ECMM455 Python Worksheet 12: Modules & Random numbers

Prof Hywel Williams

September 12, 2019

1 Aims

- Learn to import modules
- Practice using random numbers

2 Modules

2.1 Modules

A module is a file containing Python function definitions. Modules can be imported into another program:

import <module_name>

Functions from within the module can then be called using the module and function name:

module_name.function_name()

Use of modules allows programmers to benefit from good code written by others. Python is distributed with a collection of functions and modules called the "standard library". Some are built-in, others need to be imported.

2.2 Example: the *random* module

This module in the standard library contains a selection of useful functions for generating and using (pseudo)random numbers. For example:

- *random()* - returns a random float in range [0.0,1.0]
- *randint(a,b)* - returns a random integer *n* such that *a<=n<=b*
- *choice(seq)* - returns a randomly chosen member of the sequence *seq* (e.g. a list or string)
- *shuffle(x)* - performs a random permutation of the list *x* (works in place, returns *None*)
- *sample(seq,k)* - returns a list of *k* randomly chosen items from the sequence *seq*

The code below imports the *random* module and uses it to display three random integers in the range [0,10]:

```
import random
print(random.randint(0,10))
print(random.randint(0,10))
print(random.randint(0,10))
```

The code below imports the *random* module (giving it a short name of *rnd*) and generates a random sample of 3 numbers from a given list:

```
import random as rnd
s = rnd.sample([2,4,6,8,10],3)
```

3 Exercises: Using random numbers

1. In interactive mode, try the following:

   (a) Import the *random* module

   (b) Use the *random()* function from that module to generate a random number in range [0,1].

   (c) Create a list of several types of fish, then use the *choice()* function to randomly select one of them.

   (d) Use *shuffle()* to reorder your list of fishes. Display the reordered list to see how it changes.

2. Write a program called *dice.py* that performs a simulated diceroll (i.e. displays a randomly selected number from 1 to 6).

3. Write a program called *russian_ roulette.py* that, in each round, performs a probability test with P=1/6 to see if the player is killed, then displays an appropriate message. Further rounds should be played until eventually the player is killed.

4. Write a simple game called *darts.py* in which the computer "throws three darts" (i.e. generates three random numbers between 1 and 20). The user should then be prompted to throw three darts of their own, one at a time (e.g. by using raw_input() to allow the user to choose when to throw, then generate a random score for that dart). If the user's total score is higher than the computer, the user wins.