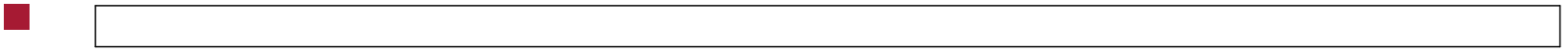
John Romero Programming Proverbs

- 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.”

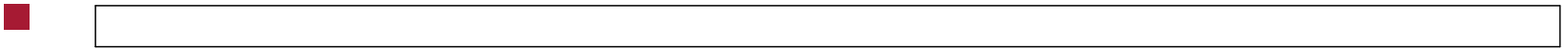
- John Romero, “The Early Days of Id Software - John Romero @ WeAreDevelopers Conference 2017”

Python While Loop



Python Functions

if statement and functions



Python Modules

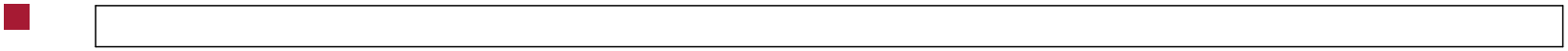
- there are many Python modules available
- which cover many topics
  - networking modules
  - graphic modules, OpenGL, GUI, graphing
  - mail, http, telnet, pop3, imap modules
  - operating system modules
- html parsing modules
- examine the Python modules python online docs ⟨`http://floppsie.comp.glam.ac.uk/python/html/index.html`⟩

urllib

- used to download files from servers using
  - ftp, http and local file access

urllib example

urllib example



smtp module

- Simple Mail Transport Protocol is the most common protocol whereby email is transmitted across the Internet

-

Python Gotya's

- be careful to ensure that your code is indented correctly
- be very careful not to name your file to a name used by a library you are importing

Python Gotya's

- for example do **not** call this file `string.py`

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

  import string

  words=string.split("hello world again")
  print words
  ```
  ```

Python Gotya's

- the python interpreter will read your file twice
  - one when you run the file
  - and again when it comes across the `import string` !

- name the file `teststring` and it will work fine
  - if you did call it `string.py` and run then you will need to remove `string.py` and also `string.pyc`

Python and file handling

- file manipulation primitives are by default available
  - no need to import library to, read, write files

Python and file handling

- creating a simple text file

-

Python and file handling

Python and file handling

- many ways to read a file
  - `file.read()` returns a string containing all characters in the file
  - `file.read(N)` returns a string containing next N characters
  - `file.readline()` returns a string containing characters up to \n
  - `file.readlines()` returns the complete file as a list of strings each separated by \n

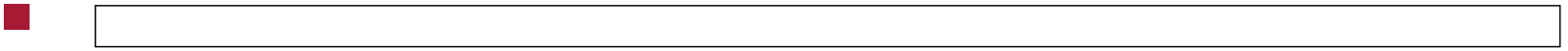
Further Python Networking

- many python modules which give access to application layer networking services
  - ftp, http, telnet, etc

Further Python Networking

- sometimes you may have to implement your own application layer protocol
- in which case you use `sockets` (a transport layer service)

server.py



client.py

To run the server client example

- open up another terminal and type this at the command line
-
```
$ python3 server.py
```
- open up another terminal and type this:
-
```
$ python3 client.py
```

IMAP library

Arguments in Python

- `getopts`, provides a useful method for handling arguments
  - in fact many languages have adopted getopts
  - C, C++, bash and python

Autoftp arguments in python

Autoftp arguments in python

- notice that the script fails if an unsupported option is issued

- ```
  ./autoftp2.py -x
  ...
  getopt.GetoptError: option -x not recognised
  ```
  ```

Better argument handling

- so we need a way to trap these errors
  - python uses an exception handler for this
-

Better argument handling

- when run it yields the following

```
./autoftp3.py -x
autoftp [-v][-p][-h]
```

When is a module not a module?

- it is often useful to create a module
  - for yourself and others to use in the future
  - to subdivide the large problem set into a number of smaller modules

- sometimes a module might be able to operate as a stand alone program
  - consider autoftp could be organised as a module

When is a module not a module?

```
if __name__ == "__main__":
    main()
```

- which means run the function `main` if this module is explicitly invoked by the user
  - note that it is not run if this module was imported

Example times module

Example program

Example program

- note that the module times takes a string and adds a ’0’ to the left hand side
  - effectively multiply by 10

- note it also uses the `if __name__ ==` condition which only calls the multiply routine if this module was invoked as the main program by the user

Example program

- ```
  ./prog.py 12
  importing the times module
  120
  ```

- ```
  ./times.py 12
  testing the times module
  120
  ```

- exercise for the reader, add a function to perform divide and modulus of a numerical integer string

printf

- if any C programmer laments the lack of a `printf` function, you can roll your own:

- **mylibc.py**

```
#!/usr/bin/python3

#
#  printf - keeps C programmers happy :-)
#

def printf (format, *args):
    sys.stdout.write (str (format) % args)
    sys.stdout.flush ()
```

- please create this file (module) as it will be very useful when you start the coursework

printf

- **mytest.py**

```
#!/usr/bin/python3

from mylibc import printf

printf ("hello world\n")
printf ("an int: %d\n", 42)
printf ("a float: %f\n", 3.1415927)
```

- why does the output for a float differ from the constant value?

printf

mytest2.py

```
#!/usr/bin/python3

from mylibc import printf

printf ("hello world\n")
printf ("an int: %d\n", 42)
printf ("a float: %f\n", 3.1415927)

printf ("a float: %19.19f\n", 3.1415927)
```

Tutorial

- type in the printf example given during the lecture and check that it works

- create the file `mylibc.def` and also create the test programs
  - try running the test programs
  - you have created your first Python module `mylibc.def`

- try out any other examples from this weeks lecture notes