```
*************************************************************
  maXbox Starter 98
*************************************************************
```

Solutions to run Python in a Box

Max Kleiner

I rant, therefore I am.

When you run a Python script, the interpreter converts a Python program into something that that the computer can understand. Executing a Python program can be done in two ways: calling the Python interpreter with a shebang line, and using the interactive Python shell or to invoke the script file from a script shell editor like crontab or maXbox. I show 4 solutions.

Suppose you have a python script file httpheader1.py:
"G:\maXbox\works2022\maxbox4\examples\httpheader1.py"

```
               ____  ______.
  _____ _____  \   \/   /\_ |__   _______  ___
 /     \\__  \  \     /  | __ \ /  _ \  \/  /
|  Y Y  \/ __ \_/     \  | \_\ (  <_> >    <
|__|_|  (____  /___/\  \ |___  /\____/__/\_ \
      \/     \/      \_/     \/            \/
```

The best way to organize python-routines is to call it from a script. So you can log and control your scripts in time and structure. For example scheduling python scripts with crontab or maXbox as a script shell is fundamental when it comes to automating tasks using python. We will see how to schedule python scripts and pass the necessary parameters as well.

So in our httpheader1.py we just have to pass one argument, namely an URL to get some header information back:

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("-u", "--url", required=True,
             help="Full URL - http://www.freecybersecurity.org")
    args = vars(parser.parse_args())
    url = args["url"]
    print(funcHTTPHeader(url))
```

A first solution for returning the header is the function getDosOutput():

```
 println(getDosOutput('py examples\httpheader1.py --url
"https://www.ipso.ch/ibz"',exePath));

>>>
Cache-Control: max-age=3600, public
Content-Language: de
Content-Type: text/html; charset=UTF-8
Date: Thu, 11 Aug 2022 08:35:14 GMT
Etag: "1660206911"
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Last-Modified: Thu, 11 Aug 2022 08:35:11 GMT
Link: <https://www.ipso.ch/>; rel="shortlink", <https://www.ipso.ch/ibz>;
rel="canonical", <https://www.ipso.ch/ibz/die-schweizer-schule-fuer-
technik-und-management>; rel="alternate"; hreflang="de",
<https://www.ipso.ch/ibz/die-schweizer-schule-fuer-technik-und-
management>; rel="revision"
```

```
48: Server: Apache/2.4.48 (Unix) OpenSSL/1.1.1k
49: Strict-Transport-Security: max-age=31536000; includeSubDomains
50: Vary: Cookie
51: X-Content-Type-Options: nosniff
52: X-Drupal-Cache: HIT
53: X-Drupal-Dynamic-Cache: UNCACHEABLE
54: X-Frame-Options: SAMEORIGIN
55: X-Generator: Drupal 8 (https://www.drupal.org)
56: X-Ua-Compatible: IE=edge
57: Connection: close
58: Transfer-Encoding: chunked
59:
60: Another simpler one is the direct shebang line:
61:
62: py examples\httpheader1.py --url "https://www.ipso.ch/ibz"
63:
64: Or in Linux we have saved this script as httpheader1.py under our home
    directory, we can make it executable by entering the following command in
    a terminal:
65:
66: $ sudo chmod +x httpheader1.py
67:
68: Note: #!/usr/bin/python3 (specifying path of script interpreter at first
    line) is necessary if you wish to make the script executable.
69: #! /usr/bin/python3
70:
71: To schedule this script to be executed, we need to enter a crontab
    scheduling expression into the crontab file. To do that, simply enter the
    following in a terminal:
72:
73:   crontab -e
74:
75: Then you might be prompted to select an editor, choose nano or maXbox and
    append the following line to the end of the opened crontab file:
76:
77: */2 * * * * /home/$(USER)/httpheader1.py --url"http:/.." >>
    /home/$(USER)/headeroutput.txt
78:
79: A note to getDosOutput(): I have a commandline application coded in delphi
    that I need to call from a normal desktop application (also coded in
    delphi). In short, I want to call the commandline app and display the text
    it outputs "live" in a listbox or memo.
80: So theres a difference if it reads the output in one step or showing how
    the output can be read while the process is still running:
81:
82: Second solution to call a Python script from the box is with the
    Python4Delphi engine:
83:
84:    eg:= TPythonEngine.Create(Nil);
85:    eg.pythonhome:= PYHOME32;
86:    eg.opendll(PYDLL32)
87:    writeln(' ');
88:
    eg.execstr(filetostring('C:\maXbox\works2021\maxbox4\examples\httpheader1.py'));
89:    println(eg.evalstr('funcHTTPHeader("https://www.ipso.ch/ibz")'));
90:
91: So with this function evalstr() takes an expression as an input and
    returns the result of the expression on evaluation in the box. The
    advantage in comparison with DosOutput is the access to the variables and
    members of the script in our script. The access can be a global or local
    scope in the context of the shell script.
92:
```

```
 93: The arguments are a string and optional globals and locals. If provided,
     globals must be a dictionary. If provided, locals can be any mapping
     object. That means we dont need a main function or a main part in the
     httpheader1.py we can call (after execute the script with eg.execstr) the
     function funcHTTPHeader direct.
 94:
 95:
     eg.execstr(filetostring('C:\maXbox\works2021\maxbox4\examples\httpheader1.py'));
 96:    println(eg.evalstr('funcHTTPHeader("https://www.ipso.ch/ibz")'));
 97:
 98: >>> Cache-Control: max-age=3600, public
 99: Content-Language: de
100: Content-Type: text/html; charset=UTF-8
101: Date: Thu, 11 Aug 2022 12:22:55 GMT
102: Etag: "1660220574"
103: Expires: Sun, 19 Nov 1978 05:00:00 GMT
104: Last-Modified: Thu, 11 Aug 2022 12:22:54 GMT
105: ▯▯▯ mX4 executed: 11/08/2022 14:22:55  Runtime: 0:0:5.620  Memload: 47%
     use
106:
107: Many programming languages have a special function that is automatically
     executed when an operating system starts to run a program, usually called
     main() and must have a specific return type and arguments according to the
     language standard. On the other hand, the  interpreter executes scripts
     starting at top of the file, and there is no specific function that it
     automatically executes.
108:
109:     args = vars(parser.parse_args())
110:     url = args["url"]
111:     print(funcHTTPHeader(url))     #return in main()
112:
113: In this code, there is a function called main() that prints the return of
     the function funcHTTPHeader(url) when the Python interpreter executes it.
     In the sense of a shell script we import the code as a module. But whats
     the difference between script and module:
114:
115: A script is a file that you intend to execute from the command line to
     accomplish a task.
116: A Python module is a file that you intend to import from within another
     module or a script, or from the interactive interpreter. In our case
     maXbox is the within another script. You can read more about modules in
     Python Modules and Packages – An Introduction.
117: https://realpython.com/python-main-function/
118:
119: The last and 4. solution is the complete integration of a script within a
     script, sounds complicated but is more work than brain.
120: We define the whole script from the file as a integrated const in our box:
121:
122: Const HTTP_HEADER_DEF =
123: 'import urllib.request                              '+LF+
124: 'import argparse                                    '+LF+
125: '############## HTTP Header ##############         '+LF+
126: 'def funcHTTPHeader(url):                           '+LF+
127: '  agent= {"User-Agent":"Mozilla/5.0 (OS X 10.13) Gecko/201
     Firefox/62.0"}'+LF+
128: '  req = urllib.request.Request(                    '+LF+
129: '          url,                                     '+LF+
130: '          data=None,                               '+LF+
131: '          headers=agent                            '+LF+
132: '          )                                        '+LF+
133: '  try:http = urllib.request.urlopen(req)                    '+LF+
134: '  except urllib.error.URLError as err:                      '+LF+
```

```
'      print("ERROR: {} {}".format(err.code,err.reason))      '+LF+
'  return http.headers                                        ';

   eg.Execstring(HTTP_HEADER_DEF);
   println(eg.evalstr('funcHTTPHeader("https://www.ipso.ch/ibz")'));
```

When the integrated interpreter python engine <eg.Execstring> reads a Python **file or** the **const in** our **case, it** first sets a few special variables **of** the scope. **Then it** executes the code from the **file or** the **const in** our script. Its almost the same **as** solution 3 **with** the difference **of** the script integration, solution 3 can als be a net share **or** remote **file**, solution 4 **is** integrated **as const and** can be edited direct **in** the box.

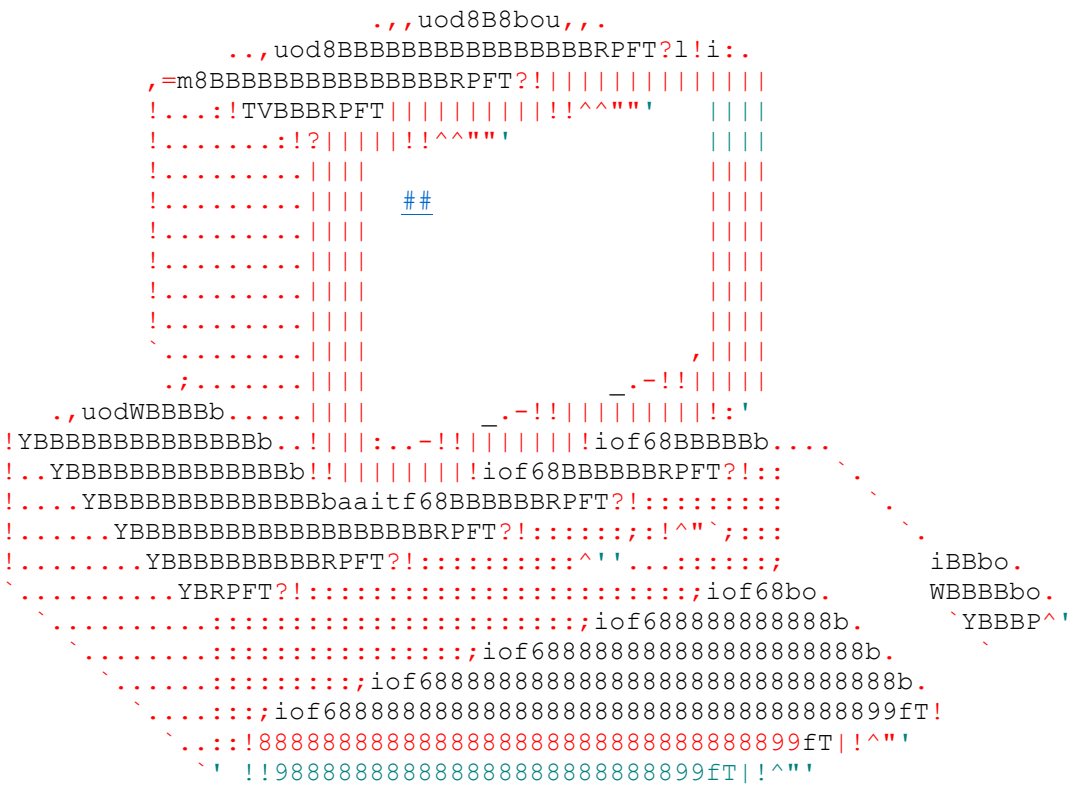
```
eg.execstr(filetostring('C:\maXbox\works2021\maxbox4\examples\httpheader1.py'));
   println(eg.evalstr('funcHTTPHeader("https://www.ipso.ch/ibz")'));
```

Python files are called modules **and** they are identified by the .py **file** extension. A module can define functions, classes, **and** variables. Again, when running file_one you will see that the **program** recognized which **of** these two modules **is** __main__ **and** executed the code according **to** our first **if else** statements:

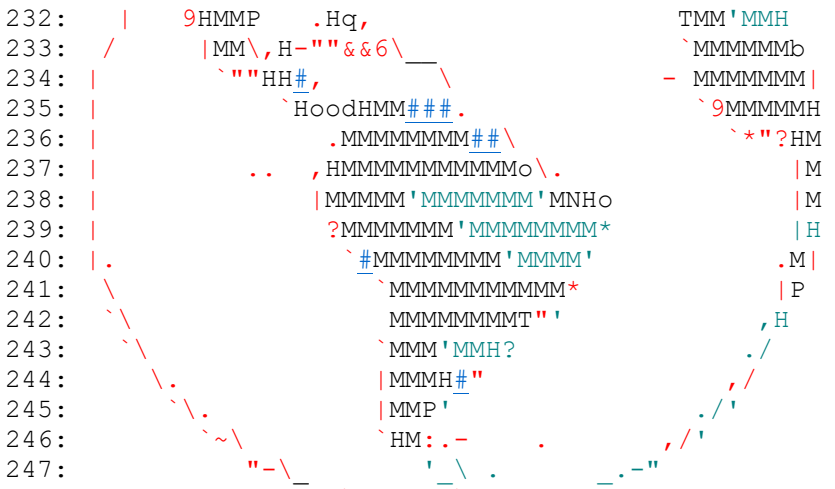
```
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
```

When modules like this httpheader1.py **with** just one single **function** funcHTTPHeader are being imported **and** run, their functions will be imported, **and** top level code executed

Sort **of** a microservice you find at:
https://www.academia.edu/31112544/Work_with_microservice_maXbox_starter48.pdf



```
                `!!888888888888888899fT|!^"'
                  `!988888888899fT|!^"'
                    `!9899fT|!^"'
                      `!^"'


Conclusion:
-----------------------------------------------------------------
 The idea of separating a script into another script is nothing new; there
are other programming paradigms which address this same concept, such as
Service Oriented Architecture (SOA) or POST-Services. Translation is the
written transfer of a message from one language to another. Interpretation
refers to oral translation, but can also mean interpreting the meaning and
intention of the message. Translation is made from the source language to
the target language.
 You may also see Python scripts executed from within packages by adding
the -m argument to the command. Most often, you will see this recommended
when you're using

      pip: python3 -m pip install 'package_name'.

 Adding the -m argument runs the code in the __main__.py module of a
package.

 We have a 4 solution step focus in this tutorial:
 ------------------------------------------------

 1. Call the script from terminal or shell:

    > py examples\httpheader1.py --url "https://www.ipso.ch/ibz"


 2. Reroute the script to another output:

    > println(getDosOutput('py examples\httpheader1.py --url
"https://www.ibz.ch"',exePath));


 3. Integrate the interpreter:

eg.execstr(filetostring('C:\maXbox\works2021\maxbox4\examples\httpheader1.py'));
    > println(eg.evalstr('funcHTTPHeader("https://www.ipso.ch/ibz")'));


 4. Integrate the script and the interpreter:

    eg.Execstring(HTTP_HEADER_DEF);
    > println(eg.evalstr('funcHTTPHeader("https://www.ipso.ch/ibz")'));


 Simplify API development for users, teams, and enterprises with the
Swagger open source and professional toolset. Find out how Swagger can
help you design, document your APIs at scale.

 >>>

                _od#HMM6&*MMMH::-_
            _dHMMMR??MMM?  ""|  `"'-?Hb_
         .~HMMMMMMMMHMMM#M?          `*HMb.
       ./?HMMMMMMMMMMM"*"""             &MHb.
      /'|MMMMMMMMMMM'                -    `*MHM\
     /   |MMM'MMHHM''                      .MMMHb
```



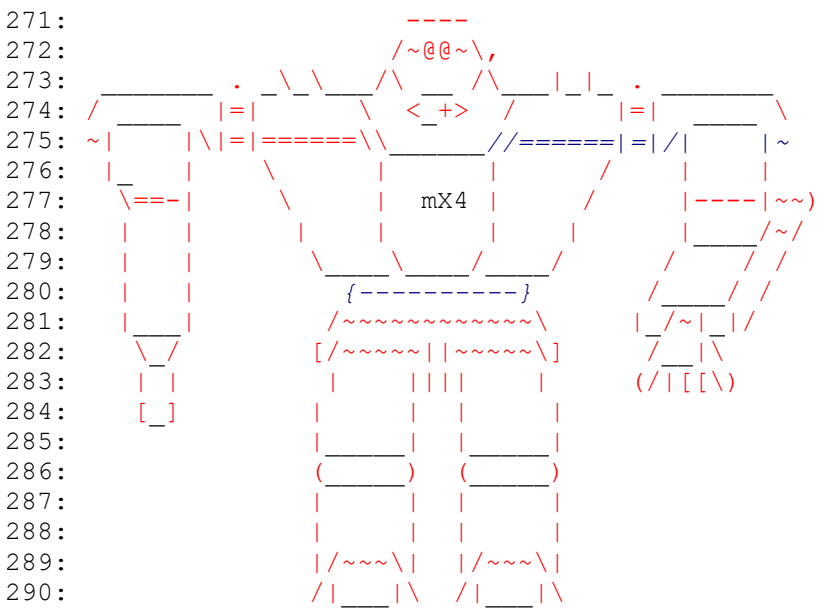
This is what we want to use in maXbox:

```
const res = await fetch("https://libretranslate.com/translate", {
method: "POST",
body: JSON.stringify({
 q: "Hello!",
 source: "en",
 target: "es"
}),
headers: { "Content-Type": "application/json" }
});
console.log(await res.json());
```

Source of the integrate script at: 1150_pydemo42httpheader.txt

http://www.softwareschule.ch/examples/1150_pydemo42httpheader.txt

Each service should be independently developed and deployed. No coordination should be needed with other service teams if no breaking API changes have been made. Each service is effectively it it's own codebase and lifecycle.



```
               <_______><_______>

 A microservice architecture shifts around complexity. Instead of a single complex system, you have a bunch of simple services with complex interactions.

  Doc:
  https://www.freecodecamp.org/news/if-name-main-python-example/
  https://www.geeksforgeeks.org/crontab-running-a-python-script-with-parameters/
  https://www.onworks.net/software/windows/app-maxbox
  http://www.softwareschule.ch/examples/1142_list_collections_pydemo42.txt
http://www.softwareschule.ch/examples/1145_271_closures_study_op_sys_tutor97.txt

  -----_____
                   _____------           __      ----_
            ___----                 ___------              \
               ----________        ----                     \
                           -----__    |             _____)
                                __-                /     \
                    _______-----    ___--          \    /)\
              ------_______      ---____            \__/  /
                           -----__    \ --    _          /\
                                  --__--__     \_____/   \_/\
                                          ----|   /          |
                                              |   |__________|
                                              |   | ((_(_)| )_)
                                              |   \_((_(_)|/(_)
                                              \              (
                                               \_____________)


namespace RegExApplication
    {
        class Program
        {
            private static void showMatch(string text, string expr)
            {
                int arex = 46;
                Console.WriteLine("The Expression: " + expr + ' '+arex);
                MatchCollection mc = Regex.Matches(text, expr);
                foreach (Match m in mc)
                {
                    Console.WriteLine(m);
                }
            }
            static void Main(string[] args)
            {
                string str = "A Thousand Splendid Suns";
                Console.WriteLine("Matching words that start with 'S': ");
                showMatch(str, @"\bS\S*");
                FileIOApplication.FileProgram.Mainfiler();
                BinaryFileApplication.Program.Mainbinary();
                WindowsFileApplication.Program.Mainwin();
               //
DelegateAppl.BoilerEventAppl.RecordBoilerInfo.Mainboiler();
                //DelegateAppl.TestDelegate.MainDelegate(["Arg 1","Arg 2",
"Arg 3"]);
                //  DelegateAppl.TestDelegate.MainDelegate("Arg 3");

                //System.Windows.Input.ICommand.Equals.
```

```
347:
348:                 Console.ReadKey();
349:                 foreach (var arg in args)
350:                 {
351:                     Console.WriteLine(arg);
352:                 }
353:                 /* for(int i = 0; i < args.length; i++) {
354:                     Console.WriteLine(args[i]);
355:                     }
356:                 Console.ReadKey();*/
357:                 //205
358:             }
359:         }
360:     }
361:
362: def generate_power_func(n):
363:     print "id(n): %X" % id(n)
364:     def nth_power(x):
365:         return x**n
366:     print "id(nth_power): %X" % id(nth_power)
367:     return nth_power
368:
369: >>> raised_to_4 = generate_power_func(4)
370: id(n): CCF7DC
371: id(nth_power): C46630
372: >>> repr(raised_to_4)
373: '<function nth_power at 0x00C46630>'
374:
375: def generate_power_func(n):
376:     print "id(n): %X" % id(n)
377:     def nth_power(x):
378:         return x**n
379:     print "id(nth_power): %X" % id(nth_power)
380:     return nth_power
381:
382: >>> raised_to_4 = generate_power_func(4)
383: id(n): CCF7DC
384: id(nth_power): C46630
385: >>> repr(raised_to_4)
386: '<function nth_power at 0x00C46630>'
387:
388: maXbox4 actual Version
389: Total of Function Calls: 35771
390: SHA1: 4.7.6.10 30bace36b037686509bbbee256e22daa52b3df70
391: CRC32: 500F69DD: 31.8 MB (33,400,088 bytes)
392: ZIP maxbox4.zip SHA1: 9DA15BFD72471108FCF746669C6AFD96E9A0E54C
393:
```