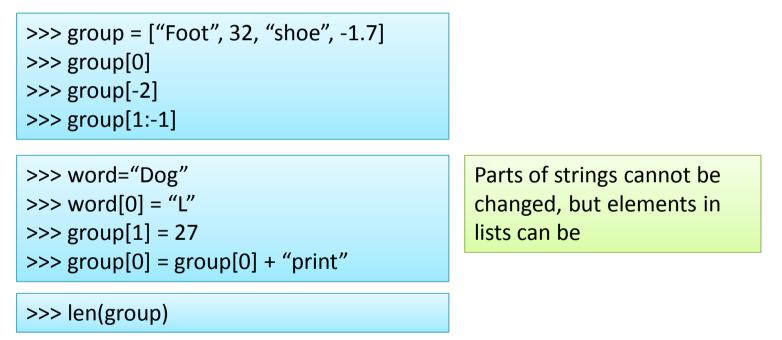
Phys 60441 Techniques of Radio Astronomy Part 1: Python Programming LECTURE 2

Tim O'Brien Room 3.214 Alan Turing Building tim.obrien@manchester.ac.uk

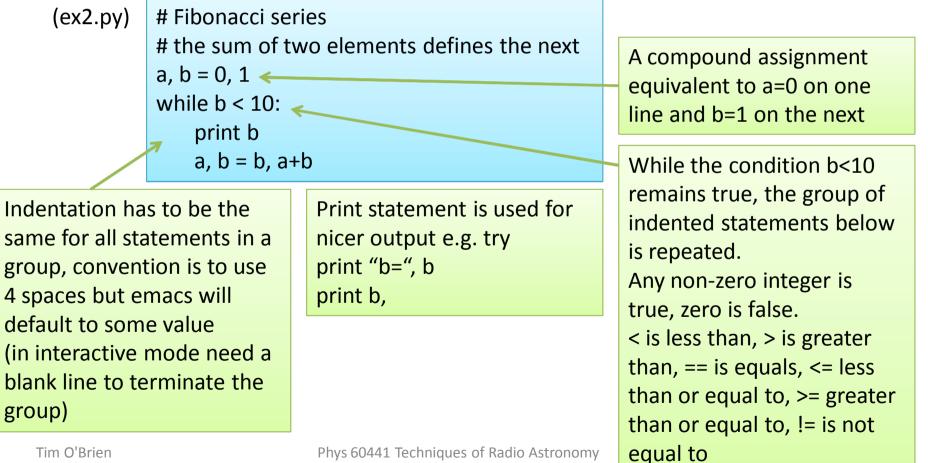
Lists

• A compound data type (elements can be same type or different types)



A while loop

• Edit a script ex2.py to contain a program to calculate the Fibonacci series



while exercise

 Write a Python script which uses a while loop to calculate and display on the screen the factorials of integers from 1 to 5

if statement and input

(ex3.py) x = int(raw_input("Please enter an integer: "))
if x < 0:
 x = 0
 print 'Negative changed to zero'
elif x == 0:
 print 'Zero'
elif x == 1:
 print 'Single'
else:
 print 'More'</pre>

Input a number from the screen

elif is shorthand for "else
if" and simply saves typing

for loop

• The for loop repeats a group of statements whilst iterating over the items of any sequence (a list or string) in the order they appear

(ex4.py)	a = ['Shakespeare', 'Dickens', 'Hartley'] for x in a: print x, len(x)	Works through a, item by item, assigning each to x
	>>> range(10) >>> range(5,10)	range creates a numerical progression
	Try using for i in range(5):	$1 \operatorname{op}(a)$ is 2
	a = ['Shakespeare', 'Dickens', 'Hartley'] for i in range(len(a)): print i, a[i]	<pre>len(a) is 3, range(3) creates a list [0,1,2], these items are then assigned in turn to i</pre>
Tim O	a = ['Shakespeare', 'Dickens', 'Hartley'] for i, author in enumerate(a): print i, author	Alternatively use the enumerate function which returns both position & value

for exercise

• Write a Python program to calculate, for the integers 1 to 10, their sum and mean

for loop continued

• Example script to find prime numbers:

(ex5.py)

```
for n in range(2, 10):
  for x in range(2, n):
    if n % x == 0:
        print n, 'equals', x, '*', n/x
        break
  else:
```

loop fell through without finding a factor print n, 'is a prime number' x % y means "remainder of x / y"

break jumps out of the smallest
enclosing for (or while) loop
continue continues with the
next iteration of the loop

The else clause is executed when the loop terminates through exhaustion of the list (with for) or when the condition becomes false (with while), but not when the loop is terminated by a break statement.

Defining functions

 You are used to existing mathematical functions like sine or cosine (later we'll see how these are implemented in Python). Here we define our own function to calculate the Fibonacci series:

· · · · ·	<pre>def fib(n): # write Fibonacci series up to n """Print a Fibonacci series up to n. """ a, b = 0, 1 while a < n: print a, a, b = b, a+b</pre>	Function name fib is followed by list of parameters in parentheses
		Statements must be indented
		First statement is documentation, display with print fibdoc

- Either type this direct into interpreter or input the script into interpreter with import ex6
- Then to call the function type ex6.fib(2000)

ex6.py contains the function definition and is known as a module. A module can contain many function definitions and executable statements.

Running modules as scripts

• Can run a module as a script with an argument on the command line:

(ex7.py)
def fib(n):
 # write Fibonacci series up to n
 """Print a Fibonacci series up to n. """
 a, b = 0, 1
 while a < n:
 print a,
 a, b = b, a+b

 if __name__ == "__main__":
 import sys
 fib(int(sys.argv[1]))

 1st argument in list on command line

Within a module, the module's name (as a string) is available as the value of the global variable _____.

If the module is run as a script with python ex7.pythen __name__ is set to __main__ (as it is deemed the main module).

The extra statements at the end in this example allow it to be run with python ex7.py 50, for example, where 50 is taken as a command-line argument for function fib.

Scope of variables in functions

```
(ex8.py) def fib(n):
# write Fibonacci series up to n
"""Print a Fibonacci series up to n."""
a, b = 0, 1
while a < n:
    print a,
    a, b = b, a+b
print x
x = "Hi there"
#print a,
fib(2000)
#print a,</pre>
```

Variables defined within the function are local to the function. For variables referenced (by which we mean "used" i.e. on right hand side of assignment statement) in the function, interpreter looks first in the local symbol table, then outside (globally).

Just type python ex8.py

Try uncommenting the print a, statements or moving statements above the function definition

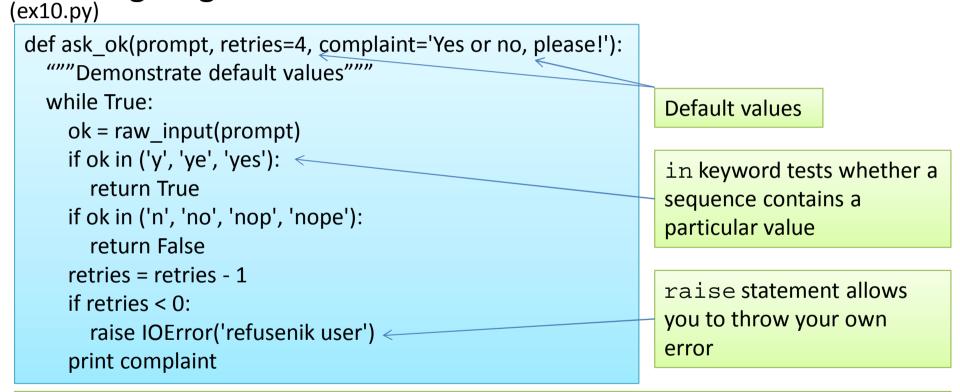
Functions which return values

• Every function returns a value whether explicitly or

	not	As fib did not define a return value, the value None is returned and is not usually printed result is a list object (initially empty). append is a method belonging to list objects (different types have different methods – you can define your own object types and methods using classes). result.append adds a new item to the list and is equivalent to result = result + [a]
(ex9.py)	<pre>>>> import ex6 >>> print ex6.fib(0)</pre>	
	<pre>def fib(n): # Fibonacci series up to n """Return a list containing Fibonacci series up to n. """ result = [] a, b = 0, 1 while a < n: result.append(a) a, b = b, a+b</pre>	
Time	O'Brien Phys 60441 Techniques of Rad	<pre>>>> import ex9 >>> ex9.fib(100) >>> answer=ex9.fib(100) >>> answer</pre>

Arguments to functions

• e.g. Arguments with default values:



Ways to call this function:

ask_ok('Do you really want to quit?')# with the one mandatory argument valueask_ok('OK to overwrite the file?', 2)# also with one of the optional argumentsask_ok('OK to overwrite the file?', 2, 'Come on, only yes or no!')# with all the arguments