Reverse Engineering with Z3

Sometimes you just want to compute backward

Let’s Get Ready

Pick one of the following ways to do today’s lab

1. Use Kali built by Fweefwop
2. “ssh level1@linux.fweefwop.club” password: linux
3. Install in your own Linux (Ubuntu, Debian, CentOs)

```
sudo apt-get update
sudo apt-get -y install python3-pip
sudo pip3 install z3-solver
```

What’s the Problem? (I) Do my school work

Example 1: the hypotenuse of right-angled triangle has length of 257. If the lengths of the all sides are integers, how long are the other 2 sides?

Example 2: Solve for x and y

$2x^2+ 3y = 269,$

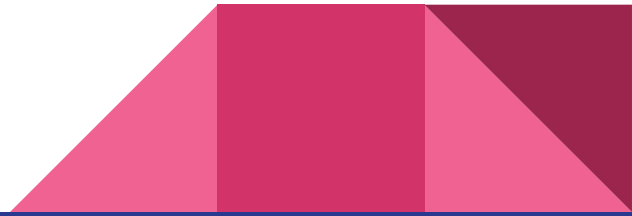
$5x+4y^2 = 2181$

What's the Problem? (II) Solve CTF problems for glory

```
guess = input()

// a humongous sequence of operations that nobody want to
solve by hand>

encrypted_flag = humongous_operations(guess)

If encrypted_flag == <some constant>:

   print("You got it!")
```

Z3 Solver/Prover

- A project from Microsoft Research
- It works like a black magic, most computer science students can’t figure out how it works. (You are welcomed to study it)
- The original interface use the LISP syntax (not so popular)
- It has a Python binding (z3py), so you can just treat it as a Python library. that’s easy!



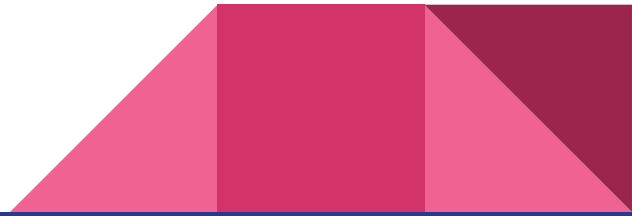
Let’s Start

```
┌──(kali㉿kali)-[~]
└─$ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from z3 import *
>>>
```

API Doc: https://z3prover.github.io/api/html/namespacez3py.html

Declare Integer Variables

```
>>> x = Int('x')
>>> y = Int('y')
>>> solve(x*x+y*y == 257**2)
[x = -255, y = 32]
>>> solve(x*x+y*y == 257**2, x>0, y>0)
[y = 255, x = 32]
>>> solve(x*x+y*y == 257**2, x>0, y>0, x>y)
[x = 255, y = 32]
```

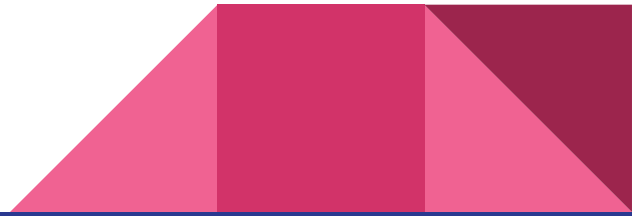


Now You do one

Solve for x and y (both are integers)

$2x^2+ 3y = 269,$

$5x+4y^2 = 2181$



Z3 can handle Real Numbers too

```
>>> x = Real('x')
>>> y = Real('y')
>>> solve(x**2 + y**2 == 3, x**3 == 2)
[y = -1.1885280594?, x = 1.2599210498?]
>>>
```

Boolean Variables

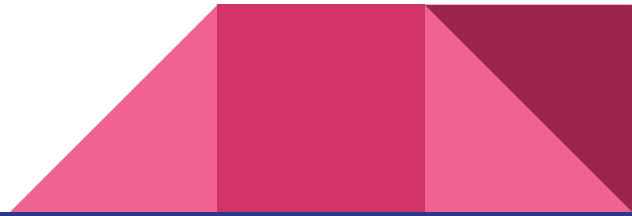
```
>>> p = Bool('p')
>>> q = Bool('q')
>>> simplify(Or(And(p, Not(p)), Not(Or(Not(p),Not(q)))))
Not(Or(Not(p), Not(q)))
>>> simplify(Or(And(p, Not(p)), Not(Or(Not(r),r))))
False
```

Machine Arithmetic (I)

```
>>> a = BitVec('a', 16)

>>> b = BitVec('b', 16)

>>> solve(a == 65535, b == a+2)

[b = 1, a = 65535]
```

Machine Arithmetic (2)

```
>>> solve(a*a*a==2)
no solution
>>> solve(a*a*a==3)
[a = 61819]
>>> solve(a*a*a==5)
[a = 20061]
>>> solve(a == 65535, b == a+2)
[b = 1, a = 65535]
```



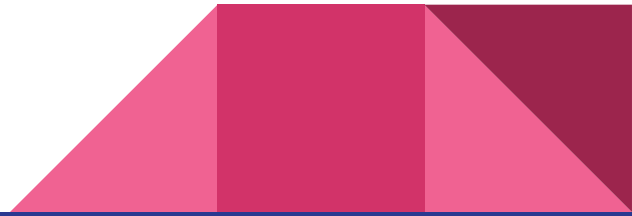
Solver

```
>>> x = Int('x')
>>> y = Int('y')
>>> s = Solver()
>>> s.add(2*x*x + 3*y == 269)
>>> s.add(5*x+4*y*y == 2181)
>>> s.check()
sat
>>> s
[2*x*x + 3*y == 269, 5*x + 4*y*y == 2181]
>>> s.model()
[x = 13, y = -23]
>>> s.model().eval(x)
13
```

Deal with a lot of Variables

Find 100 distinct integers ranges from 0 to 99, ordered descendingly

```
>>> vars = [Int('x_%d' % (i,)) for i in range(100)]
>>> s = Solver()
>>> for i in range(100):
...   s.add(vars[i] >= 0)
...   s.add(vars[i] < 100)
...
>>> for i in range(1,100):
...   s.add(vars[i]< vars[i-1])

>>> s.check()
sat
>>> s.model()
>>> s.model().eval(vars[0])
```



Labs

- Normal:
  - Solution: https://www.k3rn3l4rmy.com/writeup?id=87

- “Fwop Door” on ctf.fweefwop.club
  - Solution: https://pastebin.com/yCg3k0am