

Codeblock class: shebang

<http://www.google.com/search?q=shebang+line>

runs:

> <fname>.shebang {im_opt} <fname>.{im_fmt}

class->cmd

shebang -> shebang

Metadata options

imagine.im_out: img, fcb

imagine.shebang.im_out: img, fcb

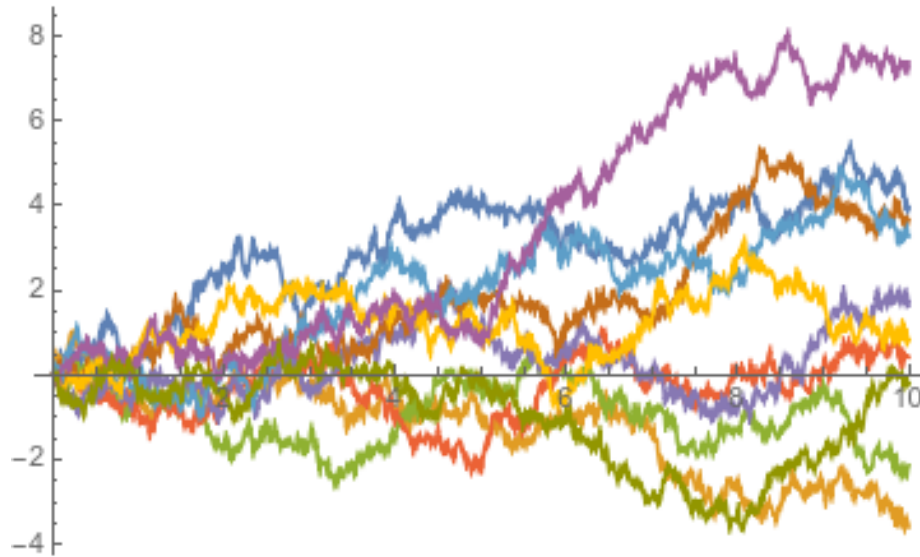
Notes

- `sudo -H bash ~/installs/wolfram/WolframEngine_12.0.1_LINUX.sh`
- create account on <https://account.wolfram.com> & get a free license
- first invoke `wolframscript` to activate the wolfram kernel & set license
- use `$ScriptCommandLine[[2]]` to pickup the destination filename

Examples

Examples from *wolfram cloud*

WienerProcess (png)

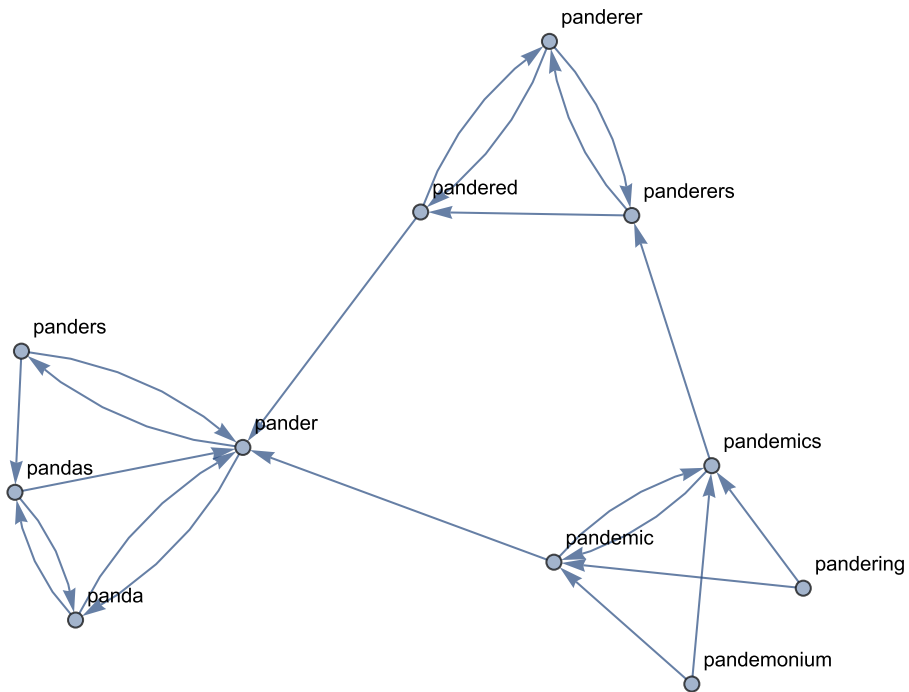


```
```shebang
#!/usr/bin/env wolframscript
Export[$ScriptCommandLine[[2]],
ListLinePlot[RandomFunction[WienerProcess[],{0,10,0.01},10]]
]
```
```

Notes:

- `png` is imagine's default output format

Nearby words

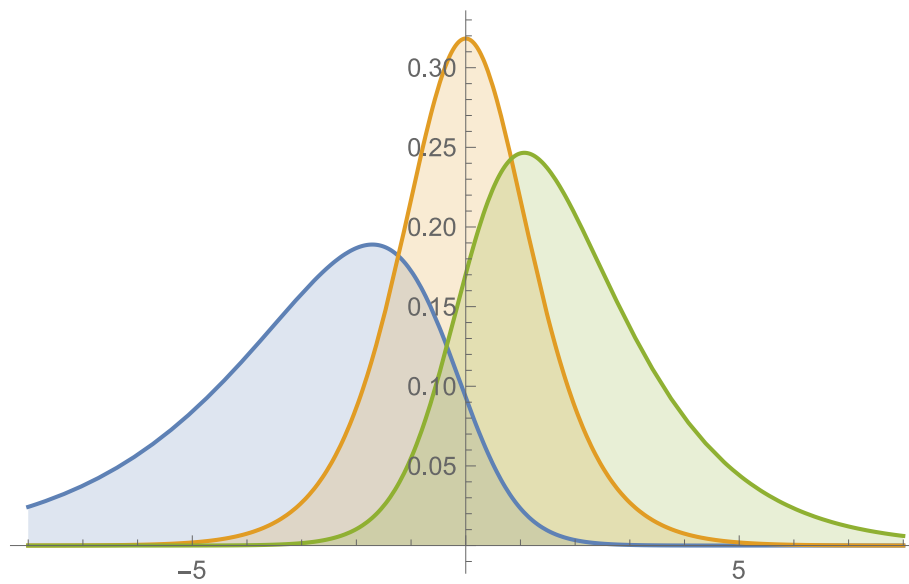


```

```{.shebang im_fmt="svg"}
#!/usr/bin/env wolframscript
words = DictionaryLookup["pand*"];
g = Graph[Flatten[Map[(Thread[# \[DirectedEdge]
 DeleteCases[Nearest[words, #, 3], #]]) &, words]],
 VertexLabels -> "Name", ImageSize -> 450
];
Export[$ScriptCommandLine[[2]], GraphPlot[g]]
```

```

Meixner (svg)

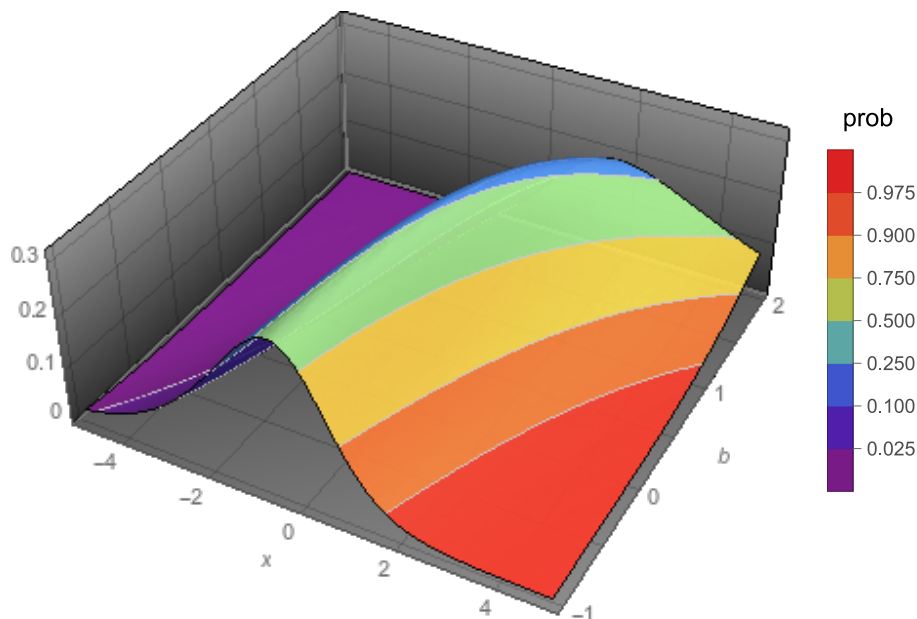


```

```{.shebang im_fmt="svg"}
#!/usr/bin/env wolframscript
Export[$ScriptCommandLine[[2]],
Plot[Table[
PDF[MeixnerDistribution[2, b, 0, 1], x], {b, {-2, 0, 1.5}}] //
Evaluate, {x, -8, 8}, Exclusions -> None, Filling -> Axis]
]
```

```

Meixner again



```

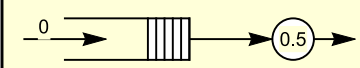
```{.shebang im_fmt="svg"}
#!/usr/bin/env wolframscript

dist = MeixnerDistribution[2, b, 0, 1];
cdf = Function[{x, b}, Evaluate[CDF[dist, x]]];
ql = {0.025, 0.10, 0.25, 0.5, 0.75, 0.90, 0.975};
cl = Table[ColorData["Rainbow"][q], {q, Join[{0.0}, ql]}];

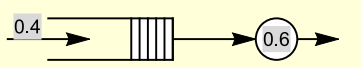
Export[$ScriptCommandLine[[2]],
Legended[Plot3D[PDF[dist, x], {x, -5, 5}, {b, -1, 2},
PlotTheme -> "Marketing", MeshFunctions -> {cdf}, Mesh -> {ql},
MeshStyle -> GrayLevel[0.8], MeshShading -> cl,
AxesLabel -> Automatic, BaseStyle -> Opacity[0.9], ImageSize -> 400,
PlotRange -> All, Exclusions -> None, PlotPoints -> 50],
BarLegend["Rainbow", ql, LegendLabel -> "prob"]]
]
```

```

Queueing properties

|  | |
|---|---------|
| Basic Properties | |
| NetworkType | Closed |
| NodeCount | 4 |
| SelectedNode | 2 |
| Performance Measures | |
| MeanSystemSize | 3.27944 |
| MeanSystemTime | 7.57403 |
| MeanQueueSize | 2.41347 |
| MeanQueueTime | 5.57403 |

{ , }

|  | |
|--|-----------------|
| Basic Properties | |
| DataSource | QueueingNetwork |
| NodeCount | 4 |
| SelectedNode | 2 |
| Performance Measures | |
| MeanSystemSize | 3.08814 |
| MeanSystemTime | 7.1314 |
| MeanQueueSize | 2.23695 |
| MeanQueueTime | 5.16575 |

```

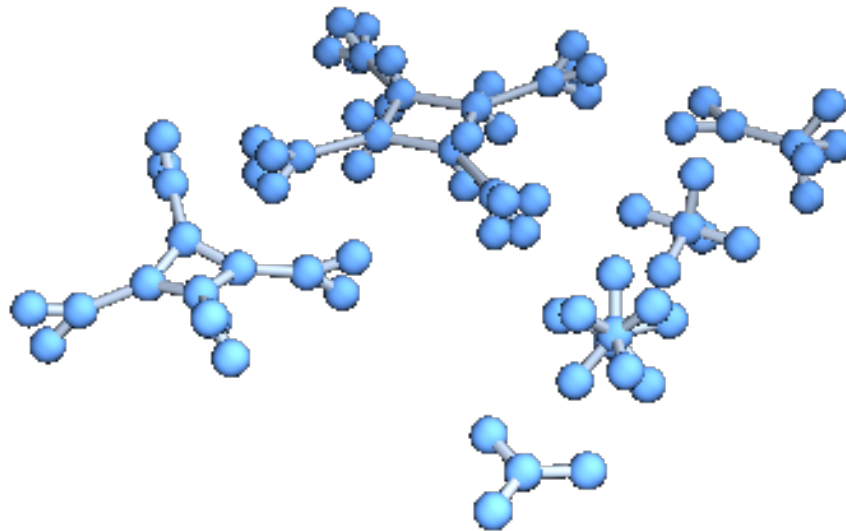
```{.shebang im_fmt="svg"}
#!/usr/bin/env wolframscript
\[Gamma] = {0, 0, 0, 0};
r = {{0, 0.5, 0.3, 0.2}, {1, 0, 0, 0}, {1, 0, 0, 0}, {1, 0, 0, 0}};
\[Mu] = {1, 0.5, 1, 2}; c = {1, 1, 1, 1}; k = 7;

\[ScriptCapitalN] = QueueingNetworkProcess\[Gamma], r, \[Mu], c, k];

data = RandomFunction\[ScriptCapitalN], {0, 10^4}];
Export[$ScriptCommandLine[[2]],
{QueueProperties[{\[ScriptCapitalN], 2}], QueueProperties[{data, 2}]}
]
```

```

3D plot



```
```{.shebang im_fmt="svg"}
#!/usr/bin/env wolframscript
Export[$ScriptCommandLine[[2]],
 GraphPlot3D[
 SimpleGraph[Table[i \[UndirectedEdge] Mod[i^2, 75], {i, 75}]],
 EdgeShapeFunction -> ({Cylinder[#1, 0.1]} &),
 VertexShapeFunction -> ({Sphere[#, 0.3]} &)]
]
```
```

Documentation

wolfram -h

OPTIONS:

| | |
|-------------------------|---|
| -h, -help | Print help text. |
| -c, -code WL | Give Wolfram Language code to execute. |
| -f, -file PATH | Give a file containing Wolfram Language code to execute. |
| -a, -api URL UUID | Use an API at the specified URL, or coming from a cloud or local object with the specified UUID. Provide arguments in the form 'key=value' following '-args'. |
| -fun, -function WL | Use a function whose arguments are the strings given using '-args' interpreted as the types given using '-signature' |
| -o, -cloud [CLOUDBASE] | Execute code in the cloud, using the specified cloud base. By default, cloud base is <code>https://wolframcloud.com</code> |
| -l, -local [KERNELPATH] | Execute code locally, using the specified path to the Wolfram Engine kernel. By default, KernelPath uses the most recent version of the Wolfram Language found on the local system. |
| --, -args ARGS... | Used with '-api' or '-function' to provide arguments. |
| -s, -signature TYPE... | Used with '-function' and '-args' to specify interpreter types for provided arguments. |
| -v, -verbose | Print additional information during execution. |
| -activate [KEY] | Activate the Wolfram Engine through the cloud or with a key. |
| -authenticate [ID PASS] | Authenticate with the cloud, specifying a particular Wolfram ID and password, and prompting if they are not given. Different authentication can be specified for different clouds. |
| -disconnect | Disconnect from the cloud, removing authentication information. |
| -configure [KEY=VAL...] | Configure WolframScript by specifying values for particular configuration variable keys. If no keys are given, this prints the current configuration. |
| -version | Print version of WolframScript. |
| -username USERNAME | Give username for authentication. |
| -password PASSWORD | Give password for authentication. |
| -permissionskey KEY | Give a permissions key used to authorize access to a cloud deployed API. |

| | |
|---------------------------------------|--|
| <code>-k, -noverifypeer</code> | Disable peer certificate verification when interacting with the cloud. |
| <code>-timeout SECONDS [VALUE]</code> | Specify the number of seconds to allow for execution. Return value if time is exceeded. |
| <code>-charset ENCODING</code> | Use encoding for output. Encodings can be None to output raw bytes, or any entry in <code>\$CharacterEncodings</code> except "Unicode". The default is to infer the value from the terminal's language settings. |
| <code>-format TYPE</code> | Specify the format in which to give output. Any format understood by <code>Export</code> can be used. |
| <code>-print [all]</code> | When running a script, print the result from executing the last line of the script, or each line if all is given. |
| <code>-linewise</code> | Execute code on each line of standard input that is read. |
| <code>-script ARGS...</code> | Counterpart to <code>wolfram -script</code> , additionally sets up <code>\$ScriptCommandLine</code> . |

man page

wolframscript(1)

General Commands Manual

wolframscript(1)

NAME

wolframscript - command-line script interpreter for Wolfram Language

SYNOPSIS

```
wolframscript -code code [-cloud [cloudbase]] [-local [kernelpath]
[arg1 ...]
```

```
wolframscript -file file|url [-cloud cloudbase] [-local [kernel-
path]] [arg1 ...]
```

```
wolframscript -api url|uuid|file [-cloud [cloudbase]] [-local
[kernelpath]] [-args key=value ...]
```

```
wolframscript -function code [-cloud [cloudbase]] [-local [-ker-
nelpath]] [-signature type ...] [-args values ...]
```

DESCRIPTION

wolframscript runs Wolfram Language code, functions, and deployed APIs, either locally or in the cloud, allowing input from standard input, command-line arguments, files, URLs, etc.

EXAMPLES

Code from Command Line

Evaluate the Wolfram Language code 2+2 on a local Wolfram Engine:

```
$ wolframscript -code 2+2
4
```

Evaluate the Wolfram Language code 2+2 in the Wolfram Cloud, prompting for authentication as needed:

```
$ wolframscript -cloud -code 2+2
4
```

Evaluate Wolfram Language code locally, escaping input for the shell:

```
$ wolframscript -code 'StringReverse["hello"]'
"olleh"
```

Evaluate code and put the results in a file:

```
$ wolframscript -code 'ExportString[Graphics3D[Sphere[ ]], \
  "PNG"]' > file.png
```

Code from File

Evaluate Wolfram Language code from a file, returning the last result generated:

```
$ wolframscript -file test.wl
12345
```

Take code from a local file, but run it in the cloud:

```
$ wolframscript -cloud -file test.wl
12345
```

Take code from a file in the cloud, but run it locally:

```
$ wolframscript -file http://wolfr.am/535sxfw4
12345
```

Script Files

A file set up to execute Wolfram Language code locally:

```
#!/wolframscript
Print[2+2]
```

Create a file to execute Wolfram Language code in the Wolfram Cloud:

```
#!/wolframscript -cloud
Print[2+2]
```

Create a file that uses a command-line argument:

```
#!/wolframscript
Print[ToExpression[$ScriptCommandLine[[1]]^2]
```

Create a file giving a function whose arguments come from the command line:

```
#!/wolframscript -function -signature City City
Print[GeoDistance[#1, #2]]&
```

Interactive Operation

Run Wolfram Language in an interactive REPL:

```
$ wolfrascript
WolframEngine 11.0.0 for Mac OS X x86 (64-bit)
Copyright 1988-2016 Wolfram Research, Inc.

In[1]:= 2+2

Out[1]= 4

In[2]:=
```

APIs

Run a cloud API:

```
$ wolfrascript -api https://wolfr.am/bNvKWq2U -args x=1 y=2
3
```

Get the code for an API from the cloud, but run the API locally:

```
$ wolfrascript -api https://wolfr.am/bNvKWq2U -local -args x=1 y=2
3
```

Additional Examples

Reverse the string on each line of an input file, writing the result to another file:

```
$ wolfrascript -function StringReverse -linewise < file1 > file2
```

OPTIONS

Code Options

- c, -code code
Give Wolfram Language code to execute.
- f, -file file
Give a file containing Wolfram Language code to execute.
- api url|uuid|file
Use an API at the specified URL, or coming from a cloud or local object with the specified UUID, or coming from the specified local file. Use arguments key=value
- fun, -function code [-s|signature type ...] [-args|-- value ...]
Use a function whose arguments are the strings value ..., interpreted as being of types type If no signature is given, all arguments are assumed to be strings. Signature

types can be any of `$InterpreterTypes`.

Execution Options

- `-o, -cloud [cloudbase]`
Execute code in the cloud, using the specified cloud base.
By default, cloudbase is `http://wolframcloud.com`.
- `-l, -local [kernelpath]`
Execute code locally, using the specified path to the Wolfram Engine kernel.
- `-format [type]`
Specify the format in which to give output.
- `-linewise`
Execute code on each line of standard input that is read.
- `-print [all]`
When running a script, print the result from executing the last line of the script, or each line if all is given.
- `-timeout seconds [value]`
Specify the number of seconds to allow for execution. Return value if the time is exceeded.
- `-v, -verbose`
Print additional information during execution.

Utility Options

- `-h, -help`
Print help text.
- `-auth, -authenticate [wolframid [password]] [-cloud cloudbase]`
Authenticate with the cloud, specifying a particular Wolfram ID and password, and prompting if they are not given. Different authentication can be specified for different clouds.
- `-config, -configure [key=value ...]`
Configure wolframscript by specifying values for particular configuration variables keys.
- `-disconnect [-cloud cloudbase]`
Disconnect from the cloud, removing authentication information.

DETAILS

Wolfram Language Scripts

All standard options can be used in `#!/wolframscript` scripts.

With `#!/wolframscript -function ...`, each argument to the function can be given on the script command line.

With `#!/wolframscript -api ...`, the parameters of the API can be given on the script command line in the form `-key value ...`.

The exit code from executing a script can be specified using `Exit[code]`.

Without `-print`, no output will be sent to `stdout` unless this is explicitly done using `Print[expr]`.

With the option `-print`, the result from the last line in the script is sent to `stdout`.

With the option `-print all`, results from each line in the script are sent to `stdout` when they are generated.

The `-linewise` option can be used to run the script multiple times, taking a single line of `stdin` as input each time.

Command-Line Input

Input given to a script on standard input can be accessed in Wolfram Language code using `$ScriptInputString`.

Arguments given on the command line can be accessed using `$ScriptCommandLine`.

Output Formatting

The default setting for `TotalWidth` is `Infinity`.

API Parameters

If an API supports extended parameters such as `x-url`, `x-format`, and `_timeout`, these can be given in `wolframscript -api`.

Code Location

In `wolframscript -api uuid`, `LocalObject["uuid"]` is used if it exists, otherwise `CloudObject["uuid"]`.

FILES

Configuration file:

WOLFRAM LANGUAGE VARIABLES

The following variables are set when wolframscript begins execution.

`$CommandLine`

A list of strings giving the complete command line used.

`$ScriptCommandLine`

A list of command-line arguments intended for the script being run. These come after options given with `-option`.

ENVIRONMENT VARIABLES

`WOLFRAMSCRIPT_AUTHENTICATIONPATH`

The path to files storing authentication information.

`WOLFRAMSCRIPT_CONFIGURATIONPATH`

The path to files storing persistent configuration information.

`WOLFRAMSCRIPT_CLOUDBASE`

The default cloud base to use in wolframscript.

`WOLFRAMSCRIPT_KERNELPATH`

The path to the default local Wolfram Engine kernel executable.

wolframscript(1)