Lecture: 2-1

- Prerequisites for this lecture are: 1-1 and .

John Romero Programming Proverbs

- 2. “It’s incredibly important that your game can always be run by your team. Bulletproof your engine by providing defaults (for input data) upon load failure.”

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

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

```
#!/usr/bin/python3

print("hello world")
```

Python Modules allow for problem decomposition

- similar to Modula-2

- **myfile.py**

```
#!/usr/bin/python3

title = "hello world"
```

- **foo.py**

```
#!/usr/bin/python3

import myfile
print(myfile.title)
```

- when run prints hello world

Alternative import

- **bar.py**

```
#!/usr/bin/python3

from myfile import title
print(title)
```

- 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 *

- ```
  #!/usr/bin/python3

  print("hi " * 4)
  ```

- 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

```
#!/usr/bin/python3

for n in range(1, 13):
    print(n, "x 8 =", n*8)
```

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

```
#!/usr/bin/python3


for n in range(2, 10):
    print("n is", n)
else:
    print("finished for loop, n is", n)
```

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

```
#!/usr/bin/python3

for n in range(2, 10):
    print("n is", n)
    for x in range (2, n):
        print("x is", x)
        if n % x == 0:
            print(n, "equals", x, "*", n/x)
            break
    else:
        print(n, "is a prime number")
```

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
```

Install tkinter for Python

- in your GNU/Linux installation start a terminal and type:

- ```
  $ sudo apt install python3-tk
  ```

- answer `Y` to all the questions, this will install tkinter for Python3 permanently on your system
  ```

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")
```