Scripts, Modules and Variables

Fermilab - TARGET 2018
Week 2

Python script

Import statements
to use other modules

Comments
Starting with #

Outside statements
Executed when running the script

```
#!/usr/bin/python
import os
import sys
import string
import optparse
...
from glideinwms.lib import condorExe
# Main function
def main(argv):
  feconfig=frontenvparse.FEConfig()  # FE configuration holder
  … # parse arguments
  feconfig.config_optparse(argparser)
  (options, other_args) = argparser.parse_args(argv[1:])

  if len(other_args)<1:
     raise ValueError, "Missing glidein_name"
  glidein_name = other_args[0]
  if len(other_args)>=2:
     log_type=other_args[1]
  else:
     log_type="STARTD"
  ...
  return 0
# S T A R T U P
if __name__ == '__main__':
  try:
     sys.exit(main(sys.argv))
  except Exception, e:
     sys.stderr.write("ERROR: Exception msg %s\n"%str(e))
     sys.exit(9)
```

#! - shebang
Indicates which interpreter can run the script

Block
Groups together a list of statements
4 spaces indentation!

Function
Gives a name to a block of code and hides variables

__main__
Value of __name__ when this is invoked as a script (as alternative could be imported)

Structuring files

Module - single file of Python code (definitions and statements) that is meant to be imported

Package - collection of Python modules under a common namespace.
In practice one is created by placing multiple python modules in a directory with a special __init__.py module (file)

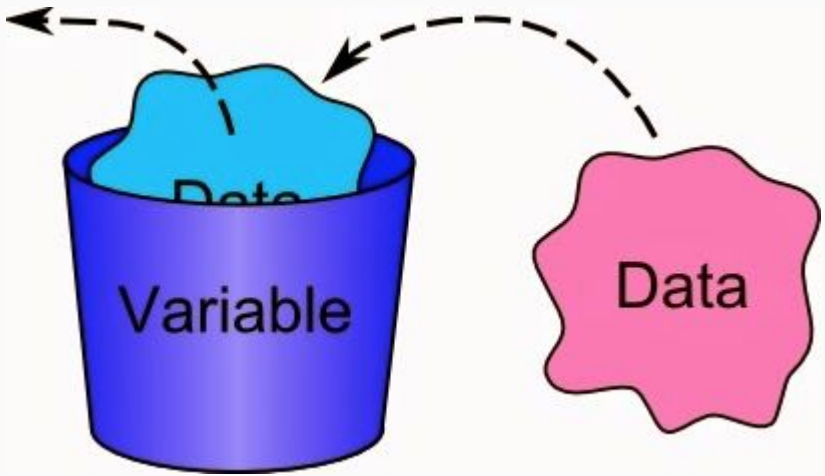
Library - generic term for code that was designed with the aim of being usable by many applications. PYTHONPATH lists known libraries.

Standard library - collection of exact syntax, token and semantics of the Python language which comes bundled with the core Python distribution (>200 modules)

Variable

Variables are used to store information to be referenced and manipulated in a computer program

Variables have a name, value, representation, a type



Variables scope and lifetime

Scope - part of a program where a variable is accessible

Lifetime - duration for which the variable exists

Global variable - defined in the main body of a file

It will be visible throughout the file, and also inside any file which imports that file.

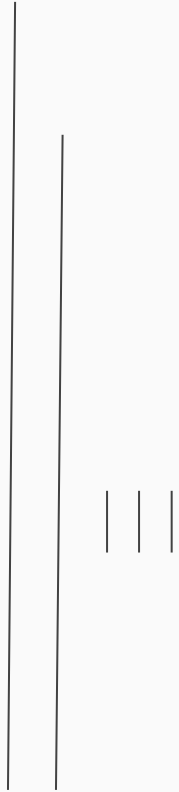
Local variable (to a function) - defined inside the function

It is accessible from the point at which it is defined until the end of the function, and exists for as long as the function is executing.

Variable lifetimes and scopes, an example

Lifetime (memory used)

a b c d a(loc)



Scope (name can be used)

a b c d a(loc)

```
# This is a global variable
a = 0

if a == 0:
    # This is still a global variable, becomes alive only if a is 0
    b = 1

def my_function(c):
    # This is a local variable
    d = 3
    print(c)
    print(d)
    # This is a local variable too, hiding the global a
    a = 5
    print(a)

# Now we call the function, passing the value 7 as the first and only parameter
my_function(7)  # This prints 7, 3, 5

# a and b still exist
print(a)  # This prints 0, the value of the global a
print(b)

# c and d don't exist anymore -- these statements will give us name errors!
print(c)
print(d)
```

Global Variables

Defined outside: a, b

Local variables

To my_function: c, d, a(loc)

Error!

You cannot refer (local) variables outside their scope

Knowing how to code may come handy...

Remember to choose a project for your presentation!

Here a creative example of someone using coding abilities to solve a problem

Project Mayhem writing some code to fight fake IRS phone scam:

https://www.youtube.com/watch?v=EzedMdx6QG4