

Testing both height limits

```
if (height > 3 or height < 0.5):
    print('height invalid')
else:
    print('height OK')
```

- The Logical Operator or can be used to combine two conditions
- If one **or** other of the conditions is true the operator will return true

Inverting the Condition

```
if (height <= 3 and height >= 0.5):
    print('height valid')
```

- This test inverts the condition to return true if the height is valid
- Note we have to invert the conditions **and** change the logical operator

Creating Blocks

```
if ( height > 5 ):
    print('height restricted')
    height = 5
```

- All the statements that are indented 'underneath' the if statement are controlled by that statement
- This is how Python does "blocks"

Creating Blocks

```
if ( height > 5 ):
    print('height restricted')
    height = 5
print('This message is always printed')
```

- The print message is always obeyed because it is not indented like the other two

Indenting Pain

- Python is one of the few languages that uses this indenting technique to show how code is controlled by conditions
- Other languages use brackets to mark the start and the end
- **But in Python you must get your indenting right or code will fail**

PRACTICAL BREAK 3

The Cinema Entry Program

UNIVERSITY OF **Hull**

What we learned

- Programs use the if construction to make decisions based on the values they are working with
- The if construction can make use of combinations of conditions
- The if construction can control a number of statements by using indenting

---

REPEATING WITH LOOPS

UNIVERSITY OF **Hull**

Loops

- We create a loop so that we can repeat one or more statements
- A condition is used to determine whether or not the loop stops
- The condition is either true or false, just like that used in an if construction

Repeating with Loops   28-Jan-14   69

THE WHILE LOOP

Repeating with Loops 28-Jan-13

The While loop

- Sometimes you want a loop that will repeat while a condition is true
  - Read numbers from the user while they keep typing in ones that are not valid
- The Python language has a while construction that will do this for us
  - This repeats statements while a condition is true

A Stupid while Loop

- We can write never ending loops if we like:

```
while(True):
    print('hello')
```

- This loop will never finish (use CTRL+C to kill a program if you ever write this)..

Repeating with Loops 28-Jan-14 72

A counting while loop

• while continues while the condition is true

```
i=0
while(i<5):
    print('hello')
    i=i+1
```

• The end condition is tested each time round the loop and at the **start** of the loop
• We can use indenting to get more than one statement repeated

Repeating with Loops   28-Jan-14   73

Reading in Numbers using While

```
width = -1
while(width < 0.5 or width > 10.0):
    widthString = input('Enter the width :')
    width = int(widthString)
print('Valid width entered: ',width)
```

• This will repeatedly read the width value until a valid one is entered
  – The condition is one we have seen before

Repeating with Loops   28-Jan-14   74

For loops

• We have already seen how we can create code which will repeat something a particular number of times
• However, since this is something that we need to do a lot, Python provides a special constructions for this, the for loop

Repeating with Loops   28-Jan-14   75

PRACTICAL BREAK 4

Rejecting Invalid Values using a while loop

Starting with Python   28-Jan-14   76

THE FOR LOOP

Repeating with Loops   28-Jan-14   77

UNIVERSITY OF **Hull**

The For loop

- The for loop has the following form:

```
for i in range (1,10):
    print(i)
```

- The variable i is given each value in the range
- The loop stops when i reaches the last value in the range (note that the value 10 is not in the range)

Repeating with Loops   28-Jan-14   78

A working For Loop

```
for i in range (0,10):
    print('Hello', i)
```

```
Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
Hello 6
Hello 7
Hello 8
Hello 9
```

- This will print out Hello 10 times
- When the value in i reaches 10 the loop stops

Repeating with Loops   28-Jan-14   79

UNIVERSITY OF **Hull**

Iterating through other items

```
for i in 'Chicken':
    print('Hello', i)
```

```
Hello C
Hello h
Hello i
Hello c
Hello k
Hello e
Hello n
```

- This will go round the loop once for each character in the string
- A string is a collection of items

Repeating with Loops   28-Jan-14   80

UNIVERSITY OF **Hull**

For Loops in Python

- A For loop in Python is really something that works through a range of values
- That range can be created as a sequence of numbers (that is what we did first)
- Alternatively it can be a collection of items
  - A string can be regarded as a collection of characters

Repeating with Loops   28-Jan-14   81

Breaking out of loop

```
for i in 'Chicken':
    print('Hello', i)
    if ( i=='k'):
        break
print('done')
```

```
Hello C
Hello h
Hello i
Hello c
Hello k
done
```

- You can use the break keyword to break out of a loop early
- This works in for loops or while loops

Iterating through other items

```
for i in 'Chicken':
    if ( i=='k'):
        continue
    print('Hello', i)
```

```
Hello C
Hello h
Hello i
Hello c
Hello e
Hello n
```

- You can use the continue keyword to end an iteration early
  - In this case we don't print the 'k'
- This works in for loops or while loops

PRACTICAL BREAK 5

Printing rectangles of squares

UNIVERSITY OF **Hull**

Summary

- Loops let us repeat statements while a condition is true
- We can also repeat statements for each element in a collection
- We can break out of a loop if we want
- We can put one loop inside another

Methods   28-Jan-14   85