نظام Sage Math للحسابات العلمية

وهو نظام برمجة رياضي من المصدر المفتوح (مجاني) و هو مبني على قمة العديد من البرامج مفتوحة المصدر مثل : NumPy, SciPy, matplotlib,Sympy, Maxima, GAP, FLINT, R و العديد من البرامج الاخرى. والنظام يصل إلى هذه القوة المشتركة من خلال لغة مشتركة معتمدة على لغة Python أو مباشرة من خلال واجهات للمستخدم interfaces أو برامج تغليف wrappers.

وهدف النظام هو خلق بديل من المصدر المفتوح للأنظمة :Magma و Maple و Mathematica و Matlab

SageMath	
Initial release	24 February 2005; 10 years ago
Stable release	6.8 / 26 July 2015; 2 months ago
Preview release	6.8.rc1 / 22 July 2015; 2 months ago
Written in	Python, Cython
Operating system	Cross-platform
Platform	Python

Size	411 MB download (Ubuntu64-bit)[1]
Type	Computer algebra system
License	GNU General Public License
Website	www.sagemath.org,cloud.sagemath.com

ويوجد نظام SageMath من خلال:

1- نظام تشغيل يعتمد على توزيع CentOS والذي يمكن إستخدامه من خلال محرك USB.

2- من خلال VirtualBox خاصة لنظام التشغيل Windows.

3- من خلال SageCloud من الموقع https://cloud.sagemath.com .

4- من خلال الموقع https://sagecell.sagemath.org .

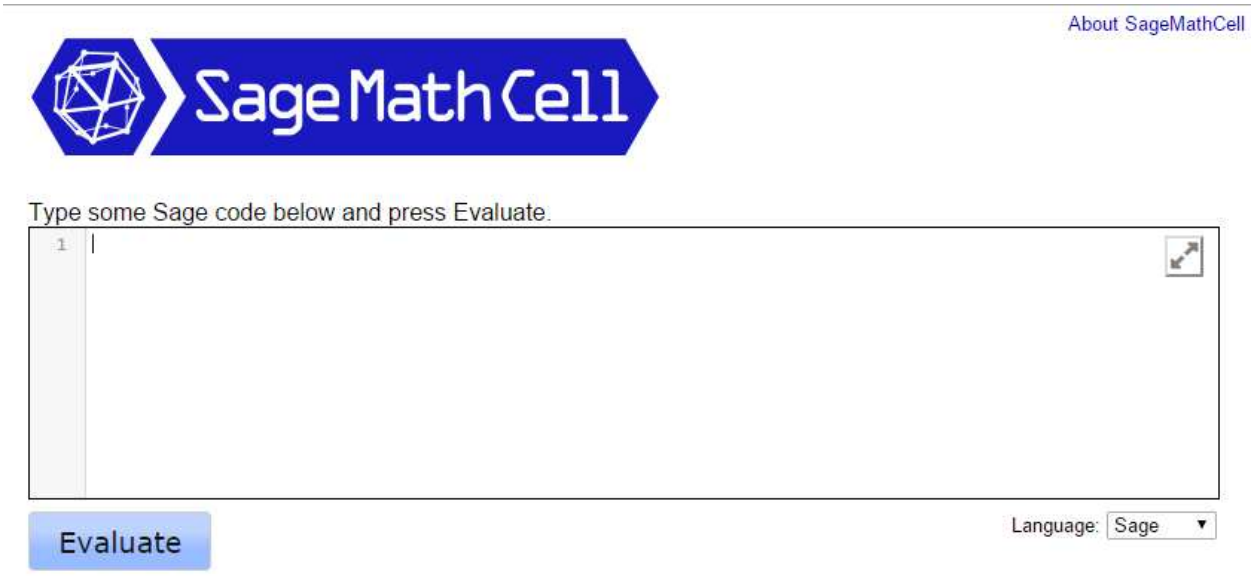
نظرة اولى على نظام SageMath:

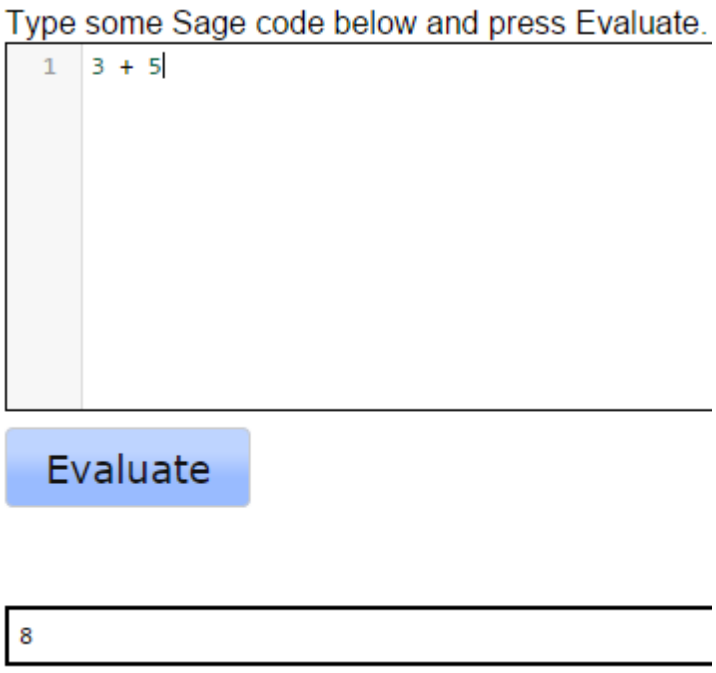
لغرض تعلم العمل مع النظام سوف نستخدم الطريقة 4 لإستعراض بعض مقدرات SageMath التي تهم الإحصاء و بحوث العمليات.

في أي متصفحة ندخل العنوان

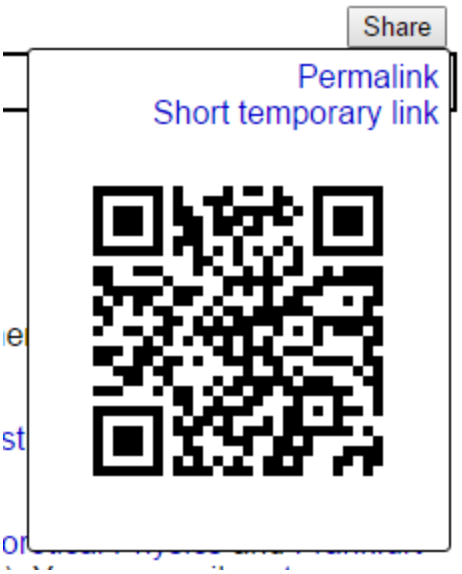
https://sagecell.sagemath.org/

فتظهر الصفحة:

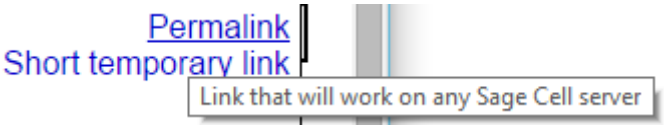




بالضغط على Share يظهر الصندوق

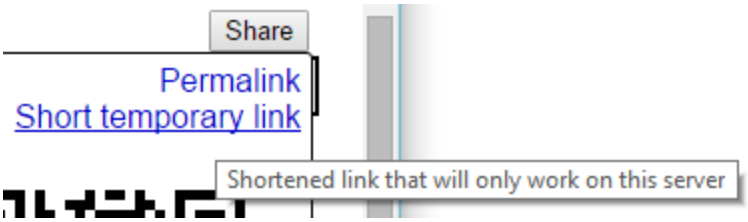


الخيار Permalink



يعطي توصيلة للصفحة و البرنامج المدخل بشكل دائم.

الخيار Short temporary link



يعطي توصيلة قصيرة مؤقتة تنتهي في اي وقت.

الـ QR code

ومن خلالة نسترجع الصفحة ومحتوياتها بإستخدام ماسحة QR و التي تتوفر على جميع الأجهزة الذكية.

أمثلة:

الحسابات البسيطة:

```
sage: 3 + 5
8
```

```
sage: 57.1 ^ 100
4.60904368661396e175
```

عكس مصفوفة

```
sage: matrix([[1,2], [3,4]])^(-1)
[  -2    1]
[ 3/2 -1/2]
```

توليد متغير رمزي symbolic variable وإسناد القيمة x له و إيجاد التكامل

$$\int_x x^2\left(1+x\right)^2\,dx$$

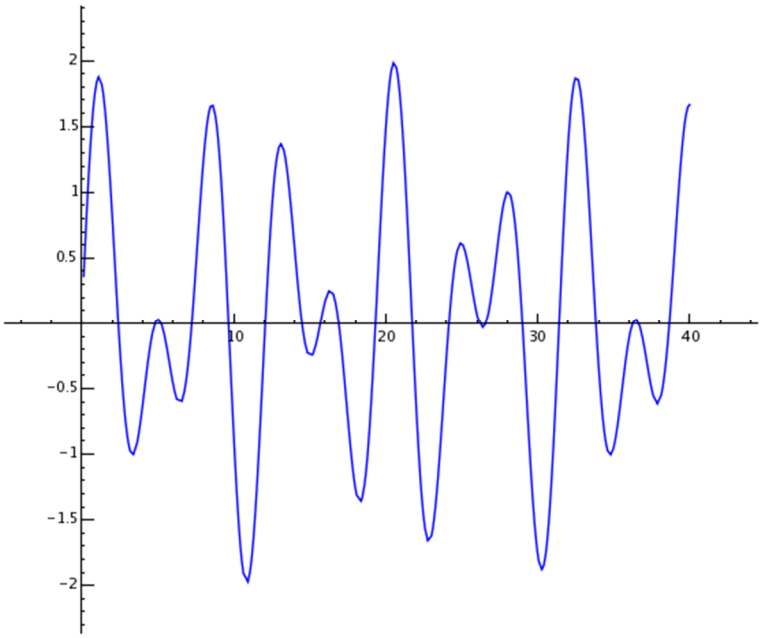
```
sage: x = var('x')
sage: integrate(sqrt(x)*sqrt(1+x), x)
1/4*((x + 1)^(3/2)/x^(3/2) + sqrt(x +
1)/sqrt(x))/((x + 1)^2/x^2 - 2*(x + 1)/x + 1) -
1/8*log(sqrt(x + 1)/sqrt(x) + 1) +
1/8*log(sqrt(x + 1)/sqrt(x) - 1)
```

تعريف متغير a و حل المعادلة $x^2+x=a$

```
sage: a = var('a')
sage: S = solve(x^2 + x == a, x); S
[x == -1/2*sqrt(4*a + 1) - 1/2, x ==
1/2*sqrt(4*a + 1) - 1/2]
sage: S[0].rhs()
-1/2*sqrt(4*a + 1) - 1/2
```

رسم الدالة $\sin\left(x\right)+\sin\left(1.6x\right)$ على الفترة $x\in\left[0,40\right]$

```
sage: show(plot(sin(x) + sin(1.6*x), 0, 40))
```



الحسابات المعقدة:

توليد مصفوفة ذات عناصر من ارقام عشوائية أبعادها 500x500 من حقل الأعداد الحقيقية مضاعفة الدقة Real Double Field (RDF) و إيجاد القيم المميزة (يستغرق Sage حوالي 2 ثانية)

```
sage: m = random_matrix(RDF,500)
sage: e = m.eigenvalues()  #about 2 seconds
```

إيجاد !100 (مضروب 100 اي 100x99x98x...x1 لاحظ ان !1000000 يستغرق حسابة حوالي 2.5 ثانية

```
sage: factorial(100)
933262154439441526816992388562667004907159682643
816214685929638952175999932299156089414639761565
```

```
1828625369792082722375825118521091686400000000000
00000000000000
sage: n = factorial(1000000)
```

نستطيع حساب قيمة π حتى 100 عدد. لاحظ أن N تعني إعطاء النتيجة عدديا.

```
sage: N(pi, digits=100)
3.14159265358979323846264338327950288419716939937
51058209749445923078164062862089986280348253421
17068
```

تعريف المتغيرين x و y على الحقل QQ أي حقل الأعداد الكسرية Rational Field ومن ثم إيجاد المفكوك factor (العوامل الأولية) للدالة $x^{99}+y^{99}$ ثم عكس العملية بتفكيك الدالة مرة اخرى expand

```
sage: R.<x,y> = QQ[]
sage: F = factor(x^99 + y^99)
sage: F
(x + y) * (x^2 - x*y + y^2) * (x^6 - x^3*y^3 +
y^6) *(x^10 - x^9*y + x^8*y^2 - x^7*y^3 +
x^6*y^4 - x^5*y^5 + x^4*y^6 - x^3*y^7 + x^2*y^8
- x*y^9 + y^10) *(x^20 + x^19*y - x^17*y^3 -
```

```
x^16*y^4 + x^14*y^6 + x^13*y^7 - x^11*y^9 -
x^10*y^10 - x^9*y^11 + x^7*y^13 + x^6*y^14 -
 x^4*y^16 - x^3*y^17 + x*y^19 + y^20) * (x^60 +
x^57*y^3 - x^51*y^9 - x^48*y^12 + x^42*y^18 +
x^39*y^21 - x^33*y^27 - x^30*y^30 - x^27*y^33 +
x^21*y^39 + x^18*y^42 - x^12*y^48 - x^9*y^51 +
x^3*y^57 + y^60)
sage: F.expand()
x^99 + y^99
```

أمثلة متنوعة:

ينصح بإدخال الترميز و التأمل في المخرجات

```
sage: factor(-2007)
-1 * 3^2 * 223

sage: A = matrix(4,4, range(16)); A
[ 0  1  2  3]
[ 4  5  6  7]
[ 8  9 10 11]
[12 13 14 15]

sage: factor(A.charpoly())
```

```
x^2 * (x^2 - 30*x - 80)
```

Integer Ring الأعداد الصحيحة ZZ

```
sage: m = matrix(ZZ,2, range(4))
sage: m[0,0] = m[0,0] - 3
sage: m
[-3  1]
[ 2  3]

sage: E = EllipticCurve([1,2,3,4,5]);
sage: E
Elliptic Curve defined by y^2 + x*y + 3*y = x^3
+ 2*x^2 + 4*x + 5
over Rational Field
sage: E.anlist(10)
[0, 1, 1, 0, -1, -3, 0, -1, -3, -3, -3]
sage: E.rank()
1

sage: k = 1/(sqrt(3)*I + 3/4 + sqrt(73)*5/9); k
36/(20*sqrt(73) + 36*I*sqrt(3) + 27)
sage: N(k)
0.165495678130644 - 0.0521492082074256*I
```

```
sage: N(k,30)       # 30 "bits"
0.16549568 - 0.052149208*I
sage: latex(k)
\frac{36}{20 \, \sqrt{73} + 36 i \, \sqrt{3} +
27}
```

الإسناد و المساواة و الحساب:

```
sage: a = 5
sage: a
5
sage: 2 == 2
True
sage: 2 == 3
False
sage: 2 < 3
True
sage: a == 5
True
```

```
sage: 2**3     #  ** means exponent
8
```

```
sage: 2^3      #  ^ is a synonym for ** (unlike in Python)
8
sage: 10 % 3  #  for integer arguments, % means mod, i.e., remainder
1
sage: 10/4
5/2
sage: 10//4   #  for integer arguments, // returns the integer quotient
2
sage: 4 * (10 // 4) + 10 % 4 == 10
True
sage: 3^2*4 + 2%5
38
```

```
sage: sqrt(3.4)
1.84390889145858
sage: sin(5.135)
-0.912021158525540
sage: sin(pi/3)
1/2*sqrt(3)
```

```
sage: exp(2)
e^2
sage: n(exp(2))
7.38905609893065
sage: sqrt(pi).numerical_approx()
1.77245385090552
sage: sin(10).n(digits=5)
-0.54402
sage: N(sin(10),digits=10)
-0.5440211109
sage: numerical_approx(pi, prec=200)
3.14159265358979323846264338327950288419716939937
51058209749
```

```
sage: a = 5   # a is an integer
sage: type(a)
<type 'sage.rings.integer.Integer'>
sage: a = 5/3  # now a is a rational number
sage: type(a)
<type 'sage.rings.rational.Rational'>
sage: a = 'hello'  # now a is a string
sage: type(a)
<type 'str'>
```

```
sage: 011
9
sage: 8 + 1
9
sage: n = 011
sage: n.str(8)   # string representation of n in
base 8
'11'
```

نظام المساعدة في SageMath:

```
sage: tan?
Type:        <class
'sage.calculus.calculus.Function_tan'>
Definition:  tan( [noargspec] )
Docstring:

    The tangent function

    EXAMPLES:
        sage: tan(pi)
        0
```

```
        sage: tan(3.1415)
        -0.00009265359005819 13
        sage: tan(3.1415/4)
        0.999953674278156
        sage: tan(pi/4)
        1
        sage: tan(1/2)
        tan(1/2)
        sage: RR(tan(1/2))
        0.546302489843790
sage: log2?
Type:        <class
'sage.functions.constants.Log2'>
Definition:  log2( [noargspec] )
Docstring:

    The natural logarithm of the real number 2.

    EXAMPLES:
        sage: log2
        log2
        sage: float(log2)
        0.6931471805599453 29
```

```
        sage: RR(log2)
        0.693147180559945
        sage: R = RealField(200); R
        Real Field with 200 bits of precision
        sage: R(log2)

0.69314718055994530941723212145817656807550013436025525412068
        sage: l = (1-log2)/(1+log2); l
        (1 - log(2))/(log(2) + 1)
        sage: R(l)

0.18123221829928249948761381864650311423330609774776013488056
        sage: maxima(log2)
        log(2)
        sage: maxima(log2).float()
        .693147180559453
        sage: gp(log2)
        0.693147180559453094172321215
# 32-bit
        0.693147180559453094172321214581765680
# 64-bit
```

```
sage: sudoku?
File:        sage/local/lib/python2.5/site-
packages/sage/games/sudoku.py
Type:        <type 'function'>
Definition:  sudoku(A)
Docstring:

    Solve the 9x9 Sudoku puzzle defined by the
matrix A.

    EXAMPLE:
        sage: A = matrix(ZZ,9,[5,0,0, 0,8,0,
0,4,9, 0,0,0, 5,0,0,
    0,3,0, 0,6,7, 3,0,0, 0,0,1, 1,5,0, 0,0,0,
0,0,0, 0,0,0, 2,0,8, 0,0,0,
    0,0,0, 0,0,0, 0,1,8, 7,0,0, 0,0,4, 1,5,0,
0,3,0, 0,0,2,
    0,0,0, 4,9,0, 0,5,0, 0,0,3])
        sage: A
        [5 0 0 0 8 0 0 4 9]
        [0 0 0 5 0 0 0 3 0]
        [0 6 7 3 0 0 0 0 1]
        [1 5 0 0 0 0 0 0 0]
```

```
[0 0 0 2 0 8 0 0 0]
[0 0 0 0 0 0 0 1 8]
[7 0 0 0 0 4 1 5 0]
[0 3 0 0 0 2 0 0 0]
[4 9 0 0 5 0 0 0 3]
sage: sudoku(A)
[5 1 3 6 8 7 2 4 9]
[8 4 9 5 2 1 6 3 7]
[2 6 7 3 4 9 5 8 1]
[1 5 8 4 6 3 9 7 2]
[9 7 4 2 1 8 3 6 5]
[3 2 6 7 9 5 4 1 8]
[7 8 2 9 3 4 1 5 6]
[6 3 5 1 7 2 8 9 4]
[4 9 1 8 5 6 7 2 3]
```

الدوال Functions و الميل الداخلي Indentation و العد Counting:

```
sage: def is_even(n):
....:     return n%2 == 0
```

الكلمة المفتاح def تعرف دالة

ميزة Sage و كذلك Python هو تحديد مجاميع الترميز بميل داخلي indentation

```
sage: is_even(2)
```

```
True
sage: is_even(3)
False
```

```
sage: def is_divisible_by(number, divisor=2):
....:     return number%divisor == 0
sage: is_divisible_by(6,2)
True
sage: is_divisible_by(6)
True
sage: is_divisible_by(6, 5)
False
```

```
sage: is_divisible_by(6, divisor=5)
False
sage: is_divisible_by(divisor=2, number=6)
True
```

```
sage: def even(n):
....:     v = []
....:     for i in range(3,n):
....:         if i % 2 == 0:
```

```
....:             v.append(i)
....:     return v
Syntax Error:
        return v
```

الخطأ نتج من الجملة `return v` لم تكن في نفس الميل كـ `even(n):`

```
sage: def even(n):
....:     v = []
....:     for i in range(3,n):
....:         if i % 2 == 0:
....:             v.append(i)
....:     return v
sage: even(10)
[4, 6, 8]
```

الميل الصحيح

```
sage: a = 5; b = a + 3; c = b^2; c
64
```

إدخال عدة جمل في سطر واحد تفصل بين كل جملة و اخرى ; فاصلة منقوطة

```
sage: 2 + \
....:     3
5
```

\ في جملة يعني إكمالها في السطر التالي (لاحظ :....) و التي تعنى طلب البرنامج لإكمال شيئ

```
sage: for i in range(3):
....:     print i
0
1
2
```

الدورة for (لاحظ كيفية إستخدام range)

```
sage: for i in range(2,5):
....:     print i
2
3
4
```

```
sage: for i in range(1,6,2):
....:     print i
1
3
5
```

```
sage: for i in range(5):
```

```
....:     print '%6s %6s %6s'%(i, i^2, i^3)
     0      0      0
     1      1      1
     2      4      8
     3      9     27
     4     16     64
```

```
sage: range(2,10)
[2, 3, 4, 5, 6, 7, 8, 9]
```

تكوين قائمة

```
sage: v = [1, "hello", 2/3, sin(x^3)]
sage: v
[1, 'hello', 2/3, sin(x^3)]
```

العنصر الأول في القائمة يسند لة 0 (لاحظ ان إسناد الأرقام المتسلسلة يبدأ من الصفر)

```
sage: v[0]
1
sage: v[3]
sin(x^3)
```

```
sage: len(v)
4
```

```
sage: v.append(1.5)
```

إضافة عنصر للقائمة

```
sage: v
[1, 'hello', 2/3, sin(x^3), 1.500000000000000]
sage: del v[1]
```

حذف عنصر من القائمة

```
sage: v
[1, 2/3, sin(x^3), 1.500000000000000]
```

تكوين قاموس dictionary (سوف نتطرق لهذا لاحقا)

```
sage: d = {'hi':-2,  3/8:pi,   e:pi}
sage: d['hi']
-2
sage: d[e]
pi
```

المصفوفات و الجبر الخطي:

```
sage: A = Matrix([[1,2,3],[3,2,1],[1,1,1]])
sage: w = vector([1,1,-4])
sage: w*A
(0, 0, 0)
```

تعريف مصفوفة A و متجة w

```
sage: A*w
```

```
(-9, 1, -2)
sage: kernel(A)
Free module of degree 3 and rank 1 over Integer
Ring
Echelon basis matrix:
[ 1  1 -4]
```

```
sage: Y = vector([0, -4, -1])
sage: X = A.solve_right(Y)
sage: X
(-2, 1, 0)
sage: A * X   # checking our answer...
(0, -4, -1)
```

```
sage: A \ Y
(-2, 1, 0)
```

حل التركيب الخطي AX=Y

```
sage: A = matrix([[0, 4], [-1, 0]])
sage: A.eigenvalues ()
[-2*I, 2*I]
sage: B = matrix([[1, 3], [3, 1]])
sage: B.eigenvectors_left()
```

```
[(4, [(1, 1)], 1), (-2, [(1, -1)], 1)]
```

```
sage: AZ = matrix(ZZ, [[2,0], [0,1]])
sage: AQ = matrix(QQ, [[2,0], [0,1]])
sage: AR = matrix(RR, [[2,0], [0,1]])
sage: AZ.echelon_form()
[2 0]
[0 1]
sage: AQ.echelon_form()
[1 0]
[0 1]
sage: AR.echelon_form()
[ 1.00000000000000 0.000000000000000]
[0.000000000000000  1.00000000000000]
```

RR حقل الأرقام الحقيقية بدقة 53 بت Real Field with 53 bits of precision

```
sage: ARDF = matrix(RDF, [[1.2, 2], [2, 3]])
sage: ARDF.eigenvalues()  # rel tol 8e-16
[-0.09317121994613098, 4.293171219946131]
sage: ACDF = matrix(CDF, [[1.2, I], [2, 3]])
sage: ACDF.eigenvectors_right()  # rel tol 3e-15
```

```
[(0.881845698329374 - 0.820914065343413*I,
[(0.750560818380954, -0.616145932704589 +
0.238794153033326*I)], 1),
(3.318154301670626 + 0.820914065343413*I,
[(0.145594698292710 + 0.375669085850210*I,
0.915245825866211)], 1)]
```

CDF حقل الأرقام المركبة دقة ثنائية Complex Double Field

فضاآت المصفوفات Matrix spaces

سوف نولد فضاء $Mat_{3\times3}(Q)$ للمصفوفات ذات الأبعاد 3x3 بحدود كسرية

```
sage: M = MatrixSpace(QQ,3)
sage: M
Full MatrixSpace of 3 by 3 dense matrices over
Rational Field
```

ملاحظة: لتوليد فضاء ذا أبعاد 3x4 مثلا نستخدم MatrixSpace(QQ,3,4)

```
sage: B = M.basis()
sage: len(B)
9
sage: B[1]
[0 1 0]
```

```
[0 0 0]
[0 0 0]
```

```
sage: A = M(range(9)); A
[0 1 2]
[3 4 5]
[6 7 8]
```

```
sage: M = MatrixSpace(GF(2),4,8)
sage: A = M([1,1,0,0, 1,1,1,1, 0,1,0,0, 1,0,1,1,
....:       0,0,1,0, 1,1,0,1, 0,0,1,1,1,1,1,0])
sage: A
[1 1 0 0 1 1 1 1]
[0 1 0 0 1 0 1 1]
[0 0 1 0 1 1 0 1]
[0 0 1 1 1 1 1 0]
sage: rows = A.rows()
sage: A.columns()
[(1, 0, 0, 0), (1, 1, 0, 0), (0, 0, 1, 1), (0,
0, 0, 1), (1, 1, 1, 1), (1, 0, 1, 1), (1, 1, 0,
1), (1, 1, 1, 0)]
sage: rows
```

```
[(1, 1, 0, 0, 1, 1, 1, 1), (0, 1, 0, 0, 1, 0, 1,
1), (0, 0, 1, 0, 1, 1, 0, 1), (0, 0, 1, 1, 1, 1,
1, 0)]
```

نعرف فضاء تحتي subspace على الفضاء المتجهي المحدود Finite Vector Space ثنائي الأبعاد F2 بالمدى spanned المولد السطرين السابقين

```
sage: V = VectorSpace(GF(2),8)
sage: S = V.subspace(rows)
sage: S
Vector space of degree 8 and dimension 4 over
Finite Field of size 2
Basis matrix:
[1 0 0 0 0 1 0 0]
[0 1 0 0 1 0 1 1]
[0 0 1 0 1 1 0 1]
[0 0 0 1 0 0 1 1]
sage: A.echelon_form()
[1 0 0 0 0 1 0 0]
[0 1 0 0 1 0 1 1]
[0 0 1 0 1 1 0 1]
[0 0 0 1 0 0 1 1]
```

القوائم و المرتبات و المتتابعات Lists, Tuples, and Sequences:

التركيب البياني او نوع البيانات قائمة يخزن عناصر من اي نوع إختياري و عناصر القائمة يبدأ ترقيمها من الصفر (0) :

```
sage: v = [2, 3, 5, 'x', SymmetricGroup(3)]; v
[2, 3, 5, 'x', Symmetric group of order 3! as a
permutation group]
sage: type(v)
<type 'list'>
sage: v[0]
2
sage: v[2]
5
```

```
sage: v = [1,2,3]
sage: v[2]
3
sage: n = 2        # SAGE Integer
sage: v[n]         # Perfectly OK!
3
sage: v[int(n)]   # Also OK.
3
```

دالة range (مجال) تولد قائمة على شاكلة Python (وليس على شاكلة Sage)

```
sage: range(1, 15)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

تكوين قائمة (معرفة لغة Python تساعد كثيرا لفهم المادة هنا)

```
sage: L = [factor(n) for n in range(1, 15)]
sage: print L
[1, 2, 3, 2^2, 5, 2 * 3, 7, 2^3, 3^2, 2 * 5, 11,
2^2 * 3, 13, 2 * 7]
sage: L[12]
13
sage: type(L[12])
<class
'sage.structure.factorization_integer.IntegerFac
torization'>
sage: [factor(n) for n in range(1, 15) if
is_odd(n)]
[1, 3, 5, 7, 3^2, 11, 13]
```

هناك مايسمى بألإسيعاب الفهمي للقوائم list comprehensions وهو عبارة عن دورة for في سطر واحد لتكوين قائمة. (انظر كتبي في المراجع)

تشريح القائمة List slicing:

بإستخدام مثلا L[m:n] على القائمة L

```
sage: L = [factor(n) for n in range(1, 20)]
sage: L[4:9]
[5, 2 * 3, 7, 2^3, 3^2]
sage: print L[:4]
[1, 2, 3, 2^2]
sage: L[14:4]
[]
sage: L[14:]
[3 * 5, 2^4, 17, 2 * 3^2, 19]
```

المرتبات Tuples:

وهي مثل القوائم ماعدى عدم إمكانية تغيرها immutable أي بعد توليدها لايمكن تغيرها.

```
sage: v = (1,2,3,4); v
(1, 2, 3, 4)
sage: type(v)
<type 'tuple'>
sage: v[1] = 5
Traceback (most recent call last):
...
TypeError: 'tuple' object does not support item
assignment
```

المتتابعات Sequences:

نوع ثالث شبه قائمة ويمكن تغييره وجميع عناصرها لها أصل أو والد مشترك common
parent يسمى كون المتتابعات sequences universe

```
sage: v = Sequence([1,2,3,4/5])
sage: v
[1, 2, 3, 4/5]
sage: type(v)
<class
'sage.structure.sequence.Sequence_generic'>
sage: type(v[1])
<type 'sage.rings.rational.Rational'>
sage: v.universe()
Rational Field
sage: v.is_immutable()
False
sage: v.set_immutable()
sage: v[0] = 3
Traceback (most recent call last):
...
```

```
ValueError: object is immutable; please change a
copy instead.
```

المتتابعات تشتق من القوائم ويمكن إستخدامها في أي مكان تستخدم فيه القوائم

```
sage: v = Sequence([1,2,3,4/5])
sage: isinstance(v, list)
True
sage: list(v)
[1, 2, 3, 4/5]
sage: type(list(v))
<type 'list'>
```

كمثال آخر المحاور الأساسية basis في فضاآت المتجهات vector spaces هي متتابعات غير قابلة للتغير immutable sequences (وهذا ضروري جدا)

```
sage: V = QQ^3; B = V.basis(); B
[
(1, 0, 0),
(0, 1, 0),
(0, 0, 1)
]
sage: type(B)
<class
'sage.structure.sequence.Sequence_generic'>
sage: B[0] = B[1]
Traceback (most recent call last):
...
ValueError: object is immutable; please change a
copy instead.
sage: B.universe()
Vector space of dimension 3 over Rational Field
```

القواميس Dictionaries:

القاموس dictionary ويسمى أحيانا صف متقارن associative array و التي يتم تكوينها بإستخدام الجداولة و تتكون من (عنصر او قيمة : مفتاح) وتسمى أزواج pairs in a dictionary

```
sage: d = {1:5, 'sage':17, ZZ:GF(7)}
sage: type(d)
<type 'dict'>
sage: d.keys()
 [1, 'sage', Integer Ring]
sage: d['sage']
17
sage: d[ZZ]
Finite Field of size 7
sage: d[1]
5
```

يمكن تحويل القاموس إلى قائمة كالتالي:

```
sage: d.items()
[(1, 5), ('sage', 17), (Integer Ring, Finite
Field of size 7)]
```

و نستطيع التكرار و الدور على ازواج القاموس

```
sage: d = {2:4, 3:9, 4:16}
sage: [a*b for a, b in d.iteritems()]
[8, 27, 64]
```

مجموعات Sets:

وتستخدم للبحث السريع فيما إذا كان عنصر في المجموعة ام لا بالإضاقة على عمليات نظرية المجموعات

```
sage: X = set([1,19,'a']);   Y = set([1,1,1,
2/3])
sage: X   # random sort order
{1, 19, 'a'}
sage: X == set(['a', 1, 1, 19])
True
sage: Y
{2/3, 1}
sage: 'a' in X
True
sage: 'a' in Y
False
sage: X.intersection(Y)
{1}
```

يمتلك Sage نوعه الخاص من المجموعات والتي لها خواص إضافية بالمقارنة مع Python

```
sage: X = Set([1,19,'a']);   Y = Set([1,1,1,
2/3])
sage: X   # random sort order
{'a', 1, 19}
sage: X == Set(['a', 1, 1, 19])
True
sage: Y
{1, 2/3}
sage: X.intersection(Y)
{1}
sage: print latex(Y)
\left\{1, \frac{2}{3}\right\}
sage: Set(ZZ)
```

```
Set of elements of Integer Ring
```

المكررات Iterators:

```
sage: v = (n^2 for n in xrange(10000000))
sage: next(v)
0
sage: next(v)
1
sage: next(v)
4
```

سوف نولد تكرار على الأرقام الأولية من الشكل 4 p + 1 حيث p اولية ايضا

```
sage: w = (4*p + 1 for p in Primes() if
is_prime(4*p+1))
sage: w          # in the next line, 0xb0853d6c
is a random 0x number
<generator object at 0xb0853d6c>
sage: next(w)
13
sage: next(w)
29
sage: next(w)
53
```

بعض الحلقات rings مثل الحقول النهائية finite fields و الأعداد الصحيحة integers لها مكرراتها الخاصة

```
sage: [x for x in GF(7)]
[0, 1, 2, 3, 4, 5, 6]
sage: W = ((x,y) for x in ZZ for y in ZZ)
sage: next(W)
(0, 0)
sage: next(W)
```

```
(0, 1)
sage: next(W)
(0, -1)
```

Loops, Functions, Control الدورات و الدوال وجمل التحكم و المقارنات Statements, and Comparisons:

```
sage: for i in range(5):
....:     print(i)  # now hit enter twice
....:
0
1
2
3
4
```

الرمز = يستخدم للإسناد بينما == يستخدم للمساواة

```
sage: for i in range(15):
....:     if gcd(i,15) == 1:
....:         print(i)
....:
1
2
4
7
8
11
13
14
```

```
sage: def legendre(a,p):
....:     is_sqr_modp=-1
```

```
....:     for i in range(p):
....:         if a % p == i^2 % p:
....:             is_sqr_modp=1
....:     return is_sqr_modp

sage: legendre(2,7)
1
sage: legendre(3,7)
-1
```

لاحظ كيف الميل الداخلي indentation يحدد تركيب مجموعة من الترميز للجمل if و for و while.

```
sage: 2 < 3.1; 3.1 <= 1
True
False
sage: 2/3 < 3/2;   3/2 < 3/1
True
True
```

لاحظ ان المقارنات مثل <, >, <=, >=, !=, == بين ارقام سوف تحول بشكل ذاتي الأرقام لنفس النوع.

```
sage: 2 < CC(3.1,1)
True
```

نستطيع مقارنة أي شيئين

```
sage: 5 < VectorSpace(QQ,3)    # output can be
somewhat random
True
sage: x < x + 1
x < x + 1
sage: bool(x < x + 1)
True
```

تستخدم الكلمة bool للمقارنة الرمزية.

```
sage: 1 is 2/2
False
sage: int(1) is int(2)/int(2)
True
sage: 1 is 1
False
sage: 1 == 2/2
True
sage: GF(5)(1) == QQ(1); QQ(1) == GF(5)(1)
False
False
sage: GF(5)(1) == ZZ(1); ZZ(1) == GF(5)(1)
True
True
sage: ZZ(1) == QQ(1)
True
```

```
sage: magma('GF(5)!1 eq Rationals()!1')
# optional - magma
true
```

الإحصاء في Sage : Sage for Statistics

يمتلك SageMath مقدرات كثيرة في الحسابات الإحصائية و بحوث العمليات مستمدة من Python (من خلال Scipy.stat) و من R.

```
sage: mean([1,2,3,5])
11/4
sage: std([1,2,2,4,5,6,8])
```

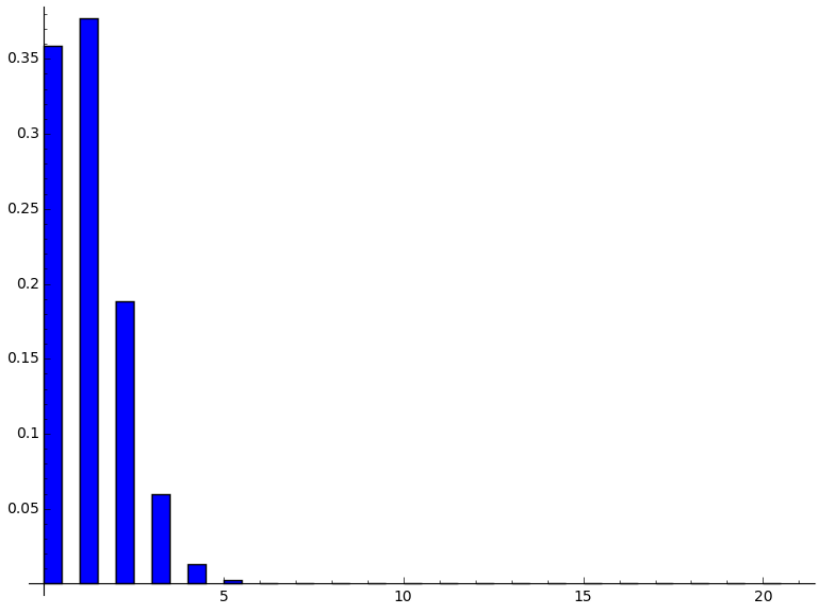
std الإنحراف المعياري

```
sqrt(19/3)
```

```
sage: data = ([1.2, 1.5, 1.3, 1.6, 1.1, 1.4,
1.5, 1.1, 1.5, 1.5])

sage: data

[1.2,1.5,1.3,1.6,1.1,1.4,1.5,1.1,1.5,1.5]

sage: mean(data)

1.37

sage: std(data)

0.182878222991269

sage: median(data)

1.45

sage: mode(data)

[1.5]
```

```
sage: max(data)

1.60000000000000

sage: min(data)

1.10000000000000
```

```
sage: import scipy.stats
sage: binom_dist = scipy.stats.binom(20,.05)
sage: bar_chart([binom_dist.pmf(x) for x in
range(21)])
```




```
sage: data = ([1.2, 1.5, 1.3, 1.6, 1.1, 1.4, 1.5, 1.1, 1.5, 1.5])

sage: scipy.stats.describe(data)
```

```
(10, (1.1, 1.6),
 1.3699, 0.03344, -0.4526, -1.296)
sage: scipy.stats.ttest_1samp(data,1.4)
(array(-0.51875), 0.6164)
sage: z=([2.8, 3.4, 3.7, 2.2, 2.0])
sage: scipy.stats.kstest(z,'norm')
(0.9772, 1.2188e-08)
sage: z=([2.8, 3.4, 3.7, 2.2, 2.0])
sage: scipy.stats.kstest(z,'t',(4,))
(0.9419, 1.3193e-06)
sage: x=([2.9, 3.0, 2.5, 2.6, 3.2])
sage: scipy.stats.ks_2samp(x,z)
(0.4, 0.697)
sage: import numpy
sage: from scipy.optimize import curve_fit
sage: def func(x, a, b):
...       return a*x + b
sage: x = numpy.linspace(0,10, 100)
```

```
sage: y = func(x, 1, 2)

sage: yn = y + 0.9 * numpy.random.normal(size =
len(x))

sage: popt , pcov = curve_fit(func, x, yn)

sage: popt; pcov

array([ 0.96397614,  2.26985691])

array([[ 0.00089214, -0.00446069],

       [-0.00446069,  0.02988812]])
```

Using R from within Sage SageMath إستخدام R من داخل

توجد طرق كثيرة لإستخدام R داخل Sage

1- أسهلها وضع r( ) حول أي شيئ نريد تحليله بواسطة R

2- نستخدم أوامر R من خلال r.method( ) لتمريرها إلى Sage

مثال:

```
sage: x=r([2.9, 3.0, 2.5, 2.6, 3.2]) # normal
subjects
sage: y=r([3.8, 2.7, 4.0, 2.4])      # with
obstructive airway disease
sage: z=r([2.8, 3.4, 3.7, 2.2, 2.0]) # with
asbestosis
sage: a = r([x,y,z]) # make a long R vector of
all the data
sage: b = r.factor(5*[1]+4*[2]+5*[3]) # create
something for R to tell which subjects are which
```

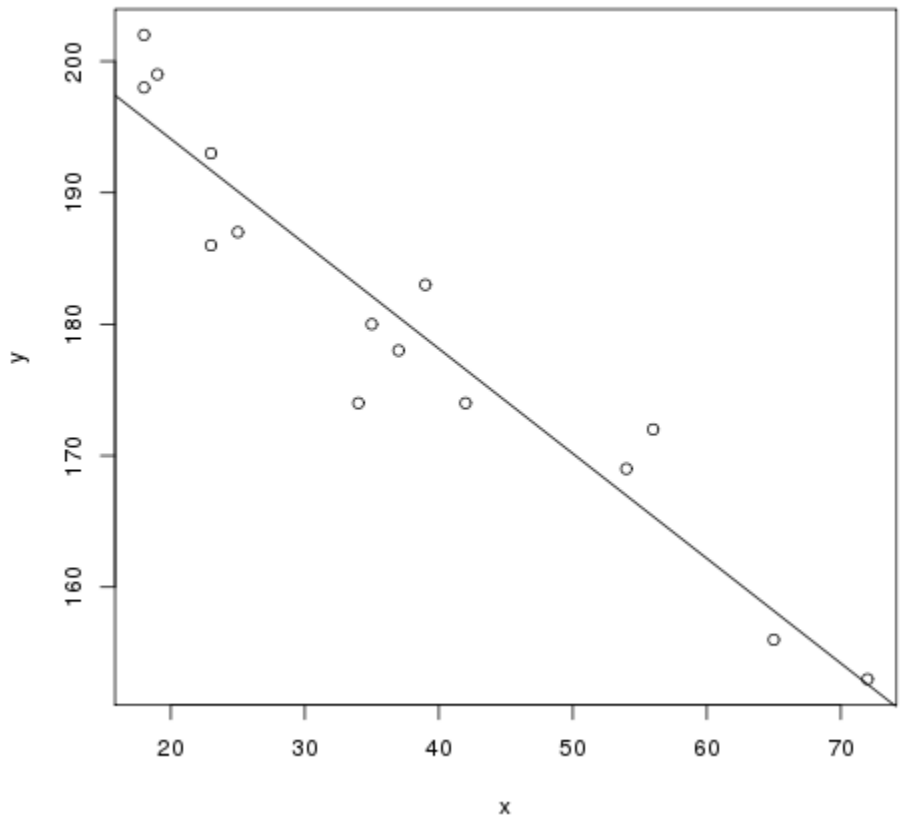
```
sage: a; b # show them
 [1] 2.9 3.0 2.5 2.6 3.2 3.8 2.7 4.0 2.4 2.8 3.4
3.7 2.2 2.0
 [1] 1 1 1 1 1 2 2 2 2 3 3 3 3 3
Levels: 1 2 3
sage: r.kruskal_test(a,b) # do the KW test!
    Kruskal-Wallis rank sum test

data:  sage17 and sage33
Kruskal-Wallis chi-squared = 0.7714, df = 2, p-
value = 0.68
```

أفضل طريقة لإستخدام R هي جعل كل جملة تحسب داخل R بإستخدام التوجيه المئوي "percent directive"%r

مثال:

```
sage: %r
sage: x =
c(18,23,25,35,65,54,34,56,72,19,23,42,18,39,37)
sage: y =
c(202,186,187,180,156,169,174,172,153,199,193,17
4,198,183,178)
sage: png()
sage: plot(x,y)
sage: lm(y ~ x)
sage: abline(lm(y ~ x))
sage: dev.off()

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   210.0485      -0.7977
```

```
null device
          1
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   210.0485      -0.7977

null device
          1
```



Sage في بحوث العمليات

البرمجة الخطية و العددية المختلطة بواسطة SageMath

مثال 1:

$$
\begin{aligned}
Max \quad & 8x_1 + 5x_2 \\
St \quad & 2x_1 + x_2 \le 1000 \\
& 3x_1 + 4x_2 \le 2400 \\
& x_1 + x_2 \le 700 \\
& x_1 - x_2 \le 350 \\
& x_1, x_2 \ge 0
\end{aligned}
$$

```
lp1 = MixedIntegerLinearProgram( maximization =
True)
x = lp1.new_variable(nonnegative = True)
lp1.set_objective( 8*x[1] + 5*x[2])
lp1.add_constraint(2*x[1] + x[2]  <= 1000)
lp1.add_constraint( 3* x[1] + 4*x[2]   <= 2400)
lp1.add_constraint(x[1] + x[2] <= 700 )
lp1.add_constraint(x[1]-  x[2]     <= 350 )
lp1.show()
lp1.solve()
lp1.get_values(x)
lp1.get_values(x[1])
lp1.get_values(x[2])
```

```
lp1.polyhedron()
```

مثال 2

$$
\begin{aligned}
Max.\quad & 12000X_1 + 20000X_2 \\
St.\quad & 2X_1 + 6X_2 \le 27 \\
& X_2 \ge 2 \\
& 3X_1 + X_2 \le 19 \\
& X_1, X_2 \ge 0 \text{ and integers}
\end{aligned}
$$

```
lp3 = MixedIntegerLinearProgram( maximization = True)
x = lp3.new_variable(nonnegative = True, integer = True)
lp3.set_objective( 12000*x[1] +20000*x[2])
lp3.add_constraint(2*x[1] + 6*x[2]  <= 27)
lp3.add_constraint( x[2] >= 2)
lp3.add_constraint(3*x[1] + x[2] <= 19 )
lp3.show()
lp3.solve()
lp3.get_values(x)
lp3.polyhedron()
```

مثال 3:

مشكلة نقل Transportation Problem

لشركة مصنعين لإنتاج معين والذي يتم توزيعه عن طريق ثلاثة مراكز توزيع. سعر إنتاج الوحدة متساوي في جميع المصانع ولكن تكلفة النقل من المصانع إلى مراكز التوزيع تختلف حسب البيانات في الجدول التالي:

	Distribution Center 1	Distribution Center 2	Distribution Center 3
Plant A	4	6	4
Plant B	6	5	2

عملية النقل تتم مرة في الأسبوع. خلال الأسبوع كل مصنع ينتج على الأكثر 60 وحدة وكل مركز توزيع يحتاج 40 وحدة للتوزيع.

المطلوب تحديد عدد الوحدات التي تنقل من كل مصنع إلى مراكز التوزيع بحيث تكون تكلفة النقل أقل مايمكن؟

```
lpnet = MixedIntegerLinearProgram( maximization
= False)

x = lpnet.new_variable(nonnegative = True)

lpnet.set_objective(4*x[1,1] + 6*x[1,2] +
6*x[2,1] + 5*x[2,2] + 4*x[3,1] + 2*x[3,2])

lpnet.add_constraint(x[1,1] + x[1,2]  >= 40)

lpnet.add_constraint(x[2,1] + x[2,2]  >= 40)

lpnet.add_constraint(x[3,1] + x[3,2]  >= 40)

lpnet.add_constraint(x[1,1] + x[2,1] + x[3,1]
<= 60)

lpnet.add_constraint(x[1,2] + x[2,2] + x[3,2]
<= 60)

lpnet.show()

lpnet.solve()
```

```
lpnet.get_values(x)


Minimization:
  4.0 x_0 + 6.0 x_1 + 6.0 x_2 + 5.0 x_3 + 4.0
x_4 + 2.0 x_5
Constraints:
  - x_0 - x_1 <= -40.0
  - x_2 - x_3 <= -40.0
  - x_4 - x_5 <= -40.0
  x_0 + x_2 + x_4 <= 60.0
  x_1 + x_3 + x_5 <= 60.0
Variables:
  x_0 is a continuous variable (min=0.0,
max=+oo)
  x_1 is a continuous variable (min=0.0,
max=+oo)
  x_2 is a continuous variable (min=0.0,
max=+oo)
  x_3 is a continuous variable (min=0.0,
max=+oo)
  x_4 is a continuous variable (min=0.0,
max=+oo)
  x_5 is a continuous variable (min=0.0,
max=+oo)
{(1, 1): 40.0,
 (1, 2): 0.0,
 (2, 1): 20.0,
 (2, 2): 20.0,
 (3, 1): 0.0,
 (3, 2): 40.0}
```

مثال 4:

مثال على التخصيص (التحديد) Assignment Problem

يريد أحد المدربين لفريق سباحة تحديد سباحين لسباق مسابقة 200 متر سباحة متنوعة متتابعة لاربعة سباحين فقط كل منهم يسبح 50 متر من المسافة. الفريق يتكون من 5 أفراد والمدرب لديه بيانات عن أفضل أداء لكل منهم (بالثانية) في مسابقات سابقة كالتالي:

	Backstroke	Breaststroke	Butterfly	Freestyle
A	37.7	43.4	33.3	29.2
B	32.9	33.1	28.5	26.4
C	33.8	42.2	38.9	29.6
D	37	34.7	30.4	28.5
E	35.4	41.8	33.6	31.1

يريد المدرب تحديد أفضل فريق مكون من 4 سباحين لقطع مسافة السباق في أقل وقت ممكن.

```
lpnet = MixedIntegerLinearProgram( maximization
= False)

x = lpnet.new_variable(nonnegative = True,
binary = True)

lpnet.set_objective(37.7*x[1,1] + 43.4*x[1,2] +
33.3*x[1,3] + 29.2*x[1,4] + 32.9*x[2,1] +
33.1*x[2,2] + 28.5*x[2,3] + 26.4*x[2,4] +
33.8*x[3,1] + 42.2*x[3,2] + 38.9*x[3,3] +
29.6*x[3,4] + 37.0*x[4,1] + 34.7*x[4,2] +
```

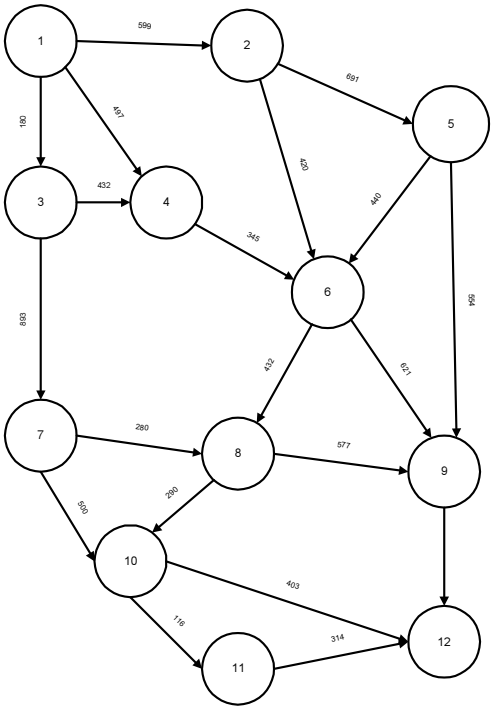
```
30.4*x[4,3] + 28.5*x[4,4] + 35.4*x[5,1] +
41.8*x[5,2] + 33.6*x[5,3] + 31.1*x[5,4])

lpnet.add_constraint(x[1,1] + x[1,2] + x[1,3] +
x[1,4]  <= 1)

lpnet.add_constraint(x[2,1] + x[2,2] + x[2,3] +
x[2,4]  <= 1)

lpnet.add_constraint(x[3,1] + x[3,2] + x[3,3] +
x[3,4]  <= 1)

lpnet.add_constraint(x[4,1] + x[4,2] + x[4,3] +
x[4,4]  <= 1)

lpnet.add_constraint(x[1,1] + x[2,1] + x[3,1] +
x[4,1]  == 1)

lpnet.add_constraint(x[1,2] + x[2,2] + x[3,2] +
x[4,2]  == 1)

lpnet.add_constraint(x[1,3] + x[2,3] + x[3,3] +
x[4,3]  == 1)

lpnet.add_constraint(x[1,4] + x[2,4] + x[3,4] +
x[4,4]  == 1)

lpnet.solve()


126.2


lpnet.show()


Minimization:
  37.7 x_0 + 43.4 x_1 + 33.3 x_2 + 29.2 x_3 +
32.9 x_4 + 33.1 x_5 + 28.5 x_6 + 26.4 x_7 + 33.8
x_8 + 42.2 x_9 + 38.9 x_10 + 29.6 x_11 + 37.0
```

```
x_12 + 34.7 x_13 + 30.4 x_14 + 28.5 x_15 + 35.4
x_16 + 41.8 x_17 + 33.6 x_18 + 31.1 x_19
Constraints:
  x_0 + x_1 + x_2 + x_3 <= 1.0
  x_4 + x_5 + x_6 + x_7 <= 1.0
  x_8 + x_9 + x_10 + x_11 <= 1.0
  x_12 + x_13 + x_14 + x_15 <= 1.0
  1.0 <= x_0 + x_4 + x_8 + x_12 <= 1.0
  1.0 <= x_1 + x_5 + x_9 + x_13 <= 1.0
  1.0 <= x_2 + x_6 + x_10 + x_14 <= 1.0
  1.0 <= x_3 + x_7 + x_11 + x_15 <= 1.0
Variables:
  x_0 is a boolean variable (min=0.0, max=1.0)
  x_1 is a boolean variable (min=0.0, max=1.0)
  x_2 is a boolean variable (min=0.0, max=1.0)
  x_3 is a boolean variable (min=0.0, max=1.0)
  x_4 is a boolean variable (min=0.0, max=1.0)
  x_5 is a boolean variable (min=0.0, max=1.0)
  x_6 is a boolean variable (min=0.0, max=1.0)
  x_7 is a boolean variable (min=0.0, max=1.0)
  x_8 is a boolean variable (min=0.0, max=1.0)
  x_9 is a boolean variable (min=0.0, max=1.0)
  x_10 is a boolean variable (min=0.0, max=1.0)
  x_11 is a boolean variable (min=0.0, max=1.0)
  x_12 is a boolean variable (min=0.0, max=1.0)
  x_13 is a boolean variable (min=0.0, max=1.0)
  x_14 is a boolean variable (min=0.0, max=1.0)
  x_15 is a boolean variable (min=0.0, max=1.0)
  x_16 is a boolean variable (min=0.0, max=1.0)
  x_17 is a boolean variable (min=0.0, max=1.0)
  x_18 is a boolean variable (min=0.0, max=1.0)
  x_19 is a boolean variable (min=0.0, max=1.0)


lpnet.get_values(x)
```

```
{(1, 1): 0.0,
 (1, 2): 0.0,
 (1, 3): 0.0,
 (1, 4): 1.0,
 (2, 1): 0.0,
 (2, 2): 0.0,
 (2, 3): 1.0,
 (2, 4): 0.0,
 (3, 1): 1.0,
 (3, 2): 0.0,
 (3, 3): 0.0,
 (3, 4): 0.0,
 (4, 1): 0.0,
 (4, 2): 1.0,
 (4, 3): 0.0,
 (4, 4): 0.0,
 (5, 1): 0.0,
 (5, 2): 0.0,
 (5, 3): 0.0,
 (5, 4): 0.0}
```

مثال 5:

مشكلة أقصر طريق



$$
\begin{aligned}
Min \quad & 599x_{1,2} + 180x_{1,3} + 497x_{1,4} + 691x_{2,5} + 420x_{2,6} + \\
& 432x_{3,4} + 893x_{3,7} + 345x_{4,6} + 440x_{5,6} + 554x_{5,9} + \\
& 432x_{6,8} + 621x_{6,9} + 280x_{7,8} + 500x_{7,10} + 577x_{8,9} + \\
& 290x_{8,10} + 268x_{9,12} + 116x_{10,11} + 403x_{10,12} + 314x_{11,12}
\end{aligned}
$$

$$
\begin{aligned}
ST \quad & x_{1,2} + x_{1,3} + x_{1,4} = 1 \\
& x_{2,5} + x_{2,6} - x_{1,2} = 0 \\
& x_{3,4} + x_{3,7} - x_{1,3} = 0 \\
& x_{4,6} - x_{1,4} - x_{3,4} = 0 \\
& x_{5,6} + x_{5,9} - x_{2,5} = 0 \\
& x_{6,8} + x_{6,9} - x_{2,6} - x_{4,6} - x_{5,6} = 0 \\
& x_{7,8} + x_{7,10} - x_{3,7} = 0 \\
& x_{8,9} + x_{8,10} - x_{6,8} - x_{7,8} = 0 \\
& x_{9,12} - x_{9,5} - x_{6,9} - x_{8,9} = 0 \\
& x_{10,11} + x_{10,12} - x_{7,10} - x_{8,10} = 0 \\
& x_{11,12} - x_{10,11} = 0 \\
& -x_{9,12} - x_{10,12} - x_{11,12} = 0 \\
All \quad & x_{i,j}\text{'s} = 0 \; or \; 1
\end{aligned}
$$

```
lpnet = MixedIntegerLinearProgram( maximization
= False)

x = lpnet.new_variable(nonnegative = True,
binary = True)

lpnet.set_objective(599*x[1,2] + 180*x[1,3] +
497*x[1,4] + 691*x[2,5] + 420*x[2,6] +
432*x[3,4] + 893*x[3,7] +


345*x[4,6] + 440*x[5,6] + 554*x[5,9] +
432*x[6,8] + 621*x[6,9] + 280*x[7,8] +
500*x[7,10] + 577*x[8,9] +
```

```
290*x[8,10] + 268*x[9,12] + 116*x[10,11] +
403*x[10,12] + 314*x[11,12])

lpnet.add_constraint(x[1,2] + x[1,3] + x[1,4]
== 1)

lpnet.add_constraint(x[2,5] + x[2,6] - x[1,2]
== 0)

lpnet.add_constraint(x[3,4] + x[3,7] - x[1,3]
== 0)

lpnet.add_constraint(x[4,6] - x[1,4] - x[3,4]
== 0)

lpnet.add_constraint(x[5,6] + x[5,9] - x[2,5]
== 0)

lpnet.add_constraint(x[6,8] + x[6,9] - x[2,6] -
x[4,6] - x[5,6] == 0)

lpnet.add_constraint(x[7,8] + x[7,10] - x[3,7]
== 0)

lpnet.add_constraint(x[8,9] + x[8,10] - x[6,8] -
x[7,8]  == 0)

lpnet.add_constraint(x[9,12] + x[9,5] - x[6,9] -
x[8,9]  == 0)

lpnet.add_constraint(x[10,11] + x[10,12] -
x[7,10] - x[8,10]  == 0)

lpnet.add_constraint(x[11,12] - x[10,11]  == 0)

lpnet.add_constraint(-x[9,12] - x[10,12] -
x[11,12]  == -1)

lpnet.solve()

lpnet.get_values(x)
```

```
{(1, 2): 0.0,
 (1, 3): 0.0,
 (1, 4): 1.0,
 (2, 5): 0.0,
 (2, 6): 0.0,
 (3, 4): 0.0,
 (3, 7): 0.0,
 (4, 6): 1.0,
 (5, 6): 0.0,
 (5, 9): 0.0,
 (6, 8): 0.0,
 (6, 9): 1.0,
 (7, 8): 0.0,
 (7, 10): 0.0,
 (8, 9): 0.0,
 (8, 10): 0.0,
 (9, 5): 0.0,
 (9, 12): 1.0,
 (10, 11): 0.0,
 (10, 12): 0.0,
 (11, 12): 0.0}
```

الحل بإستخدام نظرية الرسم Graph Theory

```
g = Graph(12)
g.add_edge(1,2,599)
g.add_edge(1,3,180)
g.add_edge(1,4,497)
g.add_edge(2,5,691)
g.add_edge(2,6,420)
g.add_edge(3,4,432)
```

```
g.add_edge(3,7,893)
g.add_edge(4,6,345)
g.add_edge(5,6,440)
g.add_edge(5,9,554)
g.add_edge(6,8,432)
g.add_edge(6,9,621)
g.add_edge(7,8,280)
g.add_edge(7,10,500)
g.add_edge(8,9,577)
g.add_edge(8,10,290)
g.add_edge(9,12,268)
g.add_edge(10,11,116)
g.add_edge(10,12,403)
g.add_edge(11,12,314)
g.show()
from sage.graphs.distances_all_pairs import
shortest_path_all_pairs
shortest_path_all_pairs(g)
{0: {0: None,
  1: None,
  2: None,
  3: None,
  4: None,
  5: None,
  6: None,
  7: None,
```

```
 8: None,
 9: None,
 10: None,
 11: None,
 12: None},
1: {0: None,
 1: None,
 2: 1,
 3: 1,
 4: 1,
 5: 2,
 6: 2,
 7: 3,
 8: 6,
 9: 5,
 10: 7,
 11: 10,
 12: 9},
2: {0: None,
 1: 2,
 2: None,
 3: 1,
 4: 1,
 5: 2,
 6: 2,
 7: 3,
 8: 6,
 9: 5,
 10: 8,
 11: 12,
 12: 9},
3: {0: None,
 1: 3,
 2: 1,
 3: None,
```

```
 4: 3,
 5: 2,
 6: 4,
 7: 3,
 8: 7,
 9: 6,
 10: 7,
 11: 10,
 12: 10},
4: {0: None,
 1: 4,
 2: 1,
 3: 4,
 4: None,
 5: 6,
 6: 4,
 7: 3,
 8: 6,
 9: 6,
 10: 7,
 11: 10,
 12: 9},
5: {0: None,
 1: 2,
 2: 5,
 3: 1,
 4: 6,
 5: None,
 6: 5,
 7: 8,
 8: 6,
 9: 5,
 10: 8,
 11: 12,
 12: 9},
```

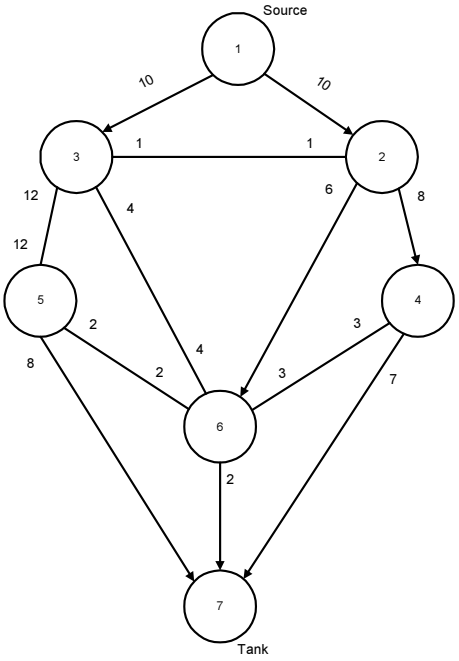
```
6: {0: None,
 1: 2,
 2: 6,
 3: 4,
 4: 6,
 5: 6,
 6: None,
 7: 8,
 8: 6,
 9: 6,
 10: 8,
 11: 10,
 12: 9},
7: {0: None,
 1: 3,
 2: 1,
 3: 7,
 4: 3,
 5: 6,
 6: 8,
 7: None,
 8: 7,
 9: 8,
 10: 7,
 11: 10,
 12: 10},
8: {0: None,
 1: 2,
 2: 6,
 3: 7,
 4: 6,
 5: 6,
 6: 8,
 7: 8,
 8: None,
```

```
 9: 8,
 10: 8,
 11: 10,
 12: 9},
9: {0: None,
 1: 2,
 2: 5,
 3: 4,
 4: 6,
 5: 9,
 6: 9,
 7: 8,
 8: 9,
 9: None,
 10: 8,
 11: 12,
 12: 9},
10: {0: None,
 1: 3,
 2: 6,
 3: 7,
 4: 3,
 5: 6,
 6: 8,
 7: 10,
 8: 10,
 9: 8,
 10: None,
 11: 10,
 12: 10},
11: {0: None,
 1: 3,
 2: 6,
 3: 7,
 4: 3,
```

```
 5: 9,
 6: 8,
 7: 10,
 8: 10,
 9: 12,
 10: 11,
 11: None,
 12: 11},
12: {0: None,
 1: 2,
 2: 5,
 3: 7,
 4: 6,
 5: 9,
 6: 9,
 7: 10,
 8: 9,
 9: 12,
 10: 12,
 11: 12,
 12: None}}
```

مثال 6:

مشكلة التدفق الأعظم



$Max \quad x_{1,2} + x_{1,3}$

St

$$x_{2,3} + x_{2,4} + x_{2,6} - x_{1,2} - x_{3,2} = 0$$
$$x_{3,2} + x_{3,5} + x_{3,6} - x_{1,3} - x_{2,3} - x_{6,3} = 0$$
$$x_{4,6} + x_{4,7} - x_{2,4} - x_{6,4} = 0$$
$$x_{5,6} + x_{5,7} - x_{3,5} - x_{6,5} = 0$$
$$x_{6,3} + x_{6,4} + x_{6,5} + x_{6,7} - x_{2,6} - x_{3,6} - x_{4,6} - x_{5,6} = 0$$
$$x_{1,2} \le 10$$
$$x_{1,3} \le 10$$
$$x_{2,3} \le 1$$
$$x_{2,4} \le 8$$
$$x_{2,6} \le 6$$
$$x_{3,2} \le 1$$
$$x_{3,5} \le 12$$
$$x_{3,6} \le 4$$
$$x_{4,6} \le 3$$
$$x_{4,7} \le 7$$
$$x_{5,6} \le 2$$
$$x_{5,7} \le 8$$
$$x_{6,3} \le 4$$
$$x_{6,4} \le 3$$
$$x_{6,5} \le 2$$
$$x_{6,7} \le 2$$

$All \quad x_{i,j}\text{'}s \ge 0$

```
lpnet = MixedIntegerLinearProgram( maximization
= True)
x = lpnet.new_variable(nonnegative = True)
lpnet.set_objective(x[1,2] + x[1,3])
```

```
lpnet.add_constraint(x[2,3] + x[2,4] + x[2,6] -
x[1,2] - x[3,2] == 0)

lpnet.add_constraint(x[3,2] + x[3,5] + x[3,6] -
x[1,3] - x[2,3] - x[6,3] == 0)

lpnet.add_constraint(x[4,6] + x[4,7] - x[2,4] -
x[6,4] == 0)

lpnet.add_constraint(x[5,6] + x[5,7] - x[3,5] -
x[6,5] == 0)

lpnet.add_constraint(x[6,3] + x[6,4] + x[6,5] +
x[6,7] - x[2,6] - x[3,6] - x[4,6] - x[5,6] == 0)

lpnet.add_constraint(x[1,2] <= 10)

lpnet.add_constraint(x[1,3] <= 10)

lpnet.add_constraint(x[2,3] <= 1)

lpnet.add_constraint(x[2,4] <= 8)

lpnet.add_constraint(x[2,6] <= 6)

lpnet.add_constraint(x[3,2] <= 1)

lpnet.add_constraint(x[3,5] <= 12)

lpnet.add_constraint(x[3,6] <= 4)

lpnet.add_constraint(x[4,6] <= 3)

lpnet.add_constraint(x[4,7] <= 7)

lpnet.add_constraint(x[5,6] <= 2)

lpnet.add_constraint(x[5,7] <= 8)

lpnet.add_constraint(x[6,3] <= 4)

lpnet.add_constraint(x[6,4] <= 3)

lpnet.add_constraint(x[6,5] <= 2)
```

```
lpnet.add_constraint(x[6,7] <= 2)
lpnet.solve()

17.0

lpnet.get_values(x)
{(1, 2): 7.0,
 (1, 3): 10.0,
 (2, 3): 0.0,
 (2, 4): 7.0,
 (2, 6): 0.0,
 (3, 2): 0.0,
 (3, 5): 8.0,
 (3, 6): 2.0,
 (4, 6): 0.0,
 (4, 7): 7.0,
 (5, 6): 0.0,
 (5, 7): 8.0,
 (6, 3): 0.0,
 (6, 4): 0.0,
 (6, 5): 0.0,
 (6, 7): 2.0}
```

مثال 7:

مشكلة الجراب

لنفرض أنك تريد القيام بنزهة سيرا على الأقدام في أحد المرتفعات الجبلية وقد جهزت قائمة بالأشياء التي تحتاج إليها. كل عنصر من هذه الإشياء له وزن معين والحقيبة لاتستوعب أكثر من 15 كجم. ايضا من ضمن القائمة انت وضعت معيار أهمية لكل عنصر فبدت القائمة كالتالي:

Item	Weight	Rating
Ant Repellent	1	2
Pepsi	3	9
Blanket	4	3
Meat	3	8
Cakes	3	10
Football	1	6
Salad	5	4
Watermelon	10	10
---	---	---
Sum	30	

المطلوب تحديد الأشياء المهمة التي تحملها معك مراعيا حجم الحقيبة.

```
from sage.numerical.knapsack import knapsack
knapsack( [(1,2), (3,9), (4,3), (3,8), (3,10),
(1,6), (5,4), (10,10)], max=15)


[38.0, [(1, 2), (3, 9), (4, 3), (3, 8), (3, 10),
(1, 6)]]
```

مثال 8:

نموذج سلسلة ماركوف Markov chain model:

نريد إيجاد حل حالة الإستقرار أي $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_n\}$ بحيث $\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}$ و $\boldsymbol{\pi}\mathbf{e} = 1$ لمصفوفة الإنتقال:

$$\mathbf{P} = \begin{pmatrix} 0.75 & 0.1 & 0.05 & 0.1 \\ 0.4 & 0.2 & 0.1 & 0.3 \\ 0.1 & 0.2 & 0.4 & 0.3 \\ 0.2 & 0.2 & 0.3 & 0.3 \end{pmatrix}$$

```
tp = matrix([[ 0.75,0.1,0.05,0.1],
[0.4,0.2,0.1,0.3],
[0.1,0.2,0.4,0.3],[0.2,0.2,0.3,0.3]])

pp =tp^1000

pp

############ OR ######################

tp = matrix([[ 0.75,0.1,0.05,0.1],
[0.4,0.2,0.1,0.3],
[0.1,0.2,0.4,0.3],[0.2,0.2,0.3,0.3]])

tp

p = vector([0.2, 0.3, 0.4, 0.1])

p

pp = p * tp^1000

pp


(0.475000000000023, 0.152500000000007,
0.167500000000008, 0.205000000000010)
################### OR ########################
```

```
tp = matrix(QQ,[[ .75,.1,.05,.1], [.4,.2,.1,.3],
[.1,.2,.4,.3],[.2,.2,.3,.3]])
eigen = tp.eigenvectors_left()
eigen
pi = [k[1][0] for k in eigen if k[0] == 1][0]
pi = [k / sum(pi) for k in pi]
pi
[19/40, 61/400, 67/400, 41/200]
```

مراجع:

1- Statistics with Python by Examples

2- Statistical Packages using R, Python and EDAT

على موقعي http://www.abarry.ws/

3- http://doc.sagemath.org/html/en/tutorial/tour_linalg.html

4- http://doc.sagemath.org/html/en/tutorial/index.html

5- http://www.sagemath.org/

6- https://sagecell.sagemath.org/