Sage Days 16: Elliptic Curves

William Stein

Sage can create curves over general base fields in various ways

1. finite fields
2. symbolics
3. $p$-adics $\mathbb{Q}_p$
4. rationals
5. number fields
6. Tate curve
7. Using Cremona's databases
8. Using the Stein-Watkins database

We create elliptic curves in a few ways:

- \( \text{EllipticCurve}(\text{GF}(5),[1,2,3,4,5]) \)
  - Elliptic Curve defined by \( y^2 + x*y + 3*y = x^3 + 2*x^2 + 4*x \) over Finite Field of size 5

- \( \text{EllipticCurve}([1/2,3/4]) \)
  - Elliptic Curve defined by \( y^2 = x^3 + 1/2*x + 3/4 \) over Rational Field

- \( \text{EllipticCurve_from_j}(1) \)
  - Elliptic Curve defined by \( y^2 + x*y = x^3 + 36*x + 3455 \) over Rational Field

- \( \text{EllipticCurve}([-1+O(3^{10}), 2/3 + O(3^{10}))] \)
  - Elliptic Curve defined by \( y^2 = x^3 + (2+2*3+2*3^2+2*3^3+2*3^4+2*3^5+2*3^6+2*3^7+2*3^8+2*3^9+O(3^{10}))*x \)
    over 3-adic Field with capped relative precision

- \( \text{EllipticCurve}('389a') \)
  - Elliptic Curve defined by \( y^2 + y = x^3 + x^2 - 2*x \) over Rational Field

- \( \text{EllipticCurve_from_cubic('x^3 + y^3 + z^3', [1,-1,0])} \)
Sage can plot elliptic curves over $\mathbb{Q}$ and finite fields

**Example:** Elliptic curves for kids example

```python
@interact
def f(label='37a', p=tuple(prime_range(1000))):
    try: E = EllipticCurve(label)
    except:
        print "invalid label %s"%label; return
    try:
        show(graphics_array([[plot(E,thickness=3),plot(E.change_ring(GF(p)))]])
    except Exception, msg:
        print msg
```

<table>
<thead>
<tr>
<th>label</th>
<th>37a</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td></td>
</tr>
</tbody>
</table>
Sage implements the group law

Example: This is an example of adding two points on a curve.

```python
E = EllipticCurve('389a'); P, Q = E.gens()
sum([plot(s, pointsize=50, color='red') for s in [P, Q, P+Q]]) + plot(E, xmax=5)
```
**Example:** This is a graphical example involving taking many multiples of one point.

```python
%time
E = EllipticCurve('389a'); P, Q = E.gens()
G = sum([plot(n*P, pointsize=5*sqrt(n^2*P.height()), color='red')
         for n in [1..35]]) + plot(E)
show(G, xmax=3, ymax=5, ymin=-5, frame=True)
print "Area is proportional to canonical height."
```
Sage also implements formal group laws

**Example:** This is an example of computing the formal group law on an elliptic curve.

```
E = EllipticCurve('11a'); F = E.formal_group(); F

Formal Group associated to the Elliptic Curve defined by y^2 + y = x^3 - x^2 - 10*x - 20 over Rational Field
```

```
show(F.group_law(3))
```

\[t_1 + O(t_1^3) + (1 + t_1^2 + O(t_1^3)) t_2 + (t_1 - 3t_1^2 + O(t_1^3)) t_2^2 + O(t_2^3)\]

```
show(F.differential(10))
```

\[1 - t^2 + 2t^3 - 19t^4 - 6t^5 + 5t^6 - 108t^7 + 691t^8 + 200t^9 + O(t^{10})\]

```
show(F.sigma(7))
```

\[t + \left(\frac{1}{2}c - \frac{1}{3}\right) t^3 + \frac{1}{2} t^4 + \left(\frac{1}{8}c^2 - \frac{1}{2}c - \frac{143}{36}\right) t^5 + \left(\frac{3}{4}c - 1\right) t^6 + O(t^7)\]

Sage can compute all the standard elliptic curve invariants

**Example:** We compute all the standard algebraic invariants of a curve over a function field.

```
E = EllipticCurve([1..5])
E.a_invariants()
[1, 2, 3, 4, 5]
```

```
E.b_invariants()
(9, 11, 29, 35)
```

```
E.c_invariants()
```
E.j_invariant()

6128487/10351

E = EllipticCurve([a1,a2,a3,a4,a6]);
(a1, a2, a3, a4, a6)

Elliptic Curve defined by y^2 + a1*x*y + a3*y = x^3 + a2*x^2 + a4*
+ a6 over Symbolic Ring

show(E.j_invariant())

\[
\frac{(a_1^2 + 4a_2)^2 - 24a_1a_3 - 48a_4^3}{(9(a_3^2 + 4a_6)(a_1a_3 + 2a_4)(a_1^2 + 4a_2) - 27(a_3^2 + 4a_6)^2 - 8(a_1a_3 + 2a_4)^3 - (a_1^2 + 4a_2)^2(a_1^2a_6 - a_1^3a_4 + a_2^2a_3^2 + 4a_2a_6 - a_4^2)}
\]

E.b_invariants()

(a1^2 + 4*a2, a1*a3 + 2*a4, a3^2 + 4*a6, a1^2*a6 - a1*a3*a4 + a2*a3^2 + 4*a2*a6 - a4^2)

---

Sage can find all isomorphisms between curves

**Example:** We explicitly find an isomorphism between two curves that are quadratic twists of each other.

E = EllipticCurve('37a'); F = E.quadratic_twist(-17);
E.is_isomorphic(F)

False

K.<a> = QuadraticField(-17); E = E.change_ring(K); F =
F.change_ring(K); E.is_isomorphic(F)

True

E.isomorphism_to(F)

**Generic morphism:**

From: Abelian group of points on Elliptic Curve defined by y^2 +
= x^3 + (-1)*x over Number Field in a with defining polynomial x^2
17

To: Abelian group of points on Elliptic Curve defined by y^2 =
x^3 + (-4624)*x + (-78608) over Number Field in a with defining polynomial x^2 + 17
Via:  (u,r,s,t) = (1/34*a, 0, 0, -1/2)

E.isomorphisms(F)

[Generic morphism:
 From: Abelian group of points on Elliptic Curve defined by y^2 +
 = x^3 + (-1)*x over Number Field in a with defining polynomial x^2
17
 To:   Abelian group of points on Elliptic Curve defined by y^2 =
x^3 + (-4624)*x + (-78608) over Number Field in a with defining
polynomial x^2 + 17
 Via:  (u,r,s,t) = (1/34*a, 0, 0, -1/2), Generic morphism:
 From: Abelian group of points on Elliptic Curve defined by y^2 +
 = x^3 + (-1)*x over Number Field in a with defining polynomial x^2
17
 To:   Abelian group of points on Elliptic Curve defined by y^2 =
x^3 + (-4624)*x + (-78608) over Number Field in a with defining
polynomial x^2 + 17
 Via:  (u,r,s,t) = (-1/34*a, 0, 0, -1/2)]

This would be a good project for somebody at Sage Days 16!

phi = E.isomorphism_to(F); phi(E.gens()[0])

Traceback (click to the left for traceback)
...
NotImplementedError: not implemented.

---

Sage can compute images of Galois representations

**Example:** We list the primes \( p \) where the mod-\( p \) representation on a particular elliptic curve isn't surjective.

\[
E = EllipticCurve('11a'); E.non_surjective(100)
\]
Example: We check that for non-CM curves of conductor $< 100$ irreducible is the same as surjective.

```python
for E in cremona_optimal_curves([1..100]):
    v = E.non_surjective(100)
    print E.cremona_label(), v
    for p, _ in v:
        if p and E.is_irreducible(p):
            print '**', E.cremona_label(), p
```

| 11a1  | [(5, '5-torsion')] |
| 14a1  | [(2, '2-torsion'), (3, '3-torsion')] |
| 15a1  | [(2, '2-torsion')] |
| 17a1  | [(2, '2-torsion')] |
| 19a1  | [(3, '3-torsion')] |
| 20a1  | [(2, '2-torsion'), (3, '3-torsion')] |
| 21a1  | [(2, '2-torsion')] |
| 24a1  | [(2, '2-torsion')] |
| 26a1  | [(3, '3-torsion')] |
| 26b1  | [(7, '7-torsion')] |
| 27a1  | [(0, 'cm')] |
| 30a1  | [(2, '2-torsion'), (3, '3-torsion')] |
| 32a1  | [(0, 'cm')] |
| 33a1  | [(2, '2-torsion')] |
| 34a1  | [(2, '2-torsion'), (3, '3-torsion')] |
| 35a1  | [(3, '3-torsion')] |
| 36a1  | [(0, 'cm')] |
| 37a1  | [] |
| 37b1  | [(3, '3-torsion')] |
| 38a1  | [(3, '3-torsion')] |
| 38b1  | [(5, '5-torsion')] |
| 39a1  | [(2, '2-torsion')] |
| 40a1  | [(2, '2-torsion')] |
| 42a1  | [(2, '2-torsion')] |
| 43a1  | [] |
| 44a1  | [(3, '3-torsion')] |
| 45a1  | [(2, '2-torsion')] |
| 46a1  | [(2, '2-torsion')] |
| 48a1  | [(2, '2-torsion')] |
| 49a1  | [(0, 'cm')] |
| 50a1  | [(3, '3-torsion'), (5, [1])] |
| 50b1  | [(3, 'reducible_3-divpoly'), (5, '5-torsion')] |
| 51a1  | [(3, '3-torsion')] |
| 52a1  | [(2, '2-torsion')] |
| 53a1  | [] |
| 54a1  | [(3, '3-torsion')] |
| 54b1  | [(3, '3-torsion')] |
| 55a1  | [(2, '2-torsion')] |
| 56a1  | [(2, '2-torsion')]] |
56b1 [(2, '2-torsion')]
57a1 []
57b1 [(2, '2-torsion')]
57c1 [(5, '5-torsion')]
58a1 []
58b1 [(5, '5-torsion')]
61a1 []
62a1 [(2, '2-torsion')]
63a1 [(2, '2-torsion')]
64a1 [(0, 'cm')]
65a1 [(2, '2-torsion')]
66a1 [(2, '2-torsion'), (3, '3-torsion')]
66b1 [(2, '2-torsion')]
66c1 [(2, '2-torsion'), (5, '5-torsion')]
67a1 []
69a1 [(2, '2-torsion')]
70a1 [(2, '2-torsion')]
72a1 [(2, '2-torsion')]
73a1 [(2, '2-torsion')]
75a1 [(5, [1])]
75b1 [(2, '2-torsion')]
75c1 [(5, '5-torsion')]
76a1 []
77a1 []
77b1 [(3, '3-torsion')]
77c1 [(2, '2-torsion')]
78a1 [(2, '2-torsion')]
79a1 []
80a1 [(2, '2-torsion')]
80b1 [(2, '2-torsion'), (3, 'reducible_3-divpoly')]
82a1 [(2, '2-torsion')]
83a1 []
84a1 [(2, '2-torsion'), (3, '3-torsion')]
84b1 [(2, '2-torsion')]
85a1 [(2, '2-torsion')]
88a1 []
89a1 []
89b1 [(2, '2-torsion')]
90a1 [(2, '2-torsion'), (3, '3-torsion')]
90b1 [(2, '2-torsion'), (3, '3-torsion')]
90c1 [(2, '2-torsion'), (3, 'reducible_3-divpoly')]
91a1 []
91b1 [(3, '3-torsion')]
92a1 [(3, '3-torsion')]
92b1 []
94a1 [(2, '2-torsion')]
96a1 [(2, '2-torsion')]
96b1 [(2, '2-torsion')]
98a1 [(2, '2-torsion'), (3, 'reducible_3-divpoly')]
99a1 [(2, '2-torsion')]
99b1 [(2, '2-torsion')]
99c1 [(2, '2-torsion')]  
99d1 [(5, [1])]  
100al [(2, '2-torsion'), (3, 'reducible_3-divpoly')]

**Example:** We construct an elliptic curve with non-surjective irreducible mod 2 representation.

```python
K.<z> = CyclotomicField(7)  
(z + 1/z).minpoly()  
x^3 + x^2 - 2*x - 1  
E = EllipticCurve([0,1,0,-2,-1]); E  
Elliptic Curve defined by y^2 = x^3 + x^2 - 2*x - 1 over Rational Field  
E.non_surjective()  
[(2, 'A3'), (3, 'reducible_3-divpoly')]  
E.is_irreducible(2)  
True  
E.conductor()  

Sage can compute torsion and division points

**Example:** We compute a big division polynomial.

```python
E = EllipticCurve('37a'); show(E.division_polynomial(5))  
\[5x^{12} - 62x^{10} + 95x^9 - 105x^8 - 60x^7 + 285x^6 - 174x^5 - 5x^4 - 5x^3 + 35x^2 - 15x + 2\]

time f = E.division_polynomial(100)  
Time: CPU 6.83 s, Wall: 7.01 s  
len(str(f)), f.degree()  
(4391928, 5001)

The above is not fast enough -- project for Sage Days 16!

%magma
E := EllipticCurve("37a");
time f := DivisionPolynomial(E,100);
   Time: 0.720

**Example:** We compute the 3-division points of a point.

E = EllipticCurve('37a'); P = E([0,0])

(3*P).division_points(3)
   [(0 : 0 : 1)]

Q = E.change_ring(GF(13))(P); Q.division_points(3)
   [(10 : 11 : 1)]

**Example:** We compute the torsion subgroup an elliptic curve over a number field.

E = EllipticCurve('11a').change_ring(CyclotomicField(5))
time E.torsion_subgroup()
   Torsion Subgroup isomorphic to Multiplicative Abelian Group isomorphic to C5 x C5 associated to the Elliptic Curve defined by y^2 + y = x^3 + (-1)*x^2 + (-10)*x + (-20) over Cyclotomic Field c order 5 and degree 4
   Time: CPU 3.86 s, Wall: 3.96 s

Magma is hundreds of times faster -- fixing this would be a good project for somebody at Sage Days 16!

F = magma(E); magma.eval('time print
   TorsionSubgroup(%s);' % F.name())
   'Abelian Group isomorphic to Z/5 + Z/5\nDefined on 2
generators\nRelations:\n5*$.1 = 0\n5*$.2 = 0\nMapping from: Abelia
group isomorphic to Z/5 + Z/5\nDefined on 2
generators\nRelations:\n5*$.1 = 0\n5*$.2 = 0 to Elliptic Curve
defined by y^2 + y = x^3 - x^2 - 10*x - 20 over Cyclotomic Field c order 5 and degree 4 given by a rule [no inverse]\nTime: 0.020'
Sage can compute the group of points on elliptic curves over finite field

**Example:** We count points over \( \mathbb{F}_p \) for a large \( p \).

```
    time
    EllipticCurve('37a').change_ring(GF(next_prime(10^50))).cardinality()
    100000000000000000000000001917684156174529696959920
    Time: CPU 0.06 s, Wall: 2.71 s
```

**Example:** We compute the group, in an example in which the \( j \) invariant is in \( \mathbb{F}_p \) but the curve is defined over a much bigger field.

```
E = EllipticCurve(GF(7), [2,3])
K.<b> = GF(7^50); F = E.change_ring(K).quadratic_twist(1/(1+b)); F

Elliptic Curve defined by y^2 = x^3 +
(4*b^49+5*b^48+2*b^47+4*b^46+2*b^45+3*b^44+4*b^43+2*b^42+b^41+4*b^*
+2*b^38+2*b^37+3*b^36+6*b^35+3*b^34+4*b^33+b^32+2*b^31+2*b^30+b^29*
*b^28+2*b^27+3*b^26+4*b^25+3*b^24+b^23+3*b^22+b^21+4*b^20+4*b^19+2*b^18+2*b^17+6*b^*
6+5*b^15+4*b^14+5*b^13+4*b^12+5*b^11+6*b^10+6*b^9+6*b^8+2*b^7+4+b^*
3*b^5+6*b^4+6*b^3+2*b^2+3*b+5) over Finite Field in b of size 7^50
```

```
F.j_invariant()
5
time F.cardinality()
17984650426471246617966551185652227001000
Time: CPU 0.07 s, Wall: 0.07 s
```

Sage can also (often) compute the group of points on elliptic curves over rational numbers

**Example:** We compute a mordell-weil group using mwrank:

```
D=6611719866; E = EllipticCurve([0,0,0,-D^2,0])

```
[(24742194842066/37249 : 373863724821481185720/7189057 : 1),
 (165541824817/16 : 51806810701954601/64 : 1),
 (15062000534642656 : 1),
 (548503784857/36 : -365985935192610019/2161),
 (11638545941238203281/246490000 : 39314069377271931544287972679/3869893000000 : 1),
 (5141360778580924481278/169697035249 : -3686515685976763513664298941602072/69905505791578807 : 1)]
```

Time: CPU 0.02 s, Wall: 13.75 s

**Example:** We compute a mordell-weil group using simon's program:

```
E = EllipticCurve([0, 0, 1, -23737, 960366]);
E.simon_two_descent()
```

```
 (36 : 390 : 1)])
```

---

**Sage can also (sometimes) compute the group of points on curve over number fields**

**Example:** We compute a mordell-weil group over a quadratic imaginary field using simon's program:

```
K.<a> = NumberField(x^2 + 23); E = EllipticCurve(K, '37')
E.simon_two_descent()
```

```
(2, 2, [(-1 : 0 : 1), (1/2*a - 5/2 : -1/2*a - 13/2 : 1)])
```

Note, `E.rank()`, `E.gens()`, etc., don't work over number fields. It would be a good Sage Days 16 project to make them all work, at least with a proof=False option, in case Simon's code never has `proof=True`.

---

**Sage can enumerate integral points and S-integral points**
Example: We compute the s-integral points of an elliptic curve... (that rank 6 example that kills magma)

```
E = EllipticCurve('389a'); time E.integral_points()
[(-2 : 0 : 1), (-1 : 1 : 1), (0 : 0 : 1), (1 : 0 : 1), (3 : 5 : 1),
 (4 : 8 : 1), (6 : 15 : 1), (39 : 246 : 1), (133 : 1539 : 1), (188 : 2584 : 1)]
Time: CPU 0.38 s, Wall: 3.00 s
```

```
time E.S_integral_points([2,3])
[(-2 : 0 : 1), (-1364/729 : 9269/19683 : 1), (-95/64 : 495/512 : 1),
 (-11/9 : 28/27 : 1), (-1 : 1 : 1), (-3/4 : 7/8 : 1), (0 : 0 : 1),
 (5/4 : 5/8 : 1), (2353/1296 : 89999/46656 : 1), (3 : 5 : 1),
 (8 : 1), (6 : 15 : 1), (39 : 246 : 1), (133 : 1539 : 1), (188 : 2584 : 1)]
Time: CPU 1.84 s, Wall: 2.80 s
```

---

**Sage can compute the Weil pairing**

Example: We compute a Weil pairing over a finite field.

```
K.<zeta5> = CyclotomicField(5)
P,Q = EllipticCurve(K,'11a1').torsion_subgroup().gens()
P.weil_pairing(Q,5)
zeta5^2
(2*P).weil_pairing(Q,5)
  -zeta5^3 - zeta5^2 - zeta5 - 1
(zeta5^2)^2
  -zeta5^3 - zeta5^2 - zeta5 - 1
```

---

**Sage can compute twists: quadratic, cubic, quartic, sextic**
Example: We construct a sextic twist of a curve over the rationals.

```
E = EllipticCurve('27a'); F = E.sextic_twist(5); F
Elliptic Curve defined by y^2 = x^3 - 1574640 over Rational Field
E.q_eigenform(20)
q - 2*q^4 - q^7 + 5*q^13 + 4*q^16 - 7*q^19 + O(q^20)
F.q_eigenform(20)
q - 2*q^4 - 5*q^7 - 5*q^13 + 4*q^16 - q^19 + O(q^20)
E.j_invariant(), F.j_invariant()
(0, 0)
```

Note that over $\mathbb{Q}(\sqrt{-3})$ our curve has automorphism group of order 6:

```
K.<a> = QuadraticField(-3)
len(E.change_ring(K).isomorphisms(E.change_ring(K)))
6
```

---

### Sage can compute the conductor, reduction type (Kodaira symbols), and Tamagawa numbers

Example: We compute the conductor of a curve over a number field.

```
E = EllipticCurve('27a');
K.<a> = QuadraticField(-3); E.change_ring(K).conductor()
Fractional ideal (9)
K.<a> = QuadraticField(7); E.change_ring(K).conductor()
Fractional ideal (27)
K.<zeta5> = CyclotomicField(5)
E = EllipticCurve([1,zeta5+2]); E.j_invariant()
-2988036864/73774381*zeta5^3 + 1871465472/73774381*zeta5^2 - 5171724288/73774381*zeta5 + 2843099136/73774381
N = E.conductor(); N
Fractional ideal (680*zeta5^3 + 32*zeta5^2 - 216*zeta5 + 32)
```
F = N.factor(); F
(Fractional ideal (2))^3 * (Fractional ideal (85*zeta5^3 + 4*zeta5 - 27*zeta5 + 4))

Example: We compute a Tamagawa number over a number field.

E.tamagawa_number(F[0][0])
2
E.tamagawa_number(F[1][0])
1
E.kodaira_symbol(F[0][0])
III

Sage can compute the period lattice and real period omega

Example: We compute the period lattice of an elliptic curve.

M = EllipticCurve('11a').period_lattice(); M
Period lattice associated to Elliptic Curve defined by y^2 + y = x - x^2 - 10*x - 20 over Rational Field

M.basis()
(1.26920930427955, 0.634604652139775 + 1.45881661693850*I)

M.omega()
1.26920930427955

w = M.basis()
points([a*w[0]+b*w[1] for a in [-4..4] for b in [-4..4]]).show(frame=True)
Sage can (often) compute Archimedean regulators

**Example:** We compute the regulator of the rank 2 curve 389a.

```python
E = EllipticCurve('389a'); E.regulator()
0.152460177943144

E.regulator(precision=200)
0.152460177943143751624372707748663081784028
```

Sage can (often) compute Shafarevich-Tate
groups

**Example:** We show that a certain Shafarevich-tate group is trivial.

```python
E = EllipticCurve('37a'); S = E.sha(); S
```

Shafarevich-Tate group for the Elliptic Curve defined by $y^2 + y = x^3 - x$ over Rational Field

```python
S.bound_kolyvagin()
```

([2], 1)

```python
S.two_selmer_bound()
```

0

**Example:** We show that the $p$-part of the Shafarevich-Tate group of a rank 2 curve is trivial for several primes for which $p$-descent is not practical.

```python
E = EllipticCurve('389a'); S = E.sha(); S
```

Shafarevich-Tate group for the Elliptic Curve defined by $y^2 + y = x^3 + x^2 - 2*x$ over Rational Field

```python
for p in [5, 7, 11, 13, 17]:
    print p, S.p_primary_bound(p)
```

5 0
7 0
11 0
13 0
17 0

---

**Sage can compute with complex $L$-series**

- Evaluation without any provable error bound using Dokchitser's algorithm
- Evaluation with provable error bound at 1 using summation of series
- Fast enumeration of zeros in the critical strip using Rubinstein's lcalc
- Evaluation of symmetric powers of elliptic curve $L$-series using Watkins's sympow.

**Example:** We evaluate each of the above for a rank 2 curve.

```python
E = EllipticCurve('389a')
L = E.lseries(); L(1+I)   # uses Dokchitser
```

-0.638409938588039 + 0.715495239204667*I
Sage can compute with $p$-adic $L$-functions

**Example:** We compute explicitly the $p$-adic $L$-series associated to an elliptic curve with ordinary reduction at $p$.

```python
E = EllipticCurve('446d1'); E.supersingular_primes(20)
[3, 19]
L5 = E.padic_lseries(5); L5
5-adic L-series of Elliptic Curve defined by y^2 + x*y = x^3 - x^2 + 4*x + 4 over Rational Field
show(L5.series(3))
O(5^5) + O(5^3)T + (5 + O(5^2)) T^2 + (2*5 + O(5^2)) T^3 + O(5^2) T^4 + O(T^5)
```

E.rank()
2

**Example:** We compute explicitly the $p$-adic $L$-series associated to an elliptic curve with supersingular reduction at $p$. 

```python
L.at1(100)
(6.12567637694320e-15, 7.78911206115050e-14)
L.L_ratio()  # proves that L(1) = 0
0
L.zeros(10)
[0.000000000, 0.000000000, 2.87609907, 4.41689608, 5.79340263, 6.98596665, 7.47490750, 8.63320525, 9.63307880, 10.3514333]
print L.sympow(2,16)
Inertia Group is  C1 MULTIPLICATIVE REDUCTION
Conductor is 389
**ERROR** P02L not found in param_data file
It can be added with './sympow -new_data 2'
sympow.new_data(2)
Make data for symmetric power 2
```
Sage can compute $p$-adic height pairings

**Example:** We compute the $p$-adic regulator of a rank 2 curve at a large prime. This was impossible a few years ago.

```python
E = EllipticCurve('446d1'); E.gens()
[(-1 : 3 : 1), (2 : -2 : 1)]
```

```python
time E.padic_regulator(97, prec=10)
47*97^2 + 57*97^3 + 19*97^4 + 2*97^5 + 32*97^6 + 46*97^7 + 30*97^8
17*97^9 + 42*97^10 + O(97^11)
Time: CPU 0.21 s, Wall: 0.29 s
```

```python
time E.padic_regulator(10007, prec=5)
1788*10007^2 + 9818*10007^3 + 2383*10007^4 + 133*10007^5 +
O(10007^6)
Time: CPU 0.23 s, Wall: 0.31 s
```

```python
time E.padic_regulator(65521, prec=5)
53131*65521^2 + 17307*65521^3 + 3589*65521^4 + 36837*65521^5 +
O(65521^6)
Time: CPU 0.68 s, Wall: 0.99 s
```

Sage can compute the modular degree

**Example:** We compute the modular degree of a rank 4 curve, something that was very difficult using all other algorithms a few years ago...

```python
E = elliptic_curves.rank(4)[0]; E
Elliptic Curve defined by y^2 + x*y = x^3 - x^2 - 79*x + 289 over
Rational Field

time E.modular_degree()

334976
Time: CPU 0.01 s, Wall: 1.75 s

factor(334976)

2^7 * 2617

Sage can compute isogenies between elliptic curves

Example: We display an isogeny class, and compute the manin constant of a non-optimal curve.

E = EllipticCurve('220a'); show(E.isogeny_graph(),figsize=2)

EllipticCurve('220a4').manin_constant()

6

Summary
• Sage can compute a huge amount with elliptic curves.

• The functionality is still rough around the edges and substantial polish is needed. Please report bugs!!

• Somebody (e.g., Robert Miller?), please implement $n$-descent for various $n$, since that is the biggest gap in functionality at present.

• There is *substantial room* for optimization. E.g., points are pure Python classes, and some basic arithmetic is much slower than in Magma or Pari.