John Romero Programming Proverbs

- 1. ''No prototypes. Just make the game. Polish as you go. Don't depend on polish happening later. Always maintain constantly shippable code. (Large teams require more planning though.)''

- John Romero, ''The Early Days of Id Software - John Romero @ WeAreDevelopers Conference 2017''

The first lecture

- this module consists of two pieces of coursework
  - in the first term, Missile Commmand ⟨`CS2S566_CW1P1M_Cover_PRCW_PRACTCW1.pdf`⟩ implemented in Python3 and Pygame
  - in the second term, a map editing tool for a tablet implemented in Python3

- both pieces of coursework are worth 50%

Access to the software in this module

- in this module Python3 will be taught on the GNU/Linux operating system

- there are two supported approaches to run Python3
  - firstly using vmware
  - secondly using the Rasbperry Pi-4

- both give the same user level experience

- please see the other two components of the lecture this week for more details on either approach

Python

- Python is a scripting language

Python Gotha's

- blocks are defined by indentation!

- turn off tabs in your favourite editor

- in your own programs examples never create a name clash with a Python library module

- Python2 vs Python3
  - we will be using Python3

Python verses similar tools

- Python is a scripting language
  - it can be compiled if necessary to increase speed

- is more powerful than many other scripting languages, Tcl
  - applicable to larger systems development (games, net admin)

- has a much cleaner syntax than Perl
  - easier to maintain

- does not compete head on with Java
  - Java is a systems language like C++

Python and games

- examples of games which use Python ⟨`http://wiki.python.org/moin/PythonGames`⟩

Python can be simple

Python Modules allow for problem decomposition

- similar to Modula-2

myfile.py

foo.py

- when run prints hello world

Alternative import

■   `bar.py`

■ note that all python modules need to be saved as *name*.py

   ■   so in our example the module `myfile` was saved into a file called `myfile.py`

Python builtin types

- python contains many builtin types
  - use them..

- builtin objects make simple programs easy to understand
  - lists, dictionaries, exist, don't reinvent the wheel

- built in objects are more efficient than custom data types

Builtin objects

■

numbers	3.14159, 1234
strings	'spam', "fred's"
lists	[1, [2, 'three'], 4]
dictionaries	{'food':'spam', 'taste':'yum'}
tuples	(1, 'spam', 4, 'U')
files	text=open('/etc/passwd', 'r').read()

Expression operators

■

or, and, not	logical operators (short circuit)
<, <=, >, >=, ==, <>, !=	comparison operators
x | y	bitwise or
z & y	bitsize and
x << y	shift left by y bits
x >> y	shift right by y bits
x[i]	indexing
x[i:y]	slicing
x.y	qualifying (imports)
x(y)	function calls

Strings

- concatenation via +
  - repeated via *

-

- yields

- ```
  hi hi hi hi
  ```

Slicing

- given a string, `s= "hello world"`
  - can obtain portion of string via: `s[2:5]`
  - yields: `llo`

- first character has index 0
  - and also -11
  - last character index is 10 in this example
  - last character index is also -1

- negative values start at right and move to the left

- strings can be sliced using positive and negative values

Using dir

- often you may wish to see what methods a module provides

  - run python interactively

- ```
  python
  Python 1.5.2
  >>> import string
  >>> dir(string)
  ['capitalize', 'capwords', 'center', 'count', \
   'digits', 'expandtabs', 'find', 'hexdigits', \
   'index', 'index_error', 'join', 'joinfields', \
   'letters', 'ljust', 'lower', 'lowercase', \
   'lstrip', 'maketrans', 'octdigits', 'replace', \
   'rfind', 'rindex', 'rjust', 'rstrip', 'split', \
   'splitfields', 'strip', 'swapcase', \
   'upper', 'uppercase', 'whitespace', 'zfill']
  ```

- displays methods available

Methods and documentation

- python online docs ⟨`http://floppsie.comp.glam.ac.uk/python/html/index.html`⟩
  - under GNU/Linux

- tutorial/laboratory
  - read through the online tutorial under the web address above
  - read about functions and scope rules
    - name resolution, LGB rule
    - local, global, builtin scope

Statements

- assignment, calls, if/else/elif, for, while, break/continue
    - `print` used to be a statement in Python 2, it is a function in Python 3

- try, except, raise,

- def, return
    - function definitions and returning values

Statements

- class

- assert

- exec

- del

- global

Example 8 times table

-

Example 8 times table

```
$ python3 eight.py
1 x 8 = 8
2 x 8 = 16
3 x 8 = 24
4 x 8 = 32
5 x 8 = 40
6 x 8 = 48
7 x 8 = 56
8 x 8 = 64
9 x 8 = 72
10 x 8 = 80
11 x 8 = 88
12 x 8 = 96
```

Example of for loop

-

Example of for loop

```
./py7.py
n is 2
n is 3
n is 4
n is 5
n is 6
n is 7
n is 8
n is 9
finished for loop, n is 9
```

Tricky example code

Tricky example code

```
./py6.py
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

Graphical hello world as an example of Python simplicity

```
#!/usr/bin/python3

import Tkinter

def makebutton(message):
        w = Tkinter.Button(text=message, command='exit')
        w.pack()
        w.mainloop()


makebutton("Hello world")
```

Tutorial

- to undertake these tutorials you will need to refer to the python online docs ⟨`https://docs.python.org/3`⟩

- write a program using a while loop
  - to write out the nine times table

- write a program using a for loop
  - to write out the seven times table

- write a program using a function and if else statement
  - to write out the 3 times table