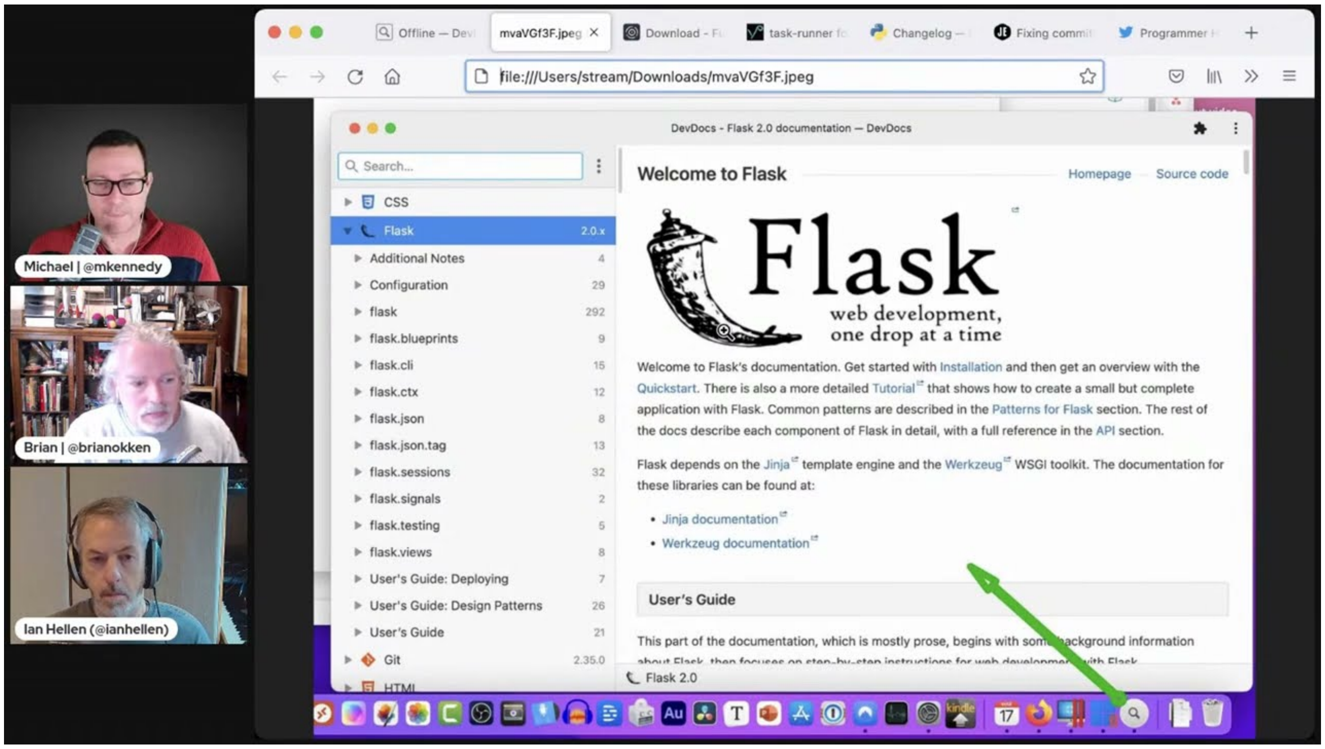
Episode #276: Tracking cyber intruders with Jupyter and Python

pythonbytes.fm/episodes/show/276/tracking-cyber-intruders-with-jupyter-and-python


Michael | @mkennedy
Brian | @brianokken
Ian Hellen (@ianhellen)
file:///Users/stream/Downloads/mvaVGf3F.jpeg
DevDocs - Flask 2.0 documentation — DevDocs
Welcome to Flask
Homepage
Source code
Flask
web development, one drop at a time
User's Guide
Flask 2.0

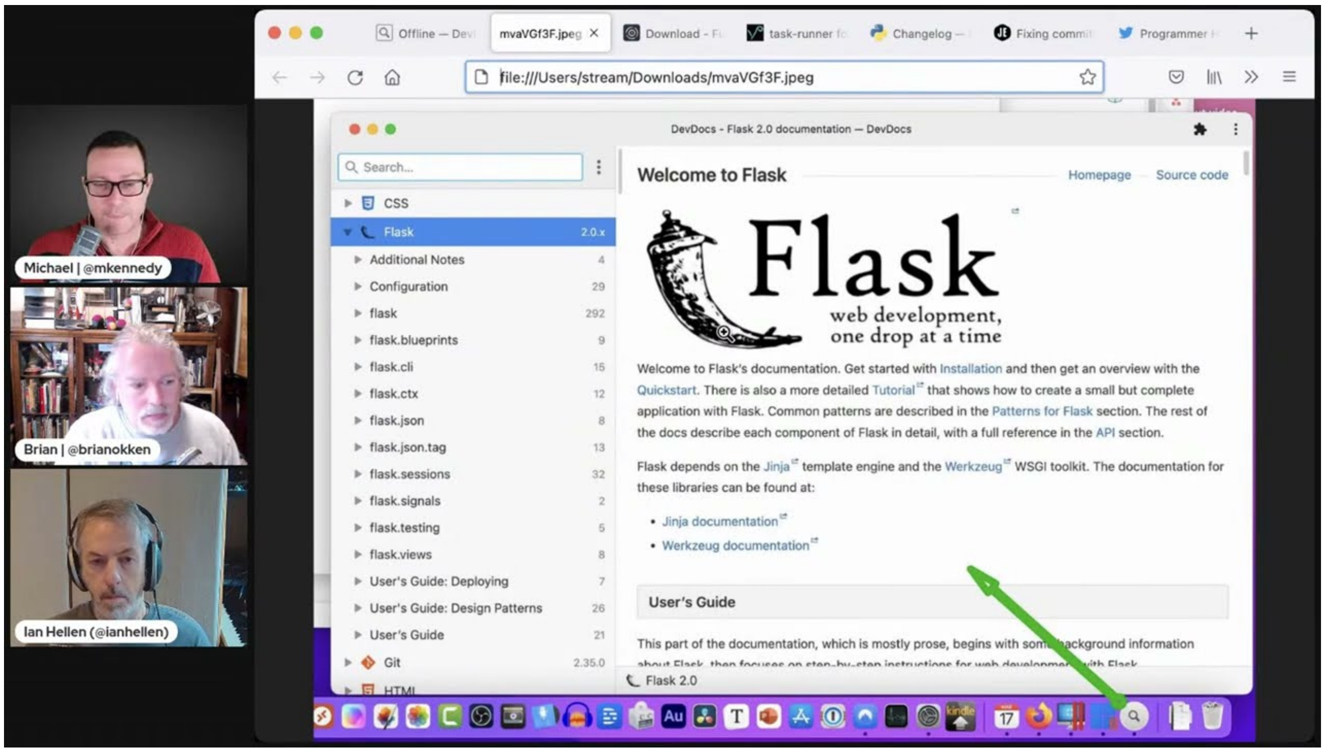
Welcome to Flask's documentation. Get started with Installation and then get an overview with the Quickstart. There is also a more detailed Tutorial that shows how to create a small but complete application with Flask. Common patterns are described in the Patterns for Flask section. The rest of the docs describe each component of Flask in detail, with a full reference in the API section.

Flask depends on the Jinja template engine and the Werkzeug WSGI toolkit. The documentation for these libraries can be found at:

- Jinja documentation
- Werkzeug documentation

Published Wed, Mar 23, 2022, recorded Tue, Mar 22, 2022.

Watch the live stream:



Welcome to Flask's documentation. Get started with Installation and then get an overview with the Quickstart. There is also a more detailed Tutorial that shows how to create a small but complete application with Flask. Common patterns are described in the Patterns for Flask section. The rest of the docs describe each component of Flask in detail, with a full reference in the API section.

Flask depends on the Jinja template engine and the Werkzeug WSGI toolkit. The documentation for these libraries can be found at:

- Jinja documentation
- Werkzeug documentation

User's Guide

This part of the documentation, which is mostly prose, begins with some background information

Play on YouTube

About the show

Sponsored by FusionAuth: pythonbytes.fm/fusionauth

Special guest: **Ian Hellen**

Brian #1: gensim.parsing.preprocessing

- Problem I'm working on
  Turn a blog title into a possible url
    - example: "Twisted and Testing Event Driven / Asynchronous Applications - Glyph"
    - would like, perhaps: "twisted-testing-event-driven-asynchrounous-applications"
- Sub-problem: remove stop words ← this is the hard part
- I started with an article called Removing Stop Words from Strings in Python
  - It covered how to do this with NLTK, Gensim, and SpaCy
  - I was most successful with `remove_stopwords()` from Gensim
    - `from gensim.parsing.preprocessing import remove_stopwords`
    - It's part of a `gensim.parsing.preprocessing` package

- I wonder what's all in there?
    - a treasure trove
    - `gensim.parsing.preprocessing.preprocess_string` is one
    - this function applies filters to a string, with the defaults almost being just what I want:
        - strip_tags()
        - strip_punctuation()
        - strip_multiple_whitespaces()
        - strip_numeric()
        - remove_stopwords()
        - strip_short()
        - stem_text() ← I think I want everything except this

          this one turns "Twisted" into "Twist", not good.
- There's lots of other text processing goodies in there also.
- Oh, yeah, and Gensim is also cool.

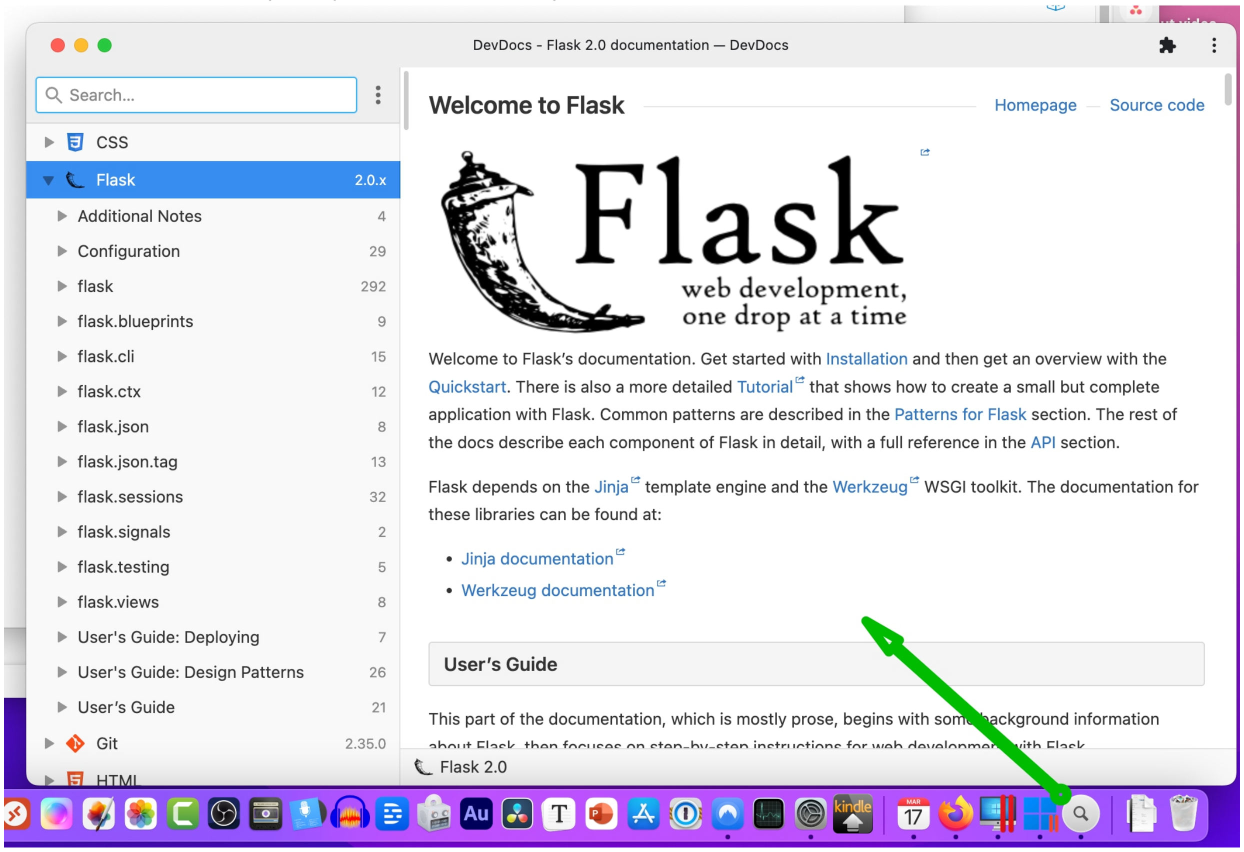
    topic modeling for training semantic NLP models
- So, I think I found a really big hammer for my little problem.

    But I'm good with that

Michael #2: DevDocs

- via Loic Thomson
- Gather and search a bunch of technology docs together at once
- For example: Python + Flask + JavaScript + Vue + CSS
- Has an offline mode for laptops / tablets

- Installs as a PWA (sadly not on Firefox)



DevDocs - Flask 2.0 documentation — DevDocs

Search...

- CSS
- Flask 2.0.x
  - Additional Notes 4
  - Configuration 29
  - flask 292
  - flask.blueprints 9
  - flask.cli 15
  - flask.ctx 12
  - flask.json 8
  - flask.json.tag 13
  - flask.sessions 32
  - flask.signals 2
  - flask.testing 5
  - flask.views 8
  - User's Guide: Deploying 7
  - User's Guide: Design Patterns 26
  - User's Guide 21
- Git 2.35.0
- HTML

Welcome to Flask — Homepage — Source code

Welcome to Flask's documentation. Get started with Installation and then get an overview with the Quickstart. There is also a more detailed Tutorial that shows how to create a small but complete application with Flask. Common patterns are described in the Patterns for Flask section. The rest of the docs describe each component of Flask in detail, with a full reference in the API section.

Flask depends on the Jinja template engine and the Werkzeug WSGI toolkit. The documentation for these libraries can be found at:

- Jinja documentation
- Werkzeug documentation

User's Guide

This part of the documentation, which is mostly prose, begins with some background information about Flask, then focuses on step-by-step instructions for web development with Flask.

Flask 2.0

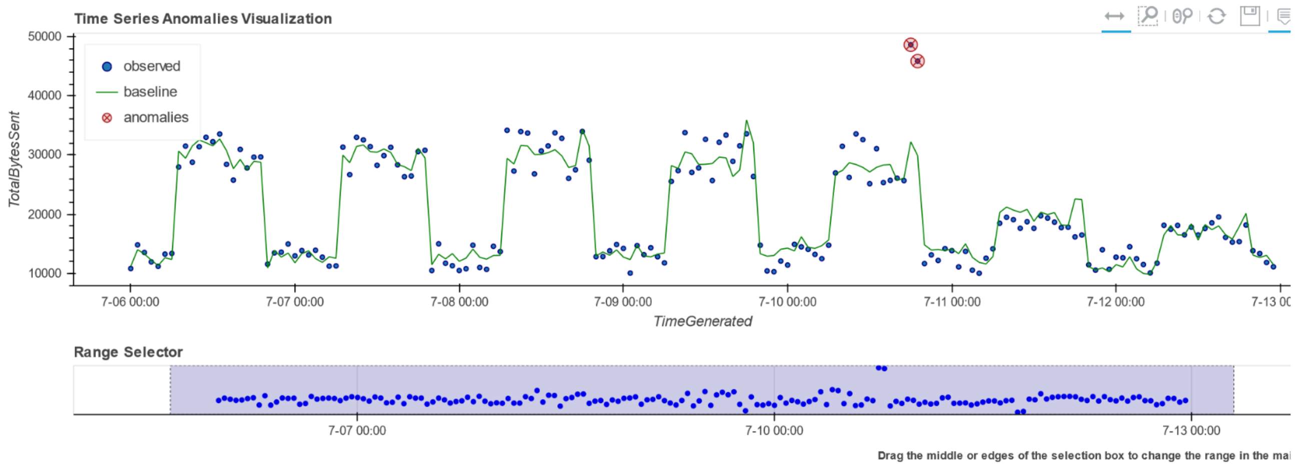
Ian #3: MSTICPy

- MSTICPy is toolset for CyberSecurity investigations and hunting in Jupyter notebooks.
- What is CyberSec hunting/investigating? - responding to security alerts and threat intelligence reports, trawling through security logs from cloud services and hosts to determine if it's a real threat or not.
- Why Jupyter notebooks?
  - SOC (Security Ops Center) tools can be excellent but all have limitations
  - You can get data from anywhere
  - Use custom analysis and visualizations
  - Control the workflow…. workflow is repeatable
- Open source pkg - created originally to support MS Sentinel Notebooks but now supports lots of providers. When I start this 3+ yrs ago I thought a lot this would be in PyPI - but no 😞
- MSTICPy has 4 main functional areas:
  - Data querying - import log data (Sentinel, Splunk, MS Defender, others…working on Elastic Search)
  - Enrichment - is this IP Address or domain known to be malicious?
  - Analysis - extract more info from data, identify anomalies (simple example - spike in logon failures)
  - Visualization - more specialized than traditional graphs - timelines, process trees.
- All components use pandas, Bokeh for visualizations

- Current focus on usability, discovery of functionality and being able to chain
- Always looking for collaborators and contributors - code, docs, queries, critiques
- https://github.com/microsoft/msticpy
- https://msticpy.readthedocs.io/

```
from msticpy.nbtools.timeseries import display_timeseries_anomolies
from msticpy.analysis.timeseries import timeseries_anomalies_stl

# Conduct our timeseries analysis
ts_analysis = timeseries_anomalies_stl(ob_bytes_per_hour)
# Visualize the timeseries and any anomalies
display_timeseries_anomolies(data=ts_analysis, y= 'TotalBytesSent');

md("We can see two clearly anomalous data points representing unusual outbound traffic.<hr>", "bold")
```

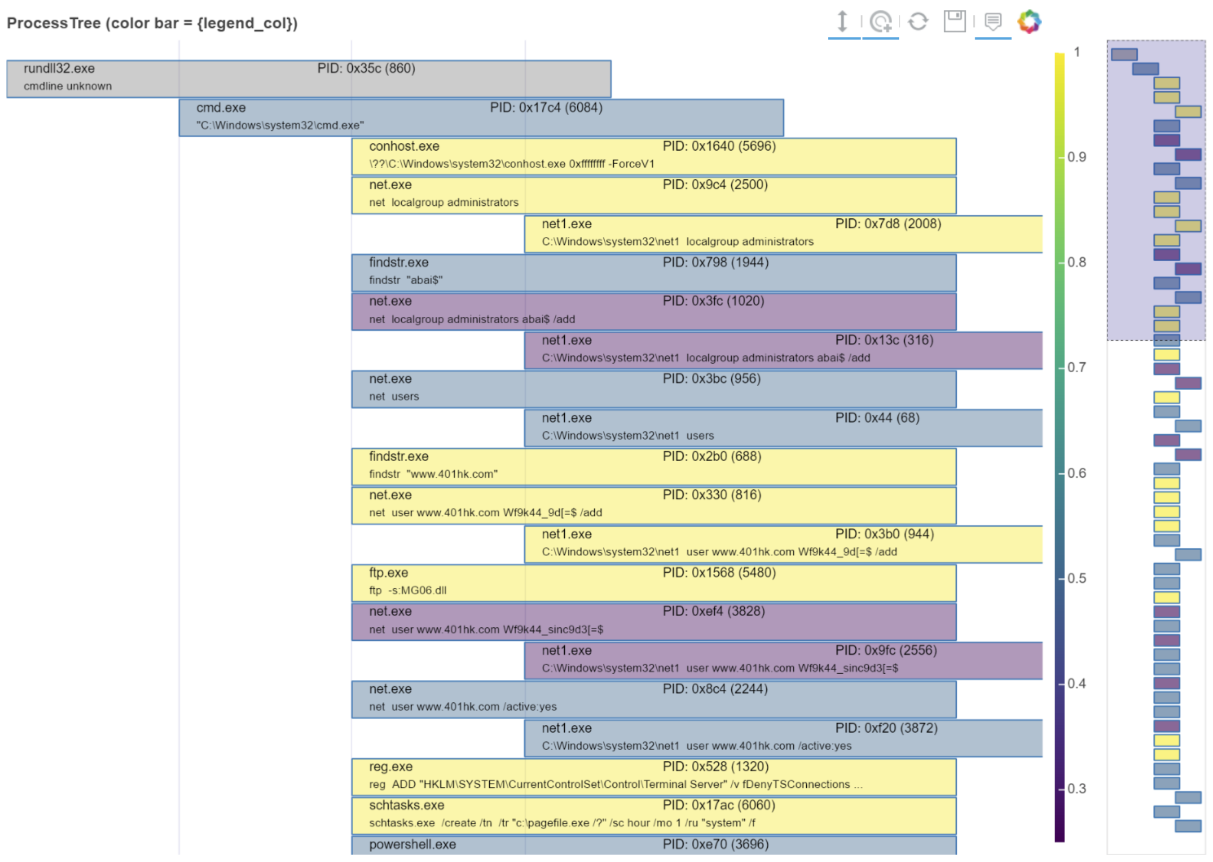
Loading BokehJS ...



```
[76]:  logon_rslt.process_tree(account="MSTICAlertsWin1\ian")
```

Progress: 100%

BokehJS 2.2.2 successfully loaded.



Process Tree (color bar = {legend_col})

```
rundll32.exe                                   PID: 0x35c (860)
cmdline unknown
    cmd.exe                                    PID: 0x17c4 (6084)
    "C:\Windows\system32\cmd.exe"
        conhost.exe                            PID: 0x1640 (5696)
        \??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
        net.exe                                PID: 0x9c4 (2500)
        net  localgroup administrators
            net1.exe                           PID: 0x7d8 (2008)
            C:\Windows\system32\net1  localgroup administrators
        findstr.exe                            PID: 0x798 (1944)
        findstr  "abai$"
        net.exe                                PID: 0x3fc (1020)
        net  localgroup administrators abai$ /add
            net1.exe                           PID: 0x13c (316)
            C:\Windows\system32\net1  localgroup administrators abai$ /add
        net.exe                                PID: 0x3bc (956)
        net  users
            net1.exe                           PID: 0x44 (68)
            C:\Windows\system32\net1  users
        findstr.exe                            PID: 0x2b0 (688)
        findstr  "www.401hk.com"
        net.exe                                PID: 0x330 (816)
        net  user www.401hk.com Wf9k44_9d[=$ /add
            net1.exe                           PID: 0x3b0 (944)
            C:\Windows\system32\net1  user www.401hk.com Wf9k44_9d[=$ /add
        ftp.exe                                PID: 0x1568 (5480)
        ftp  -s:MG06.dll
        net.exe                                PID: 0xef4 (3828)
        net  user www.401hk.com Wf9k44_sinc9d3[=$
            net1.exe                           PID: 0x9fc (2556)
            C:\Windows\system32\net1  user www.401hk.com Wf9k44_sinc9d3[=$
        net.exe                                PID: 0x8c4 (2244)
        net  user www.401hk.com /active:yes
            net1.exe                           PID: 0xf20 (3872)
            C:\Windows\system32\net1  user www.401hk.com /active:yes
        reg.exe                                PID: 0x528 (1320)
        reg  ADD "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections ...
        schtasks.exe                           PID: 0x17ac (6060)
        schtasks.exe  /create /tn  /tr "c:\pagefile.exe /?" /sc hour /mo 1 /ru "system" /f
        powershell.exe                         PID: 0xe70 (3696)
```

38.75.137.9

Type: 'ipv4', Provider: OTX, severity: high

Details

OTX	
pulse_count	4
names	['Underminer.EK - Exploit Kit IOC Feed', '', 'Underminer.EK - Exploit Kit IOC Feed', 'Underminer EK']
tags	[['Underminer.EK'], ['Underminer.EK'], ['Underminer.EK'], []]
references	[[], [], [], ['https://blog.malwarebytes.com/threat-analysis/2019/07/exploit-kits-summer-2019-review/']]

Reference:

https://otx.alienvault.com/api/v1/indicators/IPv4/38.75.137.9/general

---

Brian #4: The Right Way To Compare Floats in Python

- David Amos

- Definitely an easier read than the classic What Every Computer Scientist Should Know About Floating-Point Arithmetic
  What many of us remember
    - floating point numbers aren't exact due to representation limitations and rounding error,
    - errors can accumulate
    - comparison is tricky
- Be careful when comparing floating point numbers, even simple comparisons, like: >>> 0.1 + 0.2 == 0.3 False >>> 0.1 + 0.2 <= 0.3 False
- David has a short but nice introduction to the problems of representation and rounding.
- Three reasons for rounding
  - more significant digits than floating point allows
  - irrational numbers
  - rational but non-terminating
- So how do you compare:
  - `math.isclose()`
    be aware of `rel_tol` and `abs_tol` and when to use each.
  - `numpy.allclose()` , returns a boolean comparing two arrays
  - `numpy.isclose()` , returns an array of booleans
  - `pytest.approx()` , used a bit differently
    - `0.1 + 0.2 == pytest.approx(0.3)`
    - Also allows `rel` and `abs` comparisons
- Discussion of `Decimal` and `Fraction` types
  And the memory and speed hit you take on when using them.

Michael #5: Pypyr

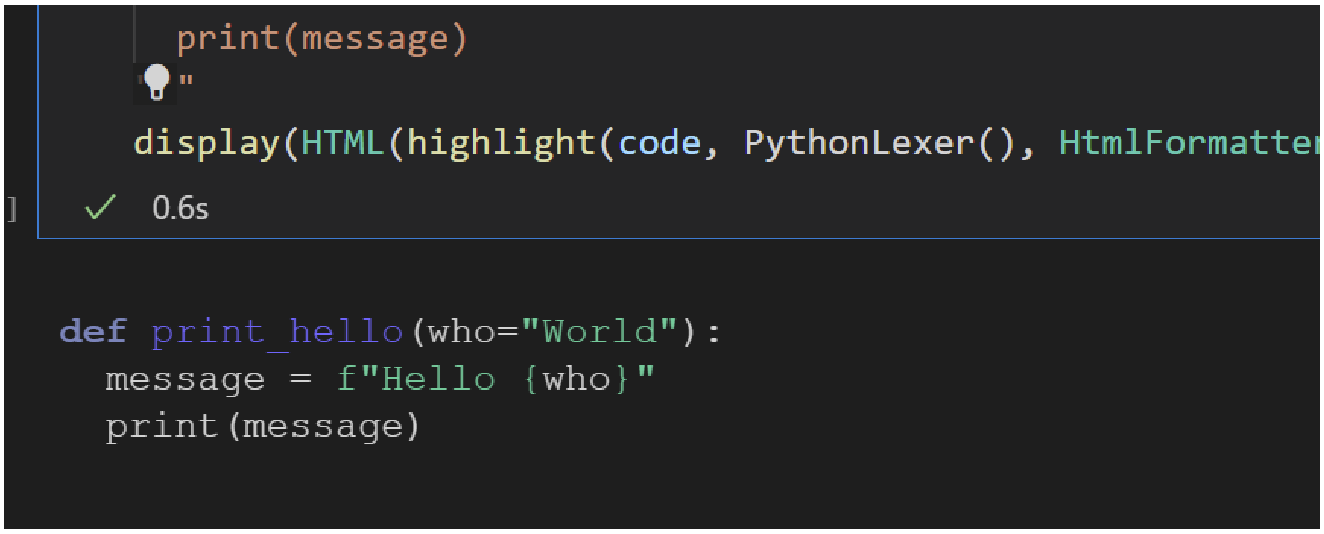
- Task runner for automation pipelines
- For when your shell scripts get out of hand. Less tricky than makefile.
- Script sequential task workflow steps in yaml
- Conditional execution, loops, error handling & retries
- Have a look at the getting started.

Ian #6: Pygments

- Python package that's useful for anyone who wants to display code
  - Jupyter notebook Markdown and GitHub markdown let you display code with syntax highlighting. (Jupyter uses Pygments behind the scenes to do this.)
  - There are tools that convert code to image format (PNG, JPG, etc) but you lose the ability to copy/paste the code
- Pygments can intelligently render syntax-highlighted code to HTML (and other formats)
- Applications:
  - Documentation (used by Sphinx/ReadtheDocs) - render code to HTML + CSS
  - Displaying code snippets dynamically in readable form

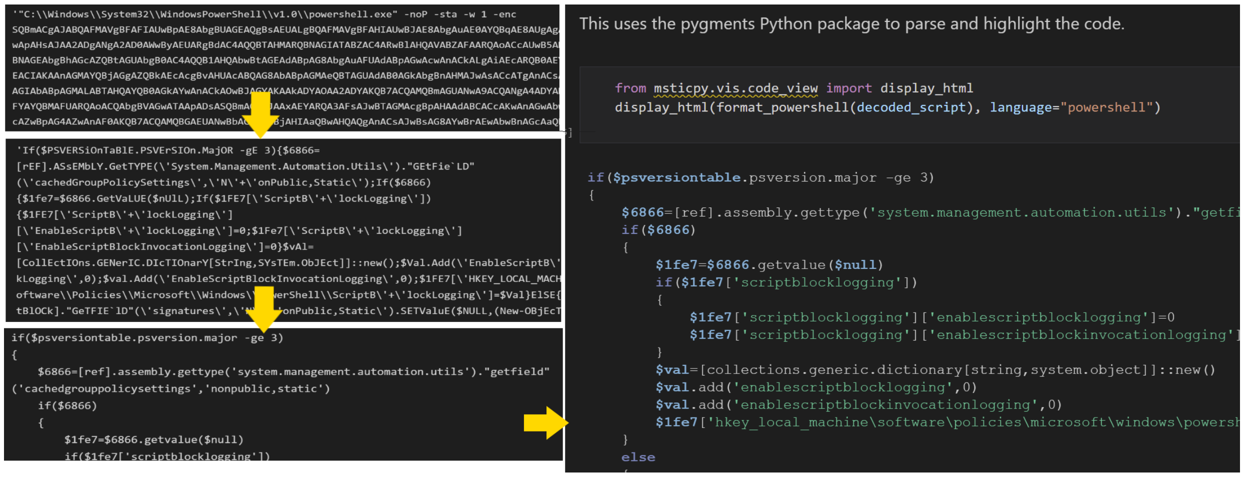
- Lots (maybe 100s) of code lexers - Python (code, traceback), Bash, C, JS, CSS, HTML, also config and data formats like TOML, JSON, XML
- Easy to use - 3 lines of code - example:

```
from IPython.display import display, HTML
from pygments import highlight
from pygments.lexers import PythonLexer
from pygments.formatters import HtmlFormatter

code = """
def print_hello(who="World"):
    message = f"Hello {who}"
    print(message)
"""
display(HTML(
    highlight(code, PythonLexer(), HtmlFormatter(full=True, nobackground=True))
))
# use HtmlFormatter(style="stata-dark", full=True, nobackground=True)
# for dark themes
```



```
    print(message)
"
display(HTML(highlight(code, PythonLexer(), HtmlFormatte
✓ 0.6s

def print_hello(who="World"):
  message = f"Hello {who}"
  print(message)
```

- Output to HTML, Latex, image formats.
- We use it in MSTICPy for displaying scripts used in attacks. Example:



```
'"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe" -noP -sta -w 1 -enc
SQBmACgAJABQAFMAVgBFAFIAUwBpAE8AbgBUAGEAQgBsAEUALgBQAFMAVgBFAHIAUwBJAE8AbgAuAE0AYQBqAE8AUgAg
wApAHsAJAA2ADgANgA2AD0AWwByAEUARgBdAC4AQQBTAHMARQBNAGIATABZAC4ARwBlAHQAVABZAFAARQAoACcAUwB5A
BNAGEAbgBhAGcAZQBtAGUAbgB0AC4AQQB1AHQAbwBtAGEAdABpAG8AbgAuAFUAdABpAGwAcwAnACkALgAiAEcARQB0AE
EACIAKAAnAGMAYQBjAGgAZQBkAEcAcgBvAHUAcABQAG8AbABpAGMAeQBTAGUAdAB0AGkAbgBnAHMAJwAsACcATgAnACs
AGIAbABpAGMALABTAHQAYQB0AGkAYwAnACkAOwBJAGYAKAAkADYAOAA2ADYAKQB7ACQAMQBmAGUANwA9ACQANgA4ADYA
FYAYQBMAFUARQAoACQAbgBVAGwATAApADsASQBmA[illegible]JAAxAEYARQA3AFsAJwBTAGMAcgBpAHAAdABCACcAKwAnAGwAb
cAZwBpAG4AZwAnAF0AKQB7ACQAMQBGAEUANwBbA[illegible]BjAHIAaQBwAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQ
```

```
'If($PSVERSiOnTaBlE.PSVErSIOn.MajOR -gE 3){$6866=
[rEF].ASsEMbLY.GetTYPE(\'System.Management.Automation.Utils\')."GEtFie`LD"
(\'cachedGroupPolicySettings\',\'N\'+\'onPublic,Static\');If($6866)
{$1fe7=$6866.GetVaLUE($nUlL);If($1FE7[\'ScriptB\'+\'lockLogging\'])
{$1FE7[\'ScriptB\'+\'lockLogging\']
[\'EnableScriptB\'+\'lockLogging\']=0;$1Fe7[\'ScriptB\'+\'lockLogging\']
[\'EnableScriptBlockInvocationLogging\']=0}$vAl=
[CollEctIOns.GENerIC.DIcTIOnarY[StrIng,SYsTEm.ObJEct]]::new();$Val.Add(\'EnableScriptB\'
kLogging\',0);$val.Add(\'EnableScriptBlockInvocationLogging\',0);$1FE7[\'HKEY_LOCAL_MACH
oftware\\Policies\\Microsoft\\Windows\\[illegible]werShell\\ScriptB\'+\'lockLogging\']=$Val}ElSE{
tBlOCk]."GeTFIE`lD"(\'signatures\',\'[illegible]onPublic,Static\').SETValuE($NULL,(New-OBjEcT
```

```
if($psversiontable.psversion.major -ge 3)
{
    $6866=[ref].assembly.gettype('system.management.automation.utils')."getfield"
('cachedgrouppolicysettings','nonpublic,static')
    if($6866)
    {
        $1fe7=$6866.getvalue($null)
        if($1fe7['scriptblocklogging'])
```

This uses the pygments Python package to parse and highlight the code.

```
from msticpy.vis.code_view import display_html
display_html(format_powershell(decoded_script), language="powershell")
```

```
if($psversiontable.psversion.major -ge 3)
{
    $6866=[ref].assembly.gettype('system.management.automation.utils')."getfi
    if($6866)
    {
        $1fe7=$6866.getvalue($null)
        if($1fe7['scriptblocklogging'])
        {
            $1fe7['scriptblocklogging']['enablescriptblocklogging']=0
            $1fe7['scriptblocklogging']['enablescriptblockinvocationlogging']
        }
        $val=[collections.generic.dictionary[string,system.object]]::new()
        $val.add('enablescriptblocklogging',0)
        $val.add('enablescriptblockinvocationlogging',0)
        $1fe7['hkey_local_machine\software\policies\microsoft\windows\powersh
    }
    else
```

Extras

Brian:

smart-open

- one of the 3 Gensim dependencies
- It’s for streaming large files, from really anywhere, and looks just like Python’s `open()`.

Michael:

Joke: What’s your secret?