

BlackHat Asia 2024 / rocky@gnu.org

uncompyle6 and **decompyle3**: How to Read and Write a High-Level Bytecode Decompiler

BlackHat Asia, 2024

Rocky Bernstein rocky@gnu.org




Slide text: <https://rocky.github.io/blackhat-asia-2024-additional/all-notes-print>

Survey




Survey

- How many people have used , , or ?

Survey

- How many people have used , , or ?
- How many people have used the above to decompile *Python* bytecode?

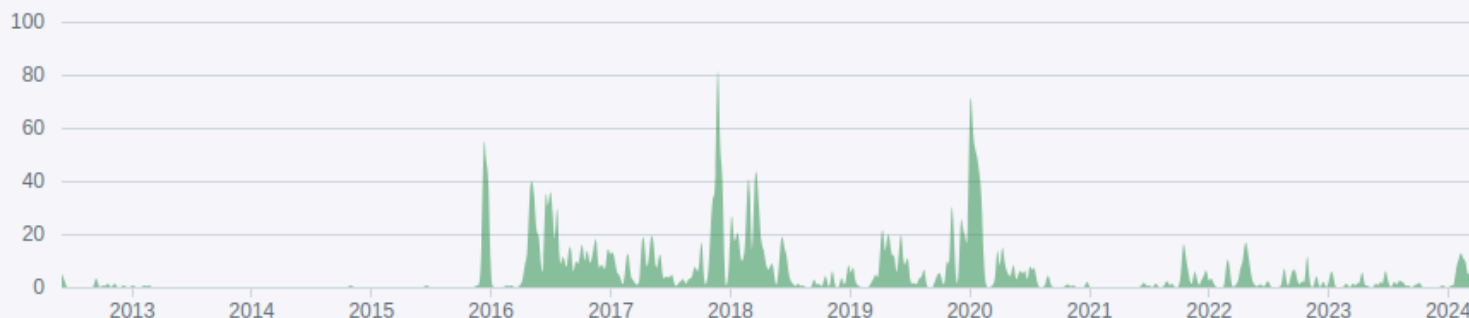
Survey

- How many people have used , , or ?
- How many people have used the above to decompile *Python* bytecode?
- How many people have used `uncompyle6`, or `decompyle3`?

Github commit statistics for uncompile6

I am the current maintainer and developer of uncompile6, and decompile3.

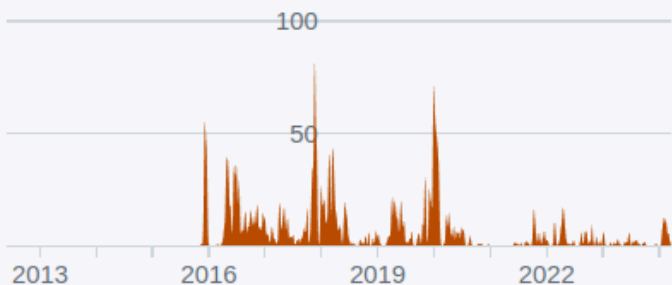
Contributions to master, excluding merge commits



rocky

3,040 commits 419,565 ++ 373,414 --

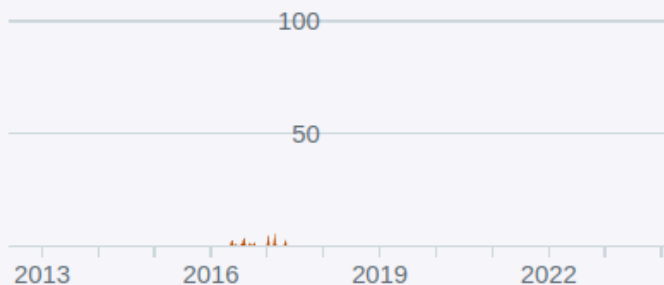
#1



moagstar

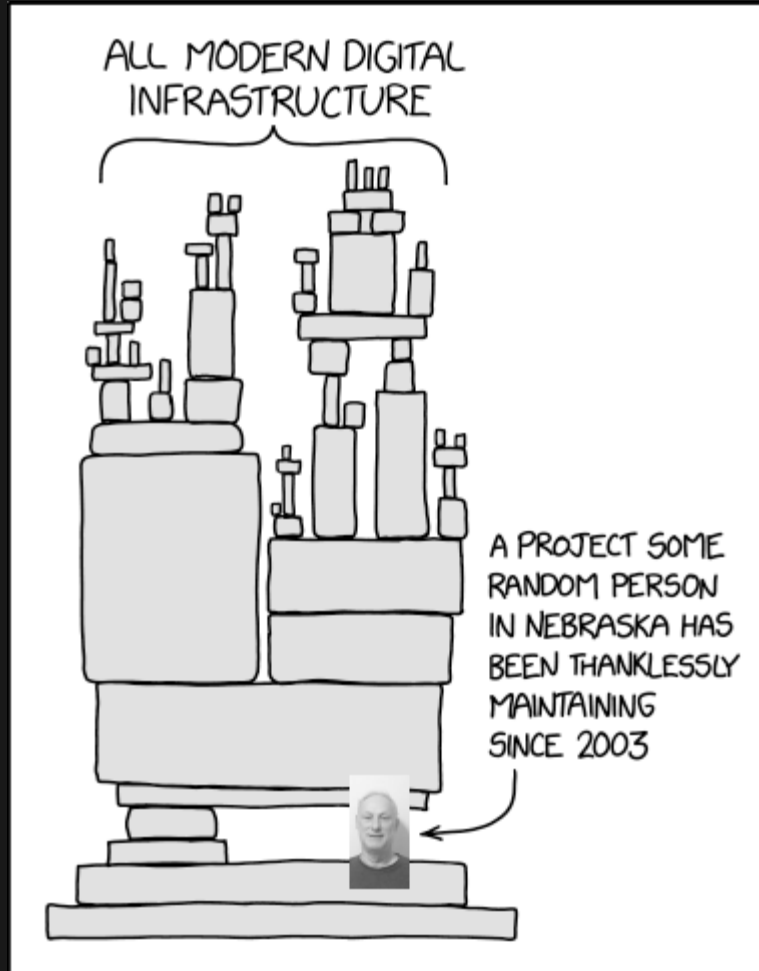
37 commits 2,577 ++ 746 --

#2



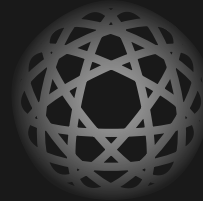
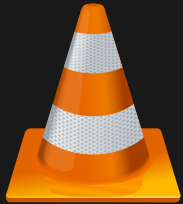
So, who *am* I?

So, who *am* I?



<https://xkcd.com/2347/>

Some Open Source Software that Includes My Code:




Background


- Malware is on the rise




Background

- Malware is on the rise 
- Decompilers are not new, but


Background

- Malware is on the rise 
- Decompilers are not new, but
- Ideas presented here are new.

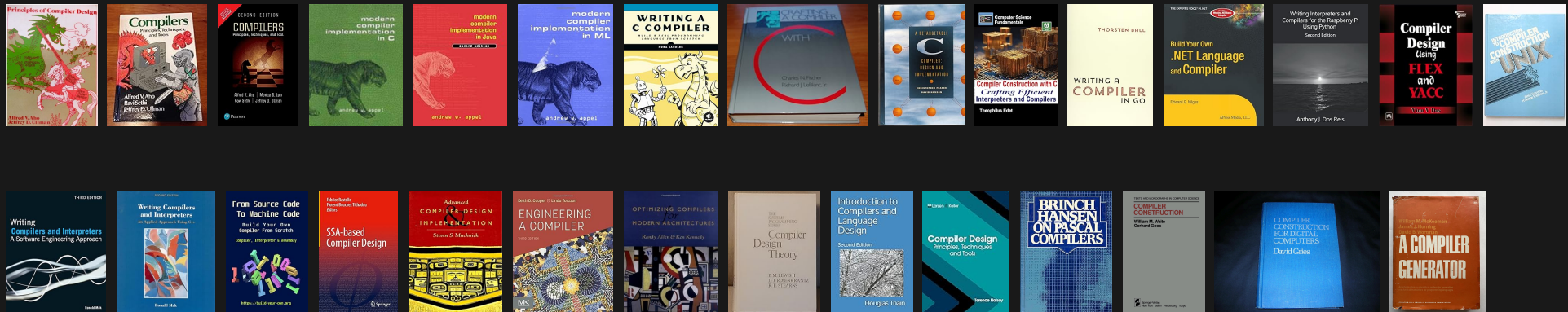
Background

- Malware is on the rise 
- Decompilers are not new, but
- Ideas presented here are new.
- "General-purpose" Decompilers

Background

- Malware is on the rise 
- Decompilers are not new, but
- Ideas presented here are new.
- "General-purpose" Decompilers
- Bytecode Decompilers (Special Purpose)

Theory and books on *making* compilers:



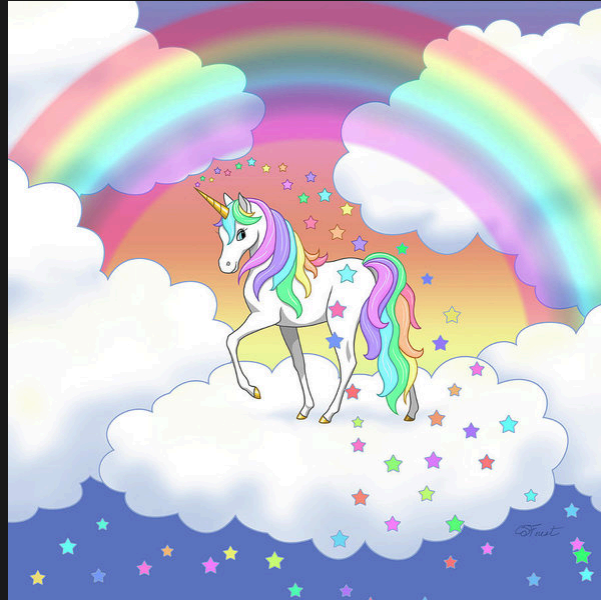
... (40 others)

Books on making *decompilers*:

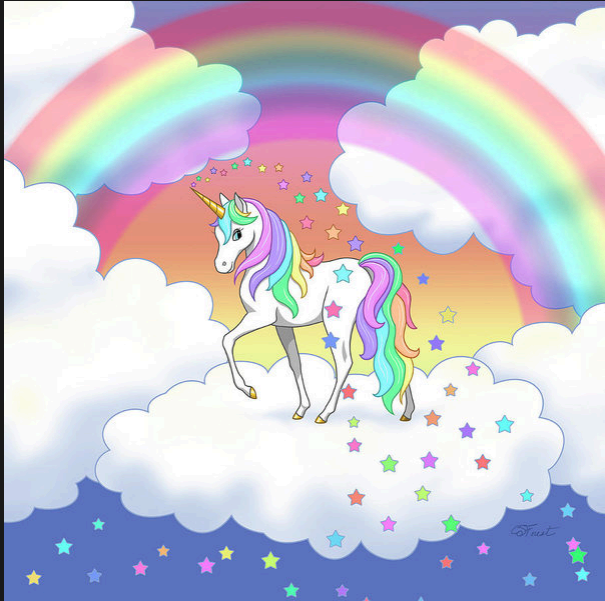
Books on making *decompilers*:



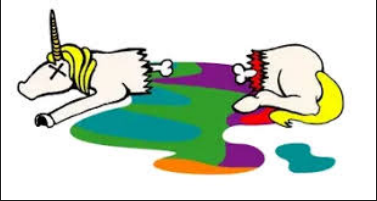
Can AI Save the Day?



Can AI Save the Day?



Not yet.



Decompilation in the World of High-Level Bytecode

Decompilation in the World of High-Level Bytecode

- Raise awareness of the differences between "General Purpose" decompiling and High-level Bytecode Decompiling

Decompilation in the World of High-Level Bytecode

- Raise awareness of the differences between "General Purpose" decompiling and High-level Bytecode Decompiling
- Introduce Decompilation as a Language-Translation problem.

Decompilation in the World of High-Level Bytecode

- Raise awareness of the differences between "General Purpose" decompiling and High-level Bytecode Decompiling
- Introduce Decompilation as a Language-Translation problem.
- Introduce Decompilation as a Compilation Process.

Key Takeaways

- Understand more about what is wrong when something goes wrong with decompilation.
- Understand the difference between *disassembly* and *decompilation*.
- Begin to see the difference between *machine code* and *high-level bytecode*.
- Understand the some limits of Python decompilation and decompilation in general.

Simple Python Program

In file `five.py`:

```
"""
BlackHat Asia Example
"""

def five():
    """Returns the string five"""
    return "5"

# Call the function we just defined.
print(five())
```

Simple Python Program

In file `five.py`:

```
"""
BlackHat Asia Example
"""

def five():
    """Returns the string five"""
    return "5"

# Call the function we just defined.
print(five())
```

Now run the code:

```
$ python five.py
5
```

Simple Python Program

In file `five.py`:

```
"""
BlackHat Asia Example
"""

def five():
    """Returns the string five"""
    return "5"

# Call the function we just defined.
print(five())
```

Now run the code:

```
$ python five.py
5
```

Instruction bytecode of main program:

```
0 |64 00 5a 00 64 01 64 02 |
8 |84 00 5a 01 65 02 65 00 |
16 |83 00 83 01 01 00 64 03 |
24 |53 00 |
```

Byte-Compiling Python

```
"""
BlackHat Asia Example
"""

def five():
    """Returns the string five"""
    return "5"

# Call the function we just defined.
print(five())
```

Byte-Compiling Python

```
"""
BlackHat Asia Example
"""

def five():
    """Returns the string five"""
    return "5"

# Call the function we just defined.
print(five())
```

Byte compile this program:

Byte-Compiling Python

```
"""
BlackHat Asia Example
"""

def five():
    """Returns the string five"""
    return "5"

# Call the function we just defined.
print(five())
```

Byte compile this program:

```
1 $ python -m compileall five.py
2 Compiling 'five.py'...
3 $ ls -l __pycache__/five.cpython-38.pyc
4 -rw-rw-r-- 1 rocky rocky 301 Feb 17 10:16 __pycache__/five.cpython-38.pyc
```


Byte-Compiling Python

```
"""
BlackHat Asia Example
"""

def five():
    """Returns the string five"""
    return "5"

# Call the function we just defined.
print(five())
```

Byte compile this program:

```
1 $ python -m compileall five.py
2 Compiling 'five.py'...
3 $ ls -l __pycache__/five.cpython-38.pyc
4 -rw-rw-r-- 1 rocky rocky 301 Feb 17 10:16 __pycache__/five.cpython-38.pyc
```

Byte-Compiling Python

```
"""
BlackHat Asia Example
"""

def five():
    """Returns the string five"""
    return "5"

# Call the function we just defined.
print(five())
```

Byte compile this program:

```
1 $ python -m compileall five.py
2 Compiling 'five.py'...
3 $ ls -l __pycache__/five.cpython-38.pyc
4 -rw-rw-r-- 1 rocky rocky 301 Feb 17 10:16 __pycache__/five.cpython-38.pyc
```

I run the bytecode:

```
$ python /tmp/five-moved.pyc
5
```

and again I get 5.

Python Bytecode Decompilation Example

```
1 $ uncompile6 /tmp/five-moved.pyc
2 # uncompile6 version 3.9.1
3 # Python bytecode version base 3.8.0 (3413)
4 # Decompiled from: Python 3.12.2 (main, Feb 14 2024, 04:48:40) [GCC 13.2.0]
5 # Embedded file name: five.py
6 # Compiled at: 2024-02-17 10:16:08
7 # Size of source mod 2**32: 153 bytes
8 """
9 BlackHat Asia Example
10 """
11
12 def five():
13     """Returns the string five"""
14     return "5"
15
16
17 print(five())
18 # okay decompiling /tmp/five-moved.pyc
```

Python Bytecode Decompilation Example

```
1 $ uncompile6 /tmp/five-moved.pyc
2 # uncompile6 version 3.9.1
3 # Python bytecode version base 3.8.0 (3413)
4 # Decompiled from: Python 3.12.2 (main, Feb 14 2024, 04:48:40) [GCC 13.2.0]
5 # Embedded file name: five.py
6 # Compiled at: 2024-02-17 10:16:08
7 # Size of source mod 2**32: 153 bytes
8 """
9 BlackHat Asia Example
10 """
11
12 def five():
13     """Returns the string five"""
14     return "5"
15
16
17 print(five())
18 # okay decompiling /tmp/five-moved.pyc
```

Python Bytecode Decompilation Example

```
1 $ uncompile6 /tmp/five-moved.pyc
2 # uncompile6 version 3.9.1
3 # Python bytecode version base 3.8.0 (3413)
4 # Decompiled from: Python 3.12.2 (main, Feb 14 2024, 04:48:40) [GCC 13.2.0]
5 # Embedded file name: five.py
6 # Compiled at: 2024-02-17 10:16:08
7 # Size of source mod 2**32: 153 bytes
8 """
9 BlackHat Asia Example
10 """
11
12 def five():
13     """Returns the string five"""
14     return "5"
15
16
17 print(five())
18 # okay decompiling /tmp/five-moved.pyc
```

Python Bytecode Decompilation Example

```
1 $ uncompile6 /tmp/five-moved.pyc
2 # uncompile6 version 3.9.1
3 # Python bytecode version base 3.8.0 (3413)
4 # Decompiled from: Python 3.12.2 (main, Feb 14 2024, 04:48:40) [GCC 13.2.0]
5 # Embedded file name: five.py
6 # Compiled at: 2024-02-17 10:16:08
7 # Size of source mod 2**32: 153 bytes
8 """
9 BlackHat Asia Example
10 """
11
12 def five():
13     """Returns the string five"""
14     return "5"
15
16
17 print(five())
18 # okay decompiling /tmp/five-moved.pyc
```

Python Bytecode Decompilation Example

```
1 $ uncompile6 /tmp/five-moved.pyc
2 # uncompile6 version 3.9.1
3 # Python bytecode version base 3.8.0 (3413)
4 # Decompiled from: Python 3.12.2 (main, Feb 14 2024, 04:48:40) [GCC 13.2.0]
5 # Embedded file name: five.py
6 # Compiled at: 2024-02-17 10:16:08
7 # Size of source mod 2**32: 153 bytes
8 """
9 BlackHat Asia Example
10 """
11
12 def five():
13     """Returns the string five"""
14     return "5"
15
16
17 print(five())
18 # okay decompiling /tmp/five-moved.pyc
```

Python Bytecode Decompilation Example

```
1 $ uncompile6 /tmp/five-moved.pyc
2 # uncompile6 version 3.9.1
3 # Python bytecode version base 3.8.0 (3413)
4 # Decompiled from: Python 3.12.2 (main, Feb 14 2024, 04:48:40) [GCC 13.2.0]
5 # Embedded file name: five.py
6 # Compiled at: 2024-02-17 10:16:08
7 # Size of source mod 2**32: 153 bytes
8 """
9 BlackHat Asia Example
10 """
11
12 def five():
13     """Returns the string five"""
14     return "5"
15
16
17 print(five())
18 # okay decompiling /tmp/five-moved.pyc
```


Source Code Differences

Source code:

```
1 """
2 BlackHat Asia Example
3 """
4
5
6 def five():
7     """Returns the string five"""
8     return "5"
9
10
11 # Call the function we just defined.
12 print(five())
```

Decompiled code:

```
1 # uncompile6 version 3.9.1
2 # Python bytecode version base 3.8.0 (3413)
3 # Decompiled from: Python 3.12.2 (main, Feb 14 2024,
4 # Embedded file name: five.py
5 # Compiled at: 2024-02-17 10:16:08
6 # Size of source mod 2**32: 153 bytes
7 """
8 BlackHat Asia Example
9 """
10
11 def five():
12     """Returns the string five"""
13     return "5"
14
15
16 print(five())
17
18 # okay decompiling /tmp/five-moved.pyc
```

Source Code Differences

Source code:

```
1 """
2 BlackHat Asia Example
3 """
4
5
6 def five():
7     """Returns the string five"""
8     return "5"
9
10
11 # Call the function we just defined.
12 print(five())
```

Decompiled code:

```
1 # uncompile6 version 3.9.1
2 # Python bytecode version base 3.8.0 (3413)
3 # Decompiled from: Python 3.12.2 (main, Feb 14 2024,
4 # Embedded file name: five.py
5 # Compiled at: 2024-02-17 10:16:08
6 # Size of source mod 2**32: 153 bytes
7 """
8 BlackHat Asia Example
9 """
10
11 def five():
12     """Returns the string five"""
13     return "5"
14
15
16 print(five())
17
18 # okay decompiling /tmp/five-moved.pyc
```

Source Code Differences

Source code:

```
1 """
2 BlackHat Asia Example
3 """
4
5
6 def five():
7     """Returns the string five"""
8     return "5"
9
10
11 # Call the function we just defined.
12 print(five())
```

Decompiled code:

```
1 # uncompile6 version 3.9.1
2 # Python bytecode version base 3.8.0 (3413)
3 # Decompiled from: Python 3.12.2 (main, Feb 14 2024,
4 # Embedded file name: five.py
5 # Compiled at: 2024-02-17 10:16:08
6 # Size of source mod 2**32: 153 bytes
7 """
8 BlackHat Asia Example
9 """
10
11 def five():
12     """Returns the string five"""
13     return "5"
14
15
16 print(five())
17
18 # okay decompiling /tmp/five-moved.pyc
```

How These Decompilers Work

How These Decompilers Work

Decompilation processed in a pipeline of these phases:

How These Decompilers Work

Decompilation processed in a pipeline of these phases:

1. Get bytecode disassembly via [xdis](#)

How These Decompilers Work

Decompilation processed in a pipeline of these phases:

1. Get bytecode disassembly via `xdis`
2. Tokenize or "lift" the disassembly

How These Decompilers Work

Decompilation processed in a pipeline of these phases:

1. Get bytecode disassembly via `xdis`
2. Tokenize or "lift" the disassembly
3. Parse tokens into a Parse Tree

How These Decompilers Work

Decompilation processed in a pipeline of these phases:

1. Get bytecode disassembly via [xdis](#)
2. Tokenize or "lift" the disassembly
3. Parse tokens into a Parse Tree
4. Abstract the Parse Tree into an Abstract Syntax Tree

How These Decompilers Work

Decompilation processed in a pipeline of these phases:

1. Get bytecode disassembly via [xdis](#)
2. Tokenize or "lift" the disassembly
3. Parse tokens into a Parse Tree
4. Abstract the Parse Tree into an Abstract Syntax Tree
5. Produce Source from the Abstract Syntax Tree

Disassembly using **pydisasm** from **xdis**

```
$ pydisasm /tmp/five_moved.pyc
```

Disassembly using **pydisasm** from **xdis**

```
$ pydisasm /tmp/five_moved.pyc
```

```
1 # pydisasm version 6.1.0
2 # Python bytecode 3.8.0 (3413)
3 # Disassembled from Python 3.8.18 (default, Sep  4 2023, 13:19:52)
4 # [GCC 12.3.0]
5 # Timestamp in code: 1708217267 (2024-02-17 19:47:47)
6 # Source code size mod 2**32: 148 bytes
7 # Method Name:      <module>
8 # Filename:        five.py
9 # Argument count:   0
10 # Position-only argument count: 0
11 # Keyword-only arguments: 0
12 # Number of locals: 0
13 # Stack size:      2
14 # Flags:           0x00000040 (NOFREE)
15 # First Line:      1
```

Disassembly using **pydisasm** from **xdis**

```
$ pydisasm /tmp/five_moved.pyc
```

```
1 # pydisasm version 6.1.0
2 # Python bytecode 3.8.0 (3413)
3 # Disassembled from Python 3.8.18 (default, Sep  4 2023, 13:19:52)
4 # [GCC 12.3.0]
5 # Timestamp in code: 1708217267 (2024-02-17 19:47:47)
6 # Source code size mod 2**32: 148 bytes
7 # Method Name:      <module>
8 # Filename:        five.py
9 # Argument count:   0
10 # Position-only argument count: 0
11 # Keyword-only arguments: 0
12 # Number of locals: 0
13 # Stack size:      2
14 # Flags:           0x00000040 (NOFREE)
15 # First line:      1
```

Disassembly using **pydisasm** from **xdis**

```
$ pydisasm /tmp/five_moved.pyc
```

```
19 #      2: 'five'
20 #      3: None
21 # Names:
22 #      0: __doc__
23 #      1: five
24 #      2: print
25 1:      0 LOAD_CONST      ("\\nBlackHat Asia Example\\n")
26      2 STORE_NAME      (__doc__)
27
28 6:      4 LOAD_CONST      (<code object five at 0x7f3f4c3d17c0, file "five.py",
29      6 LOAD_CONST      ("five")
30      8 MAKE_FUNCTION   (Neither defaults, keyword-only args, annotations, nor
31     10 STORE_NAME      (five)
32
33 12:     12 LOAD_NAME       (print)
```

Phases 1 and 2: Bytecode to Tokens

Bytecode Disassembly:

```
1  1: 0 LOAD_CONST      ('\nBlackHat Asia Example\n')
2    2 STORE_NAME      (__doc__)
3
4  6: 4 LOAD_CONST      <code object five>
5    6 LOAD_CONST      ('five')
6    8 MAKE_FUNCTION    (No parameters)
7   10 STORE_NAME      (five)
```

Parser Input Tokens:

```
1  1: 0 LOAD_STR        ('\nBlackHat Asia Example\n')
2    2 STORE_NAME      (__doc__)
3
4  6: 4 LOAD_CODE       <code_object five>
5    6 LOAD_STR        ('five')
6    8 MAKE_FUNCTION_0 (No parameters)
7   10 STORE_NAME      (five)
```

Phases 1 and 2: Bytecode to Tokens

Bytecode Disassembly:

```
1 1: 0 LOAD_CONST      ('\nBlackHat Asia Example\n')
2   2 STORE_NAME      (__doc__)
3
4 6: 4 LOAD_CONST      <code object five>
5   6 LOAD_CONST      ('five')
6   8 MAKE_FUNCTION    (No parameters)
7  10 STORE_NAME      (five)
```

Parser Input Tokens:

```
1 1: 0 LOAD_STR        ('\nBlackHat Asia Example\n')
2   2 STORE_NAME      (__doc__)
3
4 6: 4 LOAD_CODE       <code_object five>
5   6 LOAD_STR        ('five')
6   8 MAKE_FUNCTION_0 (No parameters)
7  10 STORE_NAME      (five)
```


Phases 1 and 2: Bytecode to Tokens

Bytecode Disassembly:

```
1 1: 0 LOAD_CONST      ('\nBlackHat Asia Example\n')
2   2 STORE_NAME      (__doc__)
3
4 6: 4 LOAD_CONST      <code object five>
5   6 LOAD_CONST      ('five')
6   8 MAKE_FUNCTION    (No parameters)
7  10 STORE_NAME      (five)
```

Parser Input Tokens:

```
1 1: 0 LOAD_STR        ('\nBlackHat Asia Example\n')
2   2 STORE_NAME      (__doc__)
3
4 6: 4 LOAD_CODE       <code_object five>
5   6 LOAD_STR        ('five')
6   8 MAKE_FUNCTION_0 (No parameters)
7  10 STORE_NAME      (five)
```

Phase 3: Parsing Tokens into a Parse Tree

```
"""  
BlackHat Asia Example  
"""
```

Constructing Parse Tree from Tokens:

```
LOAD_STR    '\nBlackHat Asia Example\n'  
STORE_NAME  __doc__
```

Phase 3: Parsing Tokens into a Parse Tree

```
"""  
BlackHat Asia Example  
"""
```

Constructing Parse Tree from Tokens:

```
LOAD_STR    '\nBlackHat Asia Example\n'  
STORE_NAME __doc__
```

Is parsed:

Phase 3: Parsing Tokens into a Parse Tree

```
"""  
BlackHat Asia Example  
"""
```

Constructing Parse Tree from Tokens:

```
LOAD_STR    '\nBlackHat Asia Example\n'  
STORE_NAME __doc__
```

Is parsed:

```
LOAD_STR
```

Phase 3: Parsing Tokens into a Parse Tree

```
"""  
BlackHat Asia Example  
"""
```

Constructing Parse Tree from Tokens:

```
LOAD_STR    '\nBlackHat Asia Example\n'  
STORE_NAME __doc__
```

Is parsed:

```
LOAD_STR
```

```
expr ::= LOAD_STR
```

Phase 3: Parsing Tokens into a Parse Tree

```
"""  
BlackHat Asia Example  
"""
```

Constructing Parse Tree from Tokens:

```
LOAD_STR    '\nBlackHat Asia Example\n'  
STORE_NAME __doc__
```

Is parsed:

```
LOAD_STR
```

```
expr ::= LOAD_STR
```



Parsing Tokens into a Parse Tree (Part 2)

Constructing Parse Tree from Tokens:

```
LOAD_STR  
expr ::= LOAD_STR
```

Parsing Tokens into a Parse Tree (Part 2)

Constructing Parse Tree from Tokens:

```
LOAD_STR
```

```
expr ::= LOAD_STR
```

```
STORE_NAME
```


Parsing Tokens into a Parse Tree (Part 2)

Constructing Parse Tree from Tokens:

```
LOAD_STR
```

```
expr ::= LOAD_STR
```

```
STORE_NAME
```

```
store ::= STORE_NAME
```

Parsing Tokens into a Parse Tree (Part 2)

Constructing Parse Tree from Tokens:

```
LOAD_STR  
expr ::= LOAD_STR
```

```
STORE_NAME
```

```
store ::= STORE_NAME
```



Parsing Tokens into a Parse Tree (Part 3)

```
LOAD_STR
expr ::= LOAD_STR
STORE_NAME
store ::= STORE_NAME
```

Parsing Tokens into a Parse Tree (Part 3)

```
LOAD_STR
```

```
expr ::= LOAD_STR
```

```
STORE_NAME
```

```
store ::= STORE_NAME
```

```
assign ::= expr store
```

Parsing Tokens into a Parse Tree (Part 3)

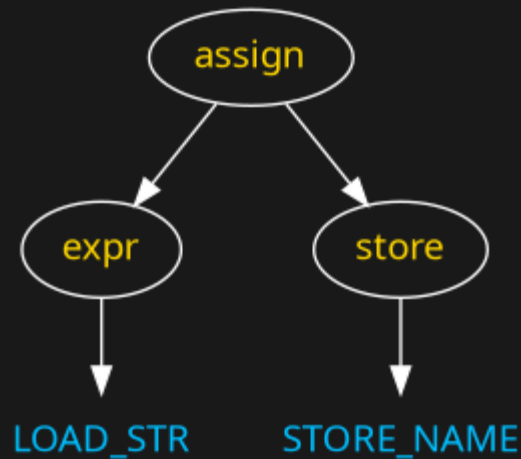
LOAD_STR

expr ::= LOAD_STR

STORE_NAME

store ::= STORE_NAME

assign ::= expr store



Parsing Tokens into a Parse Tree (Part 4)

```
LOAD_STR
expr ::= LOAD_STR
STORE_NAME
store ::= STORE_NAME
assign ::= expr store
```

Parsing Tokens into a Parse Tree (Part 4)

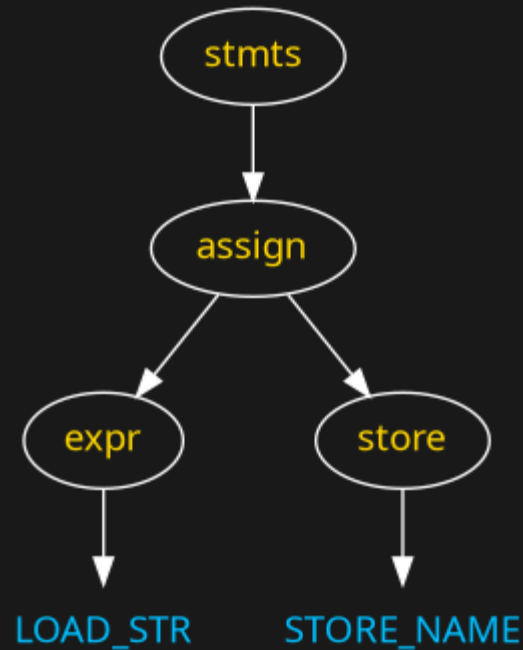
```
LOAD_STR
expr ::= LOAD_STR
STORE_NAME
store ::= STORE_NAME
assign ::= expr store
```

```
stmts ::= assign
```

Parsing Tokens into a Parse Tree (Part 4)

```
LOAD_STR
expr ::= LOAD_STR
STORE_NAME
store ::= STORE_NAME
assign ::= expr store
```

```
stmts ::= assign
```



Phases 3 - 5: Final Parse Tree to Source Text

Phase 3. Parse Tree in ASCII Format (First Line):

```
stmts
  0. assign
    0. expr
      L. 1 0 LOAD_STR  '\nBlackHat Asia Example\n'
    1. store
      2 STORE_NAME  __doc__
```

Phases 3 - 5: Final Parse Tree to Source Text

Phase 3. Parse Tree in ASCII Format (First Line):

```
stmts
  0. assign
    0. expr
      L. 1 0 LOAD_STR '\nBlackHat Asia Example\n'
    1. store
      2 STORE_NAME __doc__
```

Phase 4. Abstract Syntax Tree (First Line)

```
stmts
  0. docstring
    0 LOAD_STR '\nBlackHat Asia Example\n'
    2 STORE_NAME __doc__
```

Phases 3 - 5: Final Parse Tree to Source Text

Phase 3. Parse Tree in ASCII Format (First Line):

```
stmts
  0. assign
    0. expr
      L. 1 0 LOAD_STR '\nBlackHat Asia Example\n'
    1. store
      2 STORE_NAME __doc__
```

Phase 4. Abstract Syntax Tree (First Line)

```
stmts
  0. docstring
    0 LOAD_STR '\nBlackHat Asia Example\n'
    2 STORE_NAME __doc__
```

Phase 5. Printing the Abstract Syntax Tree (First line)

```
"""
BlackHat Asia Example
"""
```



Bytecode Disassembly

```
0 |64 00 5a 00 64 01 64 02 |  
8 |84 00 5a 01 65 02 65 00 |  
16|83 00 83 01 01 00 64 03 |  
24|53 00 |
```

Bytecode Disassembly

```
0 |64 00 5a 00 64 01 64 02 |  
8 |84 00 5a 01 65 02 65 00 |  
16|83 00 83 01 01 00 64 03 |  
24|53 00 |
```

Disassembly using `pydisasm` from `xdis`:

Bytecode Disassembly

```
0 |64 00 5a 00 64 01 64 02 |  
8 |84 00 5a 01 65 02 65 00 |  
16|83 00 83 01 01 00 64 03 |  
24|53 00 |
```

Disassembly using `pydisasm` from `xdis`:

```
$ pydisasm -F extended-bytes -S __pycache__/five.cpython-38.pyc
```

Bytecode Disassembly

```
0 |64 00 5a 00 64 01 64 02 |
8 |84 00 5a 01 65 02 65 00 |
12|65 02 65 01 83 01 01 00 |
16|83 00 83 01 01 00 64 03 |
24|53 00 |
```

```
$ pydisasm -F extended-bytes -S __pycache__/five.cpython-38.pyc
```

```
1 # ...
2 # Constants:
3 #   0: '\nBlackHat Asia Example\n'
4 #   1: <code object five at 0x7f64cb56c030, file "five.py", line 6>
5 #   2: 'five'
6 #   3: None
7 # Names:
8 #   0: __doc__
9 #   1: five
10 #   2: print
11 # """\nBlackHat Asia Example\n""
12 1:   0 |64 00| LOAD_CONST      ("\nBlackHat Asia Example\n")
13   2 |5a 00| STORE_NAME      (__doc__) ; __doc__ = '\nBlackHat Asia Example\n'
14
15 # def five():
16 6:   4 |64 01| LOAD_CONST      (<code object five at 0x7f64cb56c030, file "five.py">)
17   6 |64 02| LOAD_CONST      ("five")
18   8 |84 00| MAKE_FUNCTION   (No arguments) ; TOS = def five(...): ...
19  10 |5a 01| STORE_NAME      (five) ; five = def five(...): ...
20
```


Bytecode Disassembly

```
0 |64 00 5a 00 64 01 64 02 |
8 |84 00 5a 01 65 02 65 00 |
12|65 02 65 01 83 01 01 00 |
16|83 00 83 01 01 00 64 03 |
24|53 00 |
```

```
$ pydisasm -F extended-bytes -S __pycache__/five.cpython-38.pyc
```

```
1 # ...
2 # Constants:
3 # 0: '\nBlackHat Asia Example\n'
4 # 1: <code object five at 0x7f64cb56c030, file "five.py", line 6>
5 # 2: 'five'
6 # 3: None
7 # Names:
8 # 0: __doc__
9 # 1: five
10 # 2: print
11 # """\nBlackHat Asia Example\n""
12 1: 0 |64 00| LOAD_CONST      ("\nBlackHat Asia Example\n")
13   2 |5a 00| STORE_NAME      (__doc__) ; __doc__ = '\nBlackHat Asia Example\n'
14
15 # def five():
16 6: 4 |64 01| LOAD_CONST      (<code object five at 0x7f64cb56c030, file "five.py">)
17   6 |64 02| LOAD_CONST      ("five")
18   8 |84 00| MAKE_FUNCTION   (No arguments) ; TOS = def five(...): ...
19  10 |5a 01| STORE_NAME      (five) ; five = def five(...): ...
20
```

Bytecode Disassembly

```
0 |64 00 5a 00 64 01 64 02 |
8 |84 00 5a 01 65 02 65 00 |
12|65 02 65 01 83 01 01 00 |
16|83 00 83 01 01 00 64 03 |
24|53 00 |
```

```
$ pydisasm -F extended-bytes -S __pycache__/five.cpython-38.pyc
```

```
1 # ...
2 # Constants:
3 #   0: '\nBlackHat Asia Example\n'
4 #   1: <code object five at 0x7f64cb56c030, file "five.py", line 6>
5 #   2: 'five'
6 #   3: None
7 # Names:
8 #   0: __doc__
9 #   1: five
10 #   2: print
11 # """\nBlackHat Asia Example\n""
12 1:   0 |64 00| LOAD_CONST      ("\nBlackHat Asia Example\n")
13   2 |5a 00| STORE_NAME      (__doc__) ; __doc__ = '\nBlackHat Asia Example\n'
14
15 # def five():
16 6:   4 |64 01| LOAD_CONST      (<code object five at 0x7f64cb56c030, file "five.py">)
17   6 |64 02| LOAD_CONST      ("five")
18   8 |84 00| MAKE_FUNCTION   (No arguments) ; TOS = def five(...): ...
19  10 |5a 01| STORE_NAME      (five) ; five = def five(...): ...
20
```

Bytecode Disassembly

```
0 |64 00 5a 00 64 01 64 02 |
8 |84 00 5a 01 65 02 65 00 |
12|65 02 65 01 83 01 01 00 |
16|83 00 83 01 01 00 64 03 |
24|53 00 |
```

```
$ pydisasm -F extended-bytes -S __pycache__/five.cpython-38.pyc
```

```
1 # ...
2 # Constants:
3 #   0: '\nBlackHat Asia Example\n'
4 #   1: <code object five at 0x7f64cb56c030, file "five.py", line 6>
5 #   2: 'five'
6 #   3: None
7 # Names:
8 #   0: __doc__
9 #   1: five
10 #   2: print
11 # """\nBlackHat Asia Example\n""
12 1:   0 |64 00| LOAD_CONST      ("\nBlackHat Asia Example\n")
13   2 |5a 00| STORE_NAME      (__doc__) ; __doc__ = '\nBlackHat Asia Example\n'
14
15 # def five():
16 6:   4 |64 01| LOAD_CONST      (<code object five at 0x7f64cb56c030, file "five.py">)
17   6 |64 02| LOAD_CONST      ("five")
18   8 |84 00| MAKE_FUNCTION   (No arguments) ; TOS = def five(...): ...
19  10 |5a 01| STORE_NAME      (five) ; five = def five(...): ...
20
21 # print(five())
```

Bytecode Disassembly

```
0 |64 00 5a 00 64 01 64 02 |
8 |84 00 5a 01 65 02 65 00 |
12|65 02 65 01 83 01 01 00 |
16|83 00 83 01 01 00 64 03 |
24|53 00 |
```

```
$ pydisasm -F extended-bytes -S __pycache__/five.cpython-38.pyc
```

```
1 # ...
2 # Constants:
3 # 0: '\nBlackHat Asia Example\n'
4 # 1: <code object five at 0x7f64cb56c030, file "five.py", line 6>
5 # 2: 'five'
6 # 3: None
7 # Names:
8 # 0: __doc__
9 # 1: five
10 # 2: print
11 # """\nBlackHat Asia Example\n""
12 1: 0 |64 00| LOAD_CONST      ("\nBlackHat Asia Example\n")
13   2 |5a 00| STORE_NAME      (__doc__) ; __doc__ = '\nBlackHat Asia Example\n'
14
15 # def five():
16 6: 4 |64 01| LOAD_CONST      (<code object five at 0x7f64cb56c030, file "five.py">)
17   6 |64 02| LOAD_CONST      ("five")
18   8 |84 00| MAKE_FUNCTION   (No arguments) ; TOS = def five(...): ...
19  10 |5a 01| STORE_NAME      (five) ; five = def five(...): ...
20
```

Bytecode Disassembly

```
0 |64 00 5a 00 64 01 64 02 |
8 |84 00 5a 01 65 02 65 00 |
12|65 02 65 01 83 01 01 00 |
16|83 00 83 01 01 00 64 03 |
24|53 00 |
```

```
$ pydisasm -F extended-bytes -S __pycache__/five.cpython-38.pyc
```

```
1 # ...
2 # Constants:
3 #   0: '\nBlackHat Asia Example\n'
4 #   1: <code object five at 0x7f64cb56c030, file "five.py", line 6>
5 #   2: 'five'
6 #   3: None
7 # Names:
8 #   0: __doc__
9 #   1: five
10 #   2: print
11 # """\nBlackHat Asia Example\n""
12 1:   0 |64 00| LOAD_CONST      ("\nBlackHat Asia Example\n")
13   2 |5a 00| STORE_NAME      (__doc__) ; __doc__ = '\nBlackHat Asia Example\n'
14
15 # def five():
16 6:   4 |64 01| LOAD_CONST      (<code object five at 0x7f64cb56c030, file "five.py">)
17   6 |64 02| LOAD_CONST      ("five")
18   8 |84 00| MAKE_FUNCTION   (No arguments) ; TOS = def five(...): ...
19  10 |5a 01| STORE_NAME      (five) ; five = def five(...): ...
20
```

Bytecode Disassembly

```
0 |64 00 5a 00 64 01 64 02 |
8 |84 00 5a 01 65 02 65 00 |
12|65 02 65 01 83 01 01 00 |
16|83 00 83 01 01 00 64 03 |
24|53 00 |
```

```
$ pydisasm -F extended-bytes -S __pycache__/five.cpython-38.pyc
```

```
1 # ...
2 # Constants:
3 #   0: '\nBlackHat Asia Example\n'
4 #   1: <code object five at 0x7f64cb56c030, file "five.py", line 6>
5 #   2: 'five'
6 #   3: None
7 # Names:
8 #   0: __doc__
9 #   1: five
10 #   2: print
11 # """\nBlackHat Asia Example\n"""
12 1:   0 |64 00| LOAD_CONST      ("\nBlackHat Asia Example\n")
13     2 |5a 00| STORE_NAME         (__doc__) ; __doc__ = '\nBlackHat Asia Example\n'
14
15 # def five():
16 6:   4 |64 01| LOAD_CONST      (<code object five at 0x7f64cb56c030, file "five.py">)
17     6 |64 02| LOAD_CONST      ("five")
18     8 |84 00| MAKE_FUNCTION   (No arguments) ; TOS = def five(...): ...
19    10 |5a 01| STORE_NAME         (five) ; five = def five(...): ...
20
```

Bytecode Disassembly

```
0 |64 00 5a 00 64 01 64 02 |
8 |84 00 5a 01 65 02 65 00 |
12|65 02 65 01 83 01 01 00 |
16|83 00 83 01 01 00 64 03 |
24|53 00 |
```

```
$ pydisasm -F extended-bytes -S __pycache__/five.cpython-38.pyc
```

```
13         2 |5a 00| STORE_NAME      (__doc__) ; __doc__ = '\nBlackHat Asia Example\n'
14
15     # def five():
16 6:      4 |64 01| LOAD_CONST      (<code object five at 0x7f64cb56c030, file "five.py">)
17         6 |64 02| LOAD_CONST      ("five")
18         8 |84 00| MAKE_FUNCTION   (No arguments) ; TOS = def five(...): ...
19        10 |5a 01| STORE_NAME      (five) ; five = def five(...): ...
20
21     # print(five())
22 12:     12 |65 02| LOAD_NAME       (print)
23         14 |65 01| LOAD_NAME       (five)
24         16 |83 00| CALL_FUNCTION   (0 positional arguments) ; TOS = five()
25         18 |83 01| CALL_FUNCTION   (1 positional argument) ; TOS = print(five())
26         20 |01 00| POP_TOP
27         22 |64 03| LOAD_CONST      (None)
28        24 |53 00| RETURN_VALUE    return None
29
30
31 # Method Name:      five
32 # Filename:         five.py
33 # Argument count:   0
```

Chained Compare Bytecode

Python Expression: "a" <= __file__ <= "b"

```
1 1:      0 |64 00| LOAD_CONST      ("a")
2        2 |65 00| LOAD_NAME       (__file__)
3        4 |04 00| DUP_TOP
4        6 |03 00| ROT_THREE
5        8 |6b 01| COMPARE_OP      (<=)
6       10 |6f 12| JUMP_IF_FALSE_OR_POP (to 18)
7       12 |64 01| LOAD_CONST      ("b")
8       14 |6b 01| COMPARE_OP      (<=) ; TOS = (to 18) <= "b"
9       16 |6e 04| JUMP_FORWARD    (to 22)
10      >> 18 |02 00| ROT_TWO
11      20 |01 00| POP_TOP
12      >> 22 |01 00| POP_TOP
13     24 |64 02| LOAD_CONST      (None)
14     26 |53 00| RETURN_VALUE    return None
```


Chained Compare Bytecode

Python Expression: "a" <= __file__ <= "b"

```
1  1:      0 |64 00| LOAD_CONST      ("a")
2          2 |65 00| LOAD_NAME       (__file__)
3          4 |04 00| DUP_TOP
4          6 |03 00| ROT_THREE
5          8 |6b 01| COMPARE_OP      (<=)
6         10 |6f 12| JUMP_IF_FALSE_OR_POP (to 18)
7         12 |64 01| LOAD_CONST      ("b")
8         14 |6b 01| COMPARE_OP      (<=) ; TOS = (to 18) <= "b"
9         16 |6e 04| JUMP_FORWARD    (to 22)
10    >>  18 |02 00| ROT_TWO
11          20 |01 00| POP_TOP
12    >>  22 |01 00| POP_TOP
13          24 |64 02| LOAD_CONST      (None)
14          26 |53 00| RETURN_VALUE    return None
```

Chained Compare Parse Tree (new code)

Python Expression: "a" <= __file__ <= "b"

```
1 BB_START 1 'Basic Block 1'
2 compare_chained
3 0. expr
4 ...
5 1. compare_chained_middle
6 ...
7 4. jifop
8 0. 10 JUMP_IF_FALSE_OR_POP 18 'to 18'
9 nnn 1. 10 BB_END 1 'Basic Block 1'
10 5. 12 BB_START 2 'Basic Block 2'
11 6. compare_chained_right
12 ...
13 2. 16 JUMP_FORWARD 22 'to 22'
14 3. 16 BB_END 2 'Basic Block 2'
15 2. 18 BB_START 3 'Basic Block 3'
16 3. 18 SIBLING_BLOCK
17 4. 18 ROT_TWO
18 5. 20 POP_TOP
19 6. 20 BB_END 3 'Basic Block 3'
20 7. 20 BLOCK_END_JOIN 3 'Basic Block DominatorSet<{3}>'
```

Chained Compare Parse Tree (new code)

Python Expression: "a" <= __file__ <= "b"

```
1 BB_START 1 'Basic Block 1'
2 compare_chained
3 0. expr
4 ...
5 1. compare_chained_middle
6 ...
7 4. jifop
8 0. 10 JUMP_IF_FALSE_OR_POP 18 'to 18'
9 nnn 1. 10 BB_END 1 'Basic Block 1'
10 5. 12 BB_START 2 'Basic Block 2'
11 6. compare_chained_right
12 ...
13 2. 16 JUMP_FORWARD 22 'to 22'
14 3. 16 BB_END 2 'Basic Block 2'
15 2. 18 BB_START 3 'Basic Block 3'
16 3. 18 SIBLING_BLOCK
17 4. 18 ROT_TWO
18 5. 20 POP_TOP
19 6. 20 BB_END 3 'Basic Block 3'
20 7. 20 BLOCK_END_JOIN 3 'Basic Block DominatorSet<{3}>'
```

Classifying Scopes and Important Control-Flow Points

```
1  i: int=6
2  zero_bits = 0
3  one_bits = 0
4  # loop dominator
5  while i > 0:
6      # if dominator
7      if i % 0:
8          # first sibling
9          one_bits += 1
10     else:
11         # second sibling
12         zero_bits += 1
13     # join point
14     i << 1
15 # loop-end join point
16 print(one_bits, zero_bits)
```

Classifying Scopes and Important Control-Flow Points

```
1  i: int=6
2  zero_bits = 0
3  one_bits = 0
4  # loop dominator
5  while i > 0:
6      # if dominator
7      if i % 0:
8          # first sibling
9          one_bits += 1
10     else:
11         # second sibling
12         zero_bits += 1
13     # join point
14     i << 1
15 # loop-end join point
16 print(one_bits, zero_bits)
```

Classifying Scopes and Important Control-Flow Points

```
1  i: int=6
2  zero_bits = 0
3  one_bits = 0
4  # loop dominator
5  while i > 0:
6      # if dominator
7      if i % 0:
8          # first sibling
9          one_bits += 1
10     else:
11         # second sibling
12         zero_bits += 1
13     # join point
14     i << 1
15 # loop-end join point
16 print(one_bits, zero_bits)
```

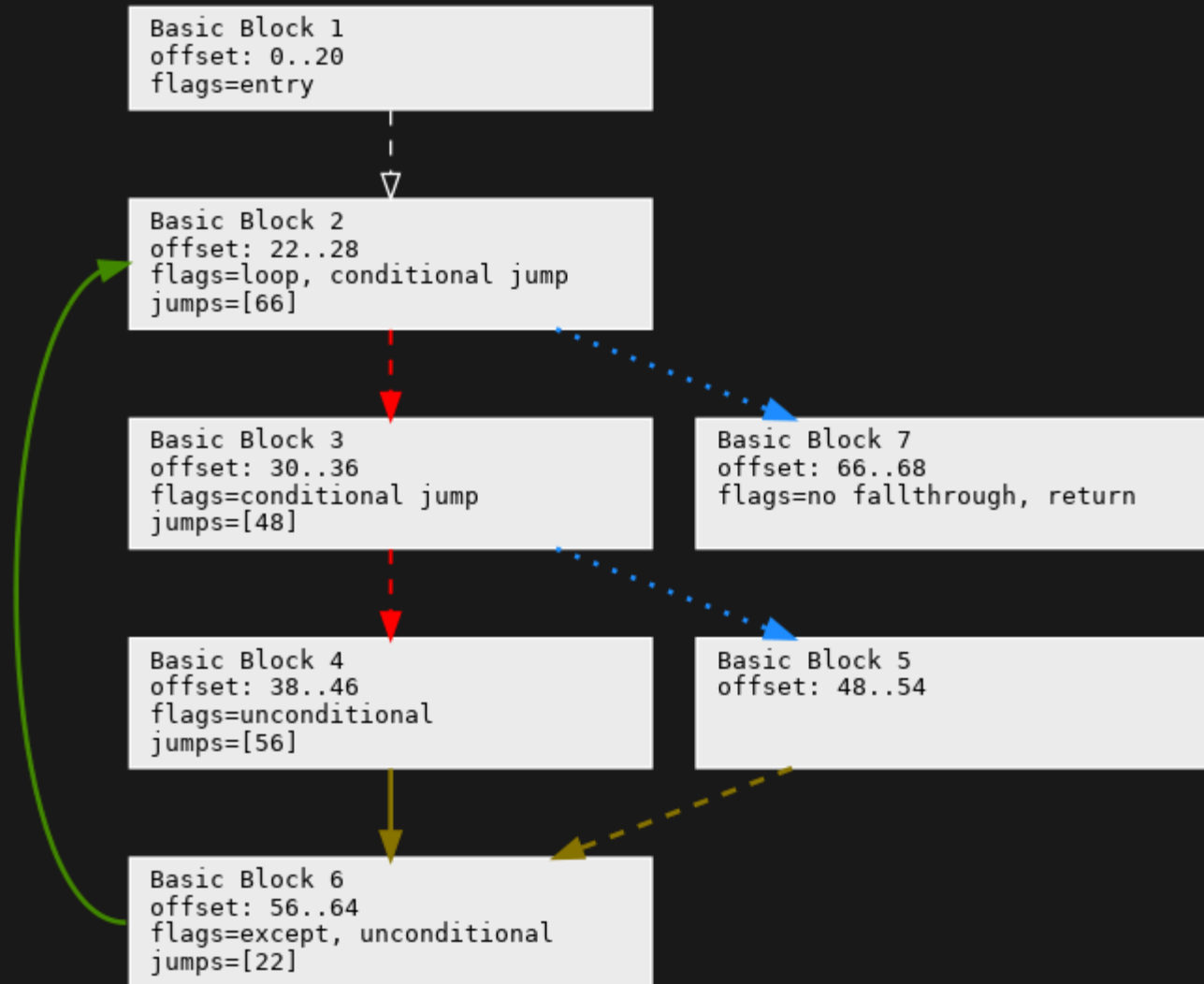
Classifying Scopes and Important Control-Flow Points

```
1  i: int=6
2  zero_bits = 0
3  one_bits = 0
4  # loop dominator
5  while i > 0:
6      # if dominator
7      if i % 0:
8          # first sibling
9          one_bits += 1
10     else:
11         # second sibling
12         zero_bits += 1
13     # join point
14     i << 1
15 # loop-end join point
16 print(one_bits, zero_bits)
```

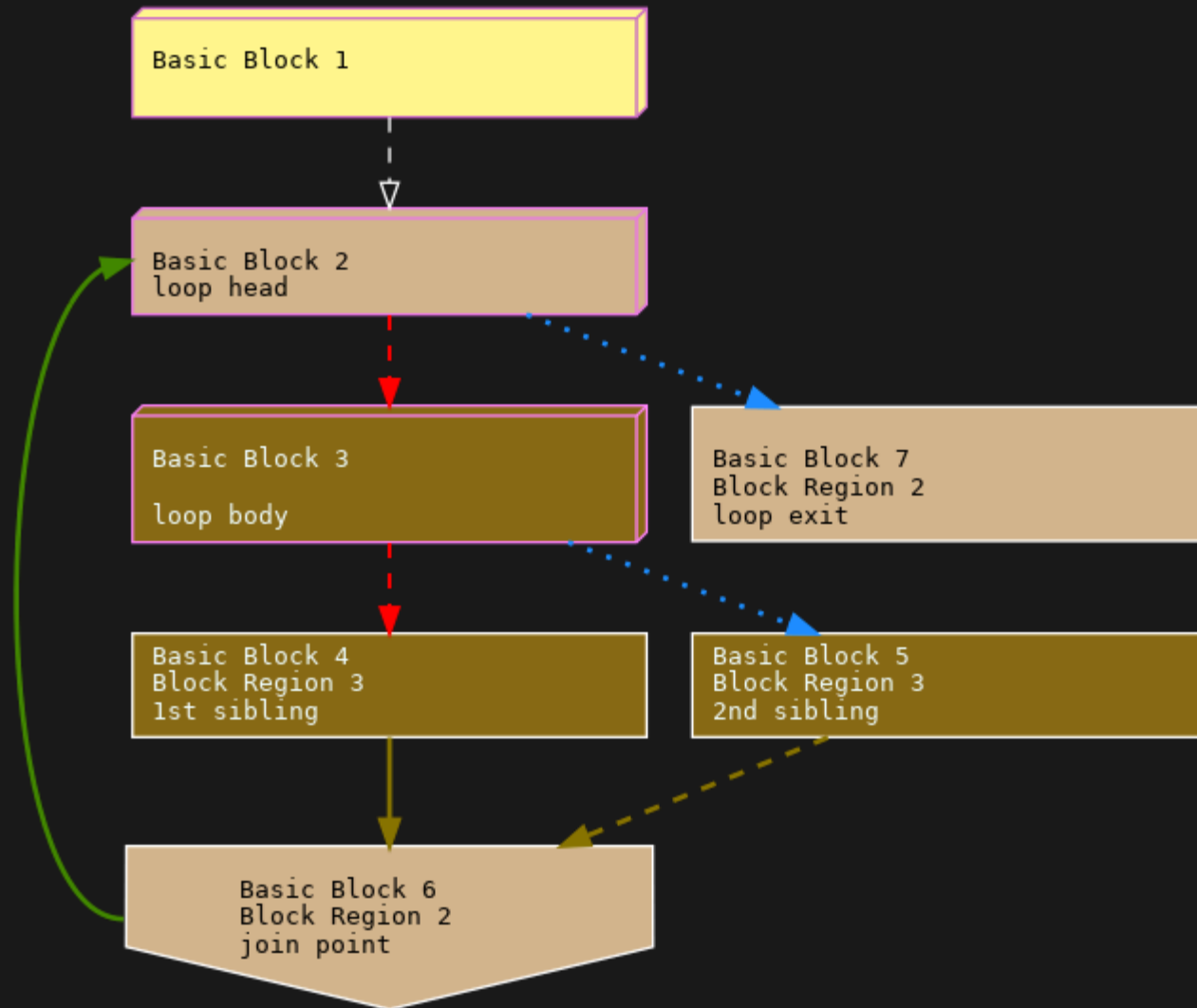
Classifying Scopes and Important Control-Flow Points

```
1  i: int=6
2  zero_bits = 0
3  one_bits = 0
4  # loop dominator
5  while i > 0:
6      # if dominator
7      if i % 0:
8          # first sibling
9          one_bits += 1
10     else:
11         # second sibling
12         zero_bits += 1
13     # join
14     i << 1
15 # loop-end join
16 print(one_bits, zero_bits)
```


Control Flow Produced from `control_flow`



Dominator Regions and Dominators



General Remarks

General Remarks

- Other Python decompilers

General Remarks

- Other Python decompilers
- General-Purpose Decompilers

General Remarks

- Other Python decompilers
- General-Purpose Decompilers
- How Control Flow Differs

General Remarks

- Other Python decompilers
- General-Purpose Decompilers
- How Control Flow Differs
- Choice of Intermediate Language

Wrapping Up

1. Understand, pinpoint, report, and even *fix* problems.
2. Understand how Python text code is related to its bytecode.
3. Extend this code for newer Python Bytecode.
4. Use these techniques in other High-level Bytecode Languages:



Thanks

- John Aycott and Hartmut Goebel

Thanks

- John Aycott and Hartmut Goebel
- BlackHat 2024 Asia Reviewers and Organizers

Thanks

- John Aycott and Hartmut Goebel
- BlackHat 2024 Asia Reviewers and Organizers
- Phil Young and Speaker-Coaching Program

Thanks

- John Aycott and Hartmut Goebel
- BlackHat 2024 Asia Reviewers and Organizers
- Phil Young and Speaker-Coaching Program
- Stuart Frankel

Thanks

- John Aycott and Hartmut Goebel
- BlackHat 2024 Asia Reviewers and Organizers
- Phil Young and Speaker-Coaching Program
- Stuart Frankel
- You

Additional Information

- Python Decompilers: <https://pypi.org/project/uncompyle6>, and <https://pypi.org/project/decompyle3/>
- Cross-Version Python Disassembler: <https://pypi.org/project/xdis>
- Python Control Flow: <https://github.com/rocky/python-control-flow>
- Older Decompiler Paper: <https://rocky.github.io/Deparsing-Paper.pdf>
- PyCon Columbia 2018 talk slides: <https://rocky.github.io/pycon2018-light.co>
- PyCon Columbia 2018 talk video: <https://www.youtube.com/watch?v=bRQr1OroXUM&feature=youtu.be>
- Additional Slides: <https://rocky.github.io/blackhat-asia/2024-additional>
- Slide Text: <https://rocky.github.io/blackhat-asia-2024-additional/all-notes-print>