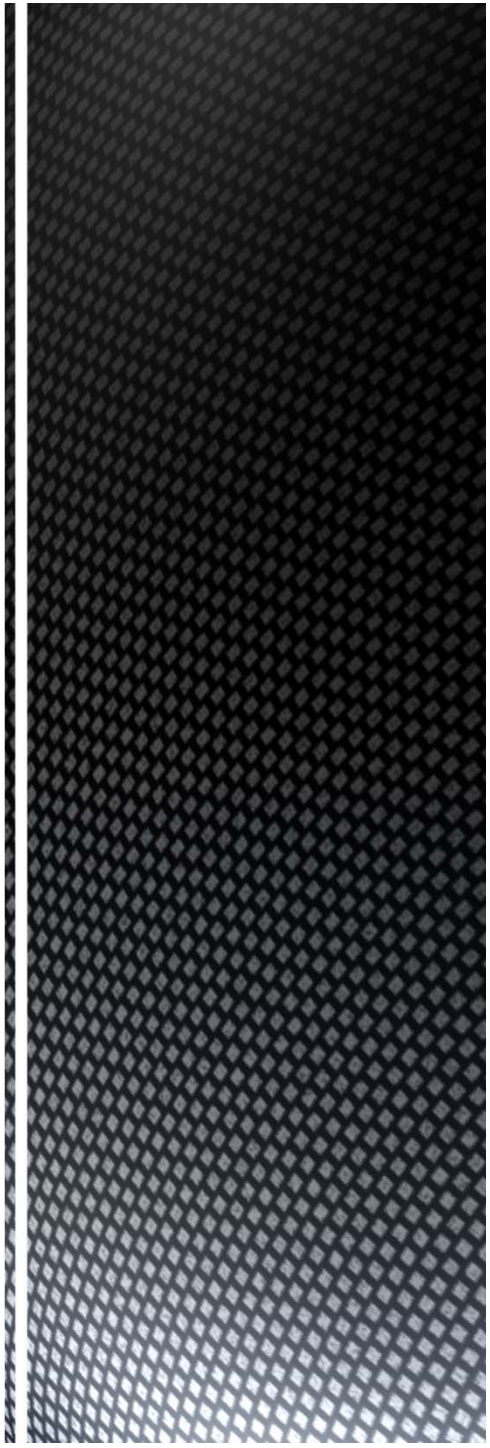
COMPSCI 107
Computer Science Fundamentals

Lecture 03 – Sequences of data



Learning outcomes

- At the end of this lecture, students should be able to:
  - Define and use functions
  - Import functions from modules

- Example code:
  - Perform calculations on the elements of a list
  - Determine if a number is prime or not
  - Convert a mark from a percentage into a letter grade

Range

- Used to generate integer numbers within a range of values
  - Includes the start value, but excludes the stop value

- range(stop)
  - range of values from 0 to stop value

```
range(10)
```

0 1 2 3 … 8 9

- range(start, stop)
  - range of values from start to stop value

```
range(3, 8)
```

3 4 5 6 7

- range(start, stop, step)
  - range of values from start to stop value, increasing by step each time

```
range(4, 10, 2)
```

4 6 8

```
range(10, 4, -2)
```

10 8 6

Functions

- A function is a sequence of instructions designed to perform a task, and is packaged as a unit.
  - Functions have a name
  - Functions accept arguments
  - Functions return values

```
def rectangle_area(width, height):
    return width * height
```

- Syntax
  - Indentation rather than braces are used to signify blocks of code
  - Variables defined within the *scope* of a function are not available outside the function

Exercises

- Write a function called triangle_area that calculates the area of a triangle given the base and height values
  - area = ½ base * height

```
>>> a = triangle_area(10, 20)
>>> print(a)
100.0
```

- Write a function that asks the user for the times tables to print, then prints out the times tables from 1 to 10 (inclusive)

```
Enter the times tables: 4
4 x 1 = 4
4 x 2 = 8
…
4 x 10 = 40
```

Example: Prime numbers

- Determine if a number is prime or not
  - A number less than 2 is not prime
  - A number equal or greater than 2 is prime if it is not divisible by any other number except itself and 1

```
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
```

Importing modules

- Code is stored in modules
  - We want to reuse code as much as possible
  - Build up libraries of code
  - Importing a module allows you to access code in that module

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sqrt(4)
2.0
```

- Python.org has a list of all the modules and functions provided

Built-in functions

- Python provides a wide range of built-in functions, including
  - round(value [, number of decimal places])
  - max(value, value [, value …])
  - min(value, value [, value …])
  - float(value)
  - int(value)
  - str(value)
  - type(value)

Sequences

- Sequences allow you to store values in an organized fashion.
  - strings, lists, tuples
  - http://en.wikibooks.org/wiki/Python_Programming/Sequences

Operation Name	Operator	Explanation
indexing	`[ ]`	Access an element of a sequence
concatenation	`+`	Combine sequences together
repetition	`*`	Concatenate a repeated number of times
membership	`in`	Ask whether an item is in a sequence
length	`len`	Ask the number of items in the sequence
slicing	`[ : ]`	Extract a part of a sequence

Strings

- Strings are a sequence of characters

```
>>> name = 'Andrew'
>>> name[0]
'A'
>>> 'd' in name
True
>>> len(name)
6
>>> name + ' ' + 'Luxton-Reilly'
'Andrew Luxton-Reilly'
```

- Strings also have a number of other functions that can be used
  - split() is especially useful

Exercise

- Write a function that determines whether a given string of text contains a vowel or not. The function should return True if the text contains a vowel.

Lists

- Lists are a built-in type in Python
  - Use square brackets to signify a list
  - Lists can contain any type of data, or any mixture of data

```
>>> [1, 2, 3]
[1, 2, 3]

>>> ['Hello', 'Is', 'there', 'anybody', 'out', 'there?']
['Hello', 'Is', 'there', 'anybody', 'out', 'there?']

>>> [1, 5.899, 'Hello']
[1, 5.899, 'Hello']
```

List functions

- Numerous list functions are supported
  - Use help(list) to find out the functions
  - Use Python.org to find out more

```
>>> x = [1, 2, 3]
>>> len(x)
3
>>> x + [4]
[1, 2, 3, 4]
>>> x += [5]
[1, 2, 3, 5]
>>> 3 in x
True
>>> x[0]
1
```

Exercises

- Write a function that sums the elements of a list that contains numbers.

- Write a function that accepts a list and returns a new list with the same contents, but in reverse order.

Slices of sequences

- A piece of a sequence can be obtained using the following syntax
  - sequence_name[x:y]
  - where x is the index of the first element and y is the index after the last element

```
>>> name = 'Andrew'
>>> name[0:0]
''
>>> name[0:1]
'A'
>>> name[1:4]
'ndr'
```

Slice step value

- Actually, the syntax allows for a third value, used to define the step size between elements included in the slice. If a value if omitted, it defaults to [start:end:1]

```
>>> name = 'Andrew'
>>> name[:4:2]
'ad'
```

- If the step size is negative, it starts at the end and steps backward towards the start.

```
>>> name = 'Andrew'
>>> name[::-1]
'werdnA'
```

For loops

- Used to iterate through a sequence

```
numbers = [2, 3, 5, 7, 11]
for i in numbers:
    print(i)
```

```
name = "Andrew"
for c in name:
    print(c)
```

Exercises

- Write a function that returns a list containing the squares of all the values between 1 and n (inclusive).

```
>>> squares(5)
[1, 4, 9, 16, 25]
```

- Write a function that counts the number of vowels in a given string.

Summary

- Functions are designed to perform a well-defined task.
  - Functions can be defined with parameters to generalise a task
  - Functions always return a value, the default return value is None

- Sequences consist of data that is organised and indexed as a sequence of values