In [1]: 1 + 1
Out[1]: 2

In [5]: 2 * 3
Out[5]: 6

In [17]: pi.numerical_approx(digits=1)
Out[17]: 3.2

In [ ]:

In [9]: a = 2**312 + 1
   a.is_prime()

Out[9]: False

In [10]: # unlimited arithmetic
   2**5000

Out[10]: 14124670321394260368352096670161473336688961751845411168136880858571181698

In [ ]:

In [25]: f(x) = cos(x) * sin(x)
g(x) = x - 1

In [26]: P1 = plot(f, (x,0,2*pi))
   P2 = plot(g, (x,0,2*pi), color='red')

In [27]: # superposition of two graphics
   P1 + P2

Out[27]:
In [28]: # exercise 1: plot the set of primitive vectors

In [37]: 

    # (x: x in S)
    # range(10) = [0,1,...,9]
    pts = [(i,j) for i in range(100) for j in range(100)]
    prims = [p for p in pts if gcd(p) == 1]

In [38]: point2d(prims)

Out[38]:
In [39]: g(a,b,c,d) = sqrt(a^2 + b^2) + sqrt(c^2 + d^2) - sqrt((a+c)^2 + (b+d)^2)

In [40]: g

Out[40]: (a, b, c, d) |--> -sqrt((a + c)^2 + (b + d)^2) + sqrt(a^2 + b^2) + sqrt(c^2 + d^2)

In [42]: #sum(g(p[0],p[1],q[0],q[1]) for p in prims for q in prims)

In [ ]: # question 2: sum over pairs with det=1 ??

In [ ]:

In [46]: # tab completion: list names available
   # question mark: access to documentation
   matrix?

In [49]: m=matrix([[1,1],[0,1]])

In [45]: p = Polyhedron([[(0,0), (1,0), (0,1), (1,1)]])

In [44]: p

Out[44]:

3
A 2-dimensional polyhedron in $\mathbb{Z}^2$ defined as the convex hull of 4 vertices

In [50]: m * p

Out[50]:
In [51]: x = var('x')
y = var('y')

In [71]: # function with values in matrices do not work
   # that's a pity!!
   f(x) = x
   m(x,y) = matrix([[cos(x), sin(x+y)], [x, 1]])

In [73]: def mat(x, y):
   return matrix([[cos(x), sin(x+y)], [x, 1]])

In [75]: mat(pi, pi/2)*p

Out[75]:
In [67]: m(pi/2, pi)

Out[67]:  
[ cos(x) sin(x + y)]   
[ x 1]

In [55]:  
# slow
search_doc('Hamilton cycle')

In [ ]:  
# the command below defines implicitly a new
# variable
f(t) = 2*t

In [ ]:

In [76]: v = vector([14.3, 2.5])

In [79]: v = v.apply_map(floor)
In [79]: (0.300000000000001, 0.500000000000000)

In [80]: v.apply_map(frac)

Out[80]: (0.300000000000001, 0.500000000000000)

In []:

In []:

In [81]: p = iet.Permutation('a b c d', 'd c b a')

In [84]: T = iet.IntervalExchangeTransformation(p, [2, 5, 3, 1])

In [86]: T.plot()

Out[86]:

In [ ]:

In [89]: # additional package

    from surface_dynamics.all import *
In [90]: E = origamis.EierlegendeWollmilchsau()

In [91]: E.plot()

Out[91]:

In [92]: E.veech_group()

Out[92]: Arithmetic subgroup with permutations of right cosets
S2=()
S3=()
L=()
R=()

In [93]: E.lyapunov_exponents_approx()

Out[93]: [0.0000515519136114749, 0.0000447590239006364]

In [94]: E.sum_of_lyapunov_exponents()

Out[94]: 1

In [ ]: