Python 練習問題

クラス 1

問題 1-1 (関数定義)

4 つの数 a, b, c, d を受け取り, ベクトル (a, b) と (c, d) の内積を計算する関数, `inner(a, b, c, d)` を書き, 適当な例を使って確かめよ.

```
def inner(a, b, c, d):
    ...

inner(...)
```

問題 1-2 (関数定義, import)

関数

$$f(x) = \frac{x}{\sin x} + \cos x$$

を計算する Python の関数 `f(x)` を書け.

これを使って $x \to +0$, $x \to \pi - 0$ の値を計算 (予想) せよ.

sin, cos, π は `math` モジュールの `sin, cos, pi` という名前でアクセスできる.

問題 1-3 (関数定義, import, 数値微分)

微分係数の定義:

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

に従えば, h を 0 に近い数 (例えば 0.001) に対して,

$$f'(a) \approx \frac{f(a+h) - f(a)}{h}$$

が成り立つ. これを利用して具体的な a に対する微分係数の近似値が簡単に計算できる. 難しい記号操作をする必要はなく, 必要なのは, $f(a)$ の値が計算できる事だけである.

これを利用して, $\log x$ の, $x = a$ における微分係数の近似値を求める関数 `log_prime(a)` を書き, 自分の知る事実 $(\log x)' \approx 1/x$ であることを, いくつかの a に対して確かめてみよ. 他の関数についてもやってみよ.

なお, e^x や $\log x$ の値を計算する関数は, `math` モジュールに, `exp, log` という名前で提供されている.

注: 上式で h をいくらにすればよいかについてはあまり深く悩む必要はないが以下に注意して適度な値を選ぶ.

- $h \to 0$ での極限を知りたいのだから, 一般的には小さい方がよい
- 一方あまり小さくしすぎると, $a + h$ の計算が, 計算機が固定の桁数しか表せないことよる「丸め誤差」を起こしてしまい, 逆に誤差が大きくなってしまう (最悪の場合, $a + h = a$ になってしまう. ためしに, $1 + 10^{-16}$ を計算してみよ). したがって無闇に h を小さくすればいいというわけではない. 一般には a に比べて h が相対的に小さすぎては困る.

問題 1-4 (関数の関数)

上記では $\log x$ の, a における導関数の値を求めたが, 全く同じ計算方法で, $f(x) = e^x$ の導関数だろうが $f(x) = \sin x$ の導関数だろうが, 計算できる．しかしプログラム上は，関数が変わるたびに似たようなプログラムを書かなくてはならない．それよりも「(任意の) 関数 f(と a) を受け取り，その $f'(a)$ を求める」関数を書くことができれば便利である．

実は Python においては関数を，他の関数に渡したり変数に代入したりできるのでこのようなことが素直に可能である．例えば以下は,「(任意の) 関数 f と値 a」を受け取り，$f(f(a))$ を計算する関数である．

```
def twice(f, a):
    return f(f(a))
```

たとえば以下で, $\log\log x$ を計算する関数が出来上がる．

```
def double_exp(x):
    return twice(math.log, x)
```

以上を参考に,「f と a を受け取り，$f'(a)$ を求める」関数 `grad_at(f, a)` を書け.

```
def grad_at(f, a):
    ...
```

これに，`math.sin`, `math.log` などを渡して，微分係数を計算してみよ．以下の計算結果は, 微分係数の近似式

$$\frac{f(a+h) - f(a)}{h}$$

で h をいくらにするかによって変わるので, ピタリ一致しなくても気にする必要はない.

```
grad_at(math.sin, math.pi/3)
 → 0.49995669789693054    # ≈ cos π/3
```

```
grad_at(math.log, 20)
 → 0.04999987500031722    # ≈ 1/20
```

もちろん自分で定義した関数も渡すことができる．

```
def p(x):
    return x ** 3 + x ** 2

grad_at(p, 2)
 → 16.00070001003928  # 3x^2 + 2x | x=2
```

問題 1-5 (関数を返す関数)

関数を別の関数に渡すことができるだけでなく, 関数を, 関数の実行結果として返すこともできる．文法的には, 関数定義文の `def` も文の一種であるので, 関数の中に書くことができる, ということに過ぎない．例えば以下は, a, b, c を受け取ると,

$$f(x) = ax^2 + bx + c$$

という「関数」f を返す関数 (`make_quad`) である．

```
def make_quad(a, b, c):
    def f(x):
        return a * x * x + b * x + c
    return f
```

これに色々な a, b, c を渡すことで色々な 2 次関数を, いくらでも生むことができる.

```
g = make_quad(1, 2, 3)
h = make_quad(4, 5, 6)
```

そうやってできた関数は普通に `def` で定義した関数と同じように呼び出せる.

```
g(1)
 → 6
```

```
h(1)
 → 15
```

この考え方を用いて, 与えられた関数に対してその導関数を返す関数 `derive(f)` を書け.

```
def derive(f):
    ...
```

それを使って例えば以下のようなことができる.

```
sin_prime = derive(math.sin)
```

(`sin_prime(x)` $\approx \cos x$ のはず...)

```
sin_prime(math.pi/4) - math.cos(math.pi/4)
 → -3.535651724428934e-05    # −3.53... × 10^−5
```

`derive` は関数を受取りまた関数を返すので, 繰り返し適用することもできる. 例えば,

```
sin_prime2 = derive(derive(math.sin))
```

`sin_prime2(`x`)` $\approx -\sin(x)$ のはずなので,

```
sin_prime2(math.pi/5) + math.sin(math.pi/5)
 → -8.088125763472398e-05   # −8.08... × 10^−5
```

この話がややこしく思えた人はあまり悩まなくて良い. 最終的にある a における導関数の値 $f'(a)$ が得たいだけなら, `grad_at(`f`, `a`)` として計算するのも, `f_prime = derive(`f`)` としてから `f_prime(`a`)` とするのも同じことである. `derive` のように, 「関数を返す関数」を書くことができるのは数学で出てくる計算を素直に表すことができそうだ, と思ってもらえれば良い. その一つの例が, `derive` を 2 度適用するだけで, 2 階の導関数が導けるということである.

問題 1-6 (複素数使えます)

Python では複素数も容易に扱える.

- 虚数単位 (i) は, `1j` または `1.0j` などと表記
- 複素数の実部, 虚部は, 式`.real`, 式`.imag`
- 複素共役は, 式`.conjugate()`
- 普通の数と同じように, 四則が行える
- `cmath` というモジュールを import すると, 複素数の演算を行う関数が使える

例:

```
1.0j * 1.0j
 → (-1+0j)
z = 3.0 + 4.0j
z.real
 → 3.0
z.imag
 → 4.0
z * z.conjugate()
 → (25+0j)
abs(z)
 → 5.0
import cmath
cmath.sqrt(1.0j)
 → (0.7071067811865476+0.7071067811865475j)
```

問題: 複素平面上の原点を中心とする半径 1 の円を C とする. 点 $\mathrm{P}(z)$ は C 上にあり, 点 $\mathrm{A}(1)$ とは異なるとする. 点 P における円 C の接線に関して, 点 A と対称な点を $\mathrm{Q}(u)$ とする. $w = \dfrac{1}{1-u}$ とおき, w と共役な複素数を $\overline{w}$ で表す.

実数 θ が与えられると,

$$z = \cos\theta + i\sin\theta$$

に対して,

$$\frac{|w + \overline{w} - 1|}{|w|}$$

を計算する Python 関数 `g(theta)` を書け. いくつかの `theta` に対して `g(theta)` を計算してみよ.

注: z から u を計算する部分は多少の数学が必要. 今の主題ではない (し, すでにやった, 思い出したくない, etc.) ので省略したい人は,

$$u = -z^2 + 2z$$

を認めても良い.

真面目にやりたい人向けのヒント: 点 z の, x 軸に関して対称な点は $\bar{z}$ であることを利用. 一般の直線に関して対称な点を求めたければ, うまいことその直線が x 軸になるように回転・平行移動する.

クラス 2

問題 2-1 (visual python の練習)

Visual Python を用いて, 次の 3 つの矢印を表示する関数 `axes()` を作れ.

- 原点と (1,0,0) を結ぶベクトルを赤で
- 原点と (0,1,0) を結ぶベクトルを青で
- 原点と (0,0,1) を結ぶベクトルを黄色で

```
from vpython import *
def axes():
    ...

axes()
```

- 矢印を表示する関数は? → 授業 HP の「有用リンク → どんなオブジェを作れるか」を参照
- 色を付けるには?

```
axis(color=color.blue)
```

のようにして色を指定する (上記は青の例). 他のオブジェでも同様. 色に指定できる式は, 上記ページ左のメニューから “Work with 3D objects” → “Color/Opacity”

- 表示窓の操作方法を練習してみよ. 右クリック + ドラッグ, ホイール
- 見栄えが良くなるよう適宜, 矢印の太さなどを調節してみよ

問題 2-2 (visual python の練習)

visual python を用いて,「天井」から「バネ」で吊るされた「球」を表示する関数 `mass_spring()` を作れ.

```
from vpython import *
def mass_spring():
    ...

mass_spring()
```

問題 2-3 (visual python の練習)

visual python を用いて，メタン分子 (CH_4)っぽいものを表示する関数 `methane()` を作れ.

```
from vpython import *
def methane():
    ...

methane()
```

問題 2-4 (visual python の vector)

visual python では，3 次元ベクトルを簡単に扱える．vector(x, y, z) で，x, y, z を成分にもつ 3 次元ベクトルが出来，あとは通常の演算や，内積，外積が簡単に計算できる．

```
from vpython import *
u = vector(1,2,3)
u
 → vector(1, 2, 3)
v = vector(4,5,6)
u + v
 → vector(5, 7, 9)
u - v
 → vector(-3, -3, -3)
u.dot(v)
 → 32.0
u.cross(v)
 → vector(-3, 6, -3)
u.x （またはu[0])
 → 1.0
u.y （またはu[1])
 → 2.0
u.z （またはu[2])
 → 3.0
```

- 空間内の 3 点 A, B, C (それぞれ vector として与えられる) に対し，三角形 ABC の面積を求める関数 area(A, B, C) を書け．
- 空間内の 2 点 A, B に対し，直線 AB と xy 平面 ($z = 0$) との交点を求める関数 intersect_xy(A, B) を書け．
- それらを用い以下の関数 $S(a)$ を計算する関数を書け．

  a を $1 < a < 3$ をみたす実数とし，座標空間内の 4 点 $\mathrm{P}_1(1, 0, 1)$, $\mathrm{P}_2(1, 1, 1)$, $\mathrm{P}_3(1, 0, 3)$, $\mathrm{Q}(0, 0, a)$ を考える．直線 $\mathrm{P}_1\mathrm{Q}$, $\mathrm{P}_2\mathrm{Q}$, $\mathrm{P}_3\mathrm{Q}$ と xy 平面の交点をそれぞれ R_1, R_2, R_3 として，三角形 $\mathrm{R}_1\mathrm{R}_2\mathrm{R}_3$ の面積を $S(a)$ とする．

原題 (東大 2016 前期入試数学第 3 問) は，この $S(a)$ を最小にする a とその時の $S(a)$ を求めよというもの．ここではさしあたり $S(2) = 4$ となることを確認しておけばよい．

クラス 3

問題 3-1 (繰り返し for, range)

次で定まる数列の第 n 項を計算する関数 `a(c, n)` を書け.

$$a_0 = c, \tag{1}$$
$$a_n = \frac{1}{3}\left(2a_{n-1} + \frac{c}{a_{n-1}^2}\right) \ (n > 0). \tag{2}$$

これは $\sqrt[3]{c}$ に収束する数列で, ある程度大きな n に対して, `a(c, n)` を計算すればそれが, $\sqrt[3]{c}$ の近似値となる.
`a(c, n)` を 3 乗して, 結果がほぼ c と同じになるか確かめよ.
for 文を一回繰り返す毎に a_n の値を表示する print 文を挿入して, $\sqrt[3]{c}$ に収束していく様子を表示してみよ (これは, 計算結果が思うようにでない場合の最も基本的な調査手段).

```
def a(c, n):
    ...

a(5, 100000) ** 3
```

参考: 一般に, $f(x) = 0$ の解を求めるのに以下のニュートン法が用いられる.

$$a_0 = c, \tag{3}$$
$$a_n = a_{n-1} - f(a_{n-1})/f'(a_{n-1}) \ (n > 0). \tag{4}$$

これが**収束するならば**, $x = x - f(x)/f'(x)$ を満たす x, すなわち $f(x) = 0$ を満たす x に収束することは明らか. 本問はこれを, $f(x) = x^3 - c$ に適用したものである.
ただし, 収束するとは限らないので無条件に使えるわけではない.
計算機で何かを求めるとき, このようなに「欲しい答えに収束するような数列」を使って計算することは非常に多い.

問題 3-2 for 文

(1) 2 整数 a, b に対し, 積:

$$a \times (a+1) \times \cdots \times b$$

を計算する関数 `prod(a, b)` を書け

(2) 以下の数列:

$$a_n = \frac{{}_{2n+1}C_n}{n!}$$

が整数になる $n \geq 1$ を, 実際に計算して予想せよ.

問題 3-3 (数値積分)

積分の定義式:

$$\int_a^b f(x)\,dx = \lim_{\forall i|x_{i+1}-x_i|\to 0} \sum_{i=0}^{n-1} f(x_i)(x_{i+1} - x_i) \tag{5}$$

に従えば, 十分大きな n に対して, 右辺を計算すればそれが積分の近似値となる:

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n-1} f(x_i)(x_{i+1} - x_i). \tag{6}$$

ここで, $a = x_0 < x_1 < \cdots < x_n = b$ である.
関数 f と, 積分区間 a, b, 点の数 n を受け取り, 積分:

$$\int_a^b f(x)\,dx$$

の近似値を, $[a,b]$ を n 等分して式 (6) の右辺を計算することで求める関数 `integrate(f, a, b, n)` を書け.

書けたらそれを使って好きな積分, 例えば

$$\int_0^1 \sqrt{1-x^2}\,dx$$

の近似値を求めよ.

for 文内に，足されている $f(x_i)$ の値を表示する print 文を挿入してみよ (計算結果がおもうようにでない場合の最も基本的な調査手段).

```
def integrate(f, a, b, n):
    ...

def circ(x):
    return math.sqrt(1 - x * x)

integrate(circ, 0.0, 1.0, 1000000)
```

参考: この方法は, 計算機で積分を計算する常套手段で, 手計算の時のように, 原始関数を求めずに実行することができる. 必要なのは任意の点に対して, $f(x)$ が計算できることだけである.

問題 3-4 if 文

どこかで見たシリーズ.

3 辺の長さが a,b,c である三角形が，鋭角三角形であるかどうかを判定する関数 (鋭角三角形であれば 1, そうでなければ 0 となる) `acute_abc(a,b,c)` を書け.

- 注 1: a,b,c の大小は仮定してはいけない.
- 注 2: $(a,b,c)=(5,2,2)$ のように，三角形をなさない場合もあるのでその場合も正しく 0 を返すこと.

これを利用して，複素数 z に対し，複素平面上の 3 点 $\mathrm{A}(1)$, $\mathrm{B}(z)$, $\mathrm{C}(z^2)$ が鋭角三角形をなすかを判定する関数 `acute_z(z)` を書け.

```
def acute_abc(a, b, c):
    ...

def acute_z(z):
    ...
```

注: 原題 (東大 2016 前期入試数学第 4 問) は，`acute_z(z)` が 1 となる範囲を図示せよというもの．後に matplotlib を使って可視化する.

問題 3-5 最小値, for + if の組み合わせ

与えられた関数 g を, 区間 $[a,b]$ において最小とする x を求める Python 関数 `minimize(g, a, b)` を書け. それを利用して,

- 問題 1-2 で定義した関数 `f(x)` を最小にする x を求めよ.
- 問題 2-4 で定義した関数 `S(a)` を最小にする a を求めよ.

注: x を a から b の間で十分細かく動かして, その中で $f(x)$ が一番小さかったときの x を答えれば良い. どれだけ十分細かくすればよいかのきちんとした判断は難しいし, どれだけ細かくしても正しく求まらない関数 (例: 連続でない関数) もある. ここではそのへんの厳密さにはこだわらなくて良い.

```
def minimize(g, a, b):
  ...
```

```
minimize(f, 1e-4, math.pi - 1e-4)  # f : 1-2で定義したもの
 → 1.5707963267948963
```

```
minimize(S, 1 + 1e-4, 3 - 1e-4)  # S : 2-4で定義したもの
 → 2.0
```

問題 3-6 for + if の組み合わせ

(1) 関数 g と 2 数 $a \leq b$ が与えられると, 方程式 $g(x) = 0$ が $[a, b]$ の範囲で解を持てばそのうちの一番小さい解を返し, 持たなければ b より大きな任意の値を返す関数 `min_root(g, a, b)` を書け.

注: 厳密な計算は難しい (一般には不可能). ここでは, 十分小さな h に対して, $g(x)$ と $g(x+h)$ の符号が異なるかまたはどちらかが 0(つまり $g(x) \cdot g(x+h) \leq 0$) であれば, $x+h$ が解であると考える (もちろんこれは近似解). 区間 $[a, b]$ 内で x を少し (h) ずつ動かして上記を計算していく.

(2) `min_root` を用い,

$$x^3 - 3a^2x = b$$

が解を 3 つ持ち, かつ 3 解を $\alpha < \beta < \gamma$ としたとき $\beta > 1$ となるかを判定する (真であれば 1, 偽であれば 0 を返す) 関数 $p(a, b)$ を書け.

クラス 4

問題 4-1 (微分方程式)

f が x の (未知の) 関数で, 以下の式 (微分方程式)

$$y(a) = c, \tag{7}$$
$$\frac{dy}{dx} = \frac{1}{y} \tag{8}$$
$$\tag{9}$$

を満たすとする. この解が $y = \sqrt{2(x-a)+c^2}$ であることは手計算でもすぐにわかる (変数分離法).

以下はそれを計算機にやらせる方法. 一般に,

$$\frac{dy}{dx} = f(x,y) \tag{10}$$

という方程式は, その定義に戻れば,

$$\lim_{h\to 0} \frac{y(x+h)-y(x)}{h} = f(x,y) \tag{11}$$

ということ. つまり, $h \ll 1$ のとき,

$$y(x+h) \approx y(x) + f(x,y)h \tag{12}$$

ということ. 言い換えれば, ある x における y の値 ($y(x)$) がわかれば, $y(x+h)$ が近似できる. そのために必要なものは右辺の $f(x,y)$ だけである. つまり微分方程式の右辺が計算できればよい.

$y(a) = c$ から出発して, $y(a+h), y(a+2h), y(a+3h), \ldots$ を次々に計算していけば, 任意の x における $y(x)$ (の近似値) を計算することができる.

与えられた関数 f, 実数 a, b, c に対し,

$$y(a) = c, \tag{13}$$
$$\frac{dy}{dx} = f(x,y) \tag{14}$$
$$\tag{15}$$

を満たす y の $x = b$ における値 $y(b)$ の近似値を計算する関数 `ord_solve(f, a, b, c)` を書け.

それを用いて, 最初にあげた微分方程式を解き, $y(2.5)$, $y(5.0)$ などを計算してみよ. それを厳密解と比較してみよ.

```
def f(x, y):
    return 1.0 / y

ord_solve(f, 1.0, 2.5, 1.0)
ord_solve(f, 1.0, 5.0, 1.0)
```

なおこの考え方は高階の微分方程式でも容易に適用できる. x が t の未知の関数とする. このとき, 二階の微分方程式の一般形は以下の形である.

$$x(a) = c, \tag{16}$$
$$\dot{x}(a) = d, \tag{17}$$
$$\ddot{x} = f(x, \dot{x}, t) \tag{18}$$

これを数値計算で解くには, $x(a) = c, \dot{x}(a) = d$ から出発して,

$$\dot{x}(t+h) = \dot{x}(t) + \ddot{x}(t)h, \tag{19}$$
$$x(t+h) = x(t) + \dot{x}(t)h, \tag{20}$$

の二つを使って, x と $\dot{x}$ を求めていく.

クラス 5

この節の問題はどれも質点の運動をシミュレートし，それができたら Visual Python で可視化（アニメーション）を行う．

共通事項:

- まずは，質点の位置や速度を時々(一定時間や一定繰り返し回ごとに) `print` で，数字として表示する関数を書け．明らかな間違いを発見するために重要．
- それが書けたら `print` する代わりに，Visual Python を使ってアニメーションせよ．
- アニメーションをする際，質点だけだと，カメラの自動追従機能によって，物体が動いているように見えない．質点の近くに適当な静止物体（例えば板）をおいて，動きがわかるようにすると良い．
- アニメーションを表示するためには物体が動くたびに `rate(`r`)` という関数を呼ぶ．これを呼ばないと，画面を更新してくれないので，途中経過が表示されず，最終的な位置だけが突如表示されることになってしまう．`rate(`r`)` は，画面を更新するとともに，`rate` が秒間 r 回程度呼ばれるよう，中で時間調整をする．例えば `rate(30)` は，$1.0/30 = 0.03333\cdots$ 秒程度の時間調整を行う．アニメーションが実時間に合わせて進行するようにしたいのであれば，シミュレーション上 h だけ時間が経過したら，`rate(`$1/h$`)` を呼びだせばよい．
- シミュレーションをやりっぱなしでその結果を鵜呑みにするわけにはいかない．出した結果があっていそうか，確かめるため，できることをやってみよ．アニメーション自体が役にたつが他にも手はある．
  - どうなるはずかがわかりやすい設定でシミュレーションしてみる
  - 成り立つはずの物理法則（保存則）が成り立っているか計算してみる

  など

問題 5-1 (微分方程式 + 可視化 (1))

重力加速度を受けて運動する質点の動きをシミュレーションする関数 `point_mass(`x_0`,` v_0`,` T`)` を書け．

- T はシミュレートする時間 (時刻 0 から T まで)．
- x_0, v_0 は初期位置，初速とする．

チェック項目と発展課題:

- ボールを何度で打ち出したら一番遠くまで飛ぶか，という答えのよく知られた問題があるが，本当にそうなるか確かめてみよ
- 空気抵抗 (速度に比例して速度の反対向きに加わる力) を加えて見よ．その場合，一番遠くまで飛ぶ角度はどうなるか?
- 地面や壁があって衝突によって跳ね返るという状況をアニメーションしてみよ

問題 5-2 (微分方程式 + 可視化 (2))

万有引力で引き合って運動する 2 つの質点の動きを，シミュレーションする関数 `point_mass2(`m_0`,` x_0`,` v_0`,` m_1`,` x_1`,` v_1`,` T`)` を書け．

- T はシミュレートする時間 (時刻 0 から T まで)．
- m_0, x_0, v_0 は質点 0 の質量，初期位置，初速度，
- m_1, x_1, v_1 は質点 1 のそれらとする．

初期状態や質量を変えて以下のような状況をシミュレートしてみよ．

- 太陽と地球
- 太陽と彗星
- 連星

問題 5-3 (微分方程式 + 可視化 (3))

バネで吊り下げられた質点の動きをシミュレーションする関数 `spring_mass`(m, k, y_0, T) を書け.

- T はシミュレートする時間 (時刻 0 から T まで).
- m は質量，k はバネ定数.
- y_0 は初期位置の y 座標とする．但しバネの自然長の位置を 0 とする．
- 初速は 0, 天井の位置は適当でよい（見た目以外に影響はないので）

チェック項目と発展課題:

- バネの周期を正しく計算できているか
- 空気抵抗を入れて減衰振動にしてみよ

クラス 6

問題 6-1 (リスト内包表記)

リスト内包表記を使って以下の関数を, なるべく短くきれいに書いてみよう.

1. a, b, n が与えられ，$[a,b]$ を $(n-1)$ 等分する点を (端点 a,b を含み n 点) リストにしたものを返す関数 `linspace(`a`,` b`,` n`)`.

```
linspace(1, 2, 5)
 → [1.0, 1.25, 1.5, 1.75, 2.0]
```

2. n が与えられ

$$1^3 + 2^3 + \cdots + n^3$$

を返す関数 `sum_cube(`n`)`. 以前はループで以下のように書いたものだが, リスト内包表記を使うと?

```
def sum_cube(n):
    s = 0
    for i in range(1, n+1):
        s = s + i ** 3
    return s
```

ヒント: リスト L の要素の和を返す関数は `sum(`L`)`

3. 関数 f と，a, b が与えられ，積分

$$\int_a^b f(x)\,dx$$

の近似値を計算する関数．これも以前書いたがリスト内包表記を使うと?

4. 関数 f と，a, b が与えられ,

$$\min_{a \le x \le b} \{\ f(x)\ \}$$

を求める関数 `minimize(`f`,` a`,` b`)`.

ヒント: リスト L の最小値を返す関数は `min(`L`)`

```
def q(x):
    return x * x - 4 * x + 3

minimize(q, a, b)
 → -1.0
```

5. 前問で，f を最小にする x の値も一緒に返す関数 `minimize_arg(`f`,` a`,` b`)` にするには?

ヒント: タプルをうまく使う．Python では二つのタプルの大小関係は，いわゆる「辞書式順序」と定義されている．

$$(a,b) \le (c,d) \iff a < c \text{ または } a = c \text{ かつ } b \le d$$

また，`min` はタプルのリストに対しても働く.

```
def q(x):
    return x * x - 4 * x + 3

minimize_arg(q, a, b)
 → (2.0,-1.0)
```

6. 同じ長さのリスト X, Y を受け取り，それぞれをベクトルとみなした時の内積を返す関数.

ヒント： zip
動作例:

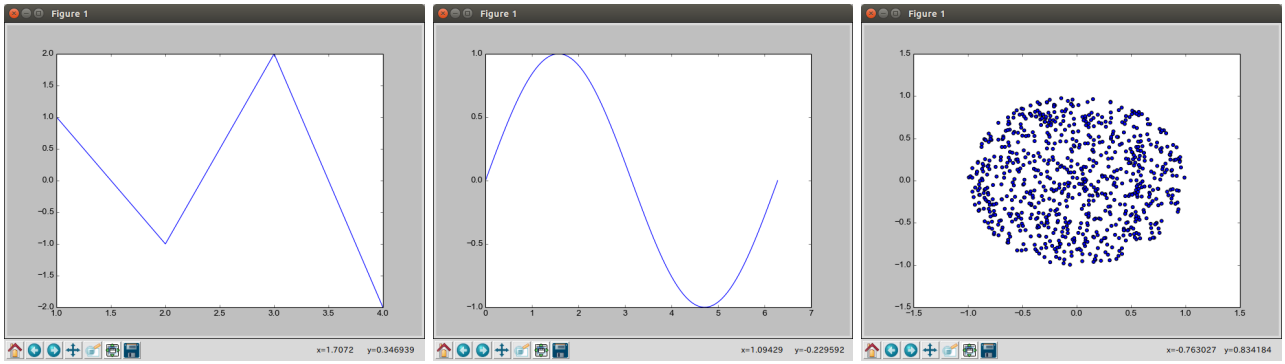
```
dot([1,2,3,4], [4,5,6,7])
 → 60
```

クラス 7

問題 7-1　matplotlib (1 変数の関数)

以下を matplotlib を使って描画してみよ

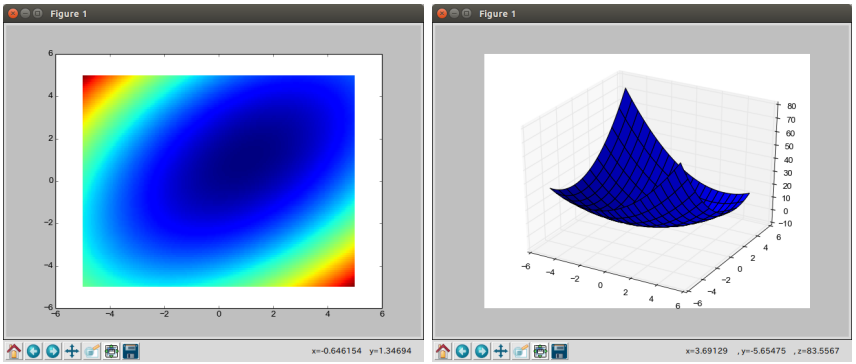
1. 最もプリミティブな確認. リストだけを使って，点 $(1,1)$, $(2,4)$, $(3,9)$, $(4,16)$ をつないだグラフ
2. $y = \sin(x)$ のグラフを $0 \leq x \leq 2\pi$ の範囲で (numpy の機能をかっこよく使って).
3. $x^2 + y^2 \leq 1$ の範囲にランダムにばらまかれた点 (plot の代わりに scatter).



問題 7-2　matplotlib (2 変数の関数)

以下を matplotlib を使って描画してみよ

1. $z = x^2 - xy + y^2 - x - y$ を，$-5 \leq x \leq 5$, $-5 \leq y \leq 5$ の範囲で，色で (データの与え方はスライドもしくは cheet sheet 参照).
2. 前問題と同じ関数を，ただし 3D の曲面で



クラス 8

問題 8-1　場の方程式

x と t の関数 $u(x,t)$ に関する偏微分方程式:

$$\begin{aligned}
\frac{\partial u}{\partial t} &= 2\frac{\partial u}{\partial x} \quad (-5 < x < 5) \\
u(x,0) &= \begin{cases} x+1 & (-1 \le x \le 0) \\ -x+1 & (0 \le x \le 1) \\ 0 & (|x| > 1) \end{cases} \\
u(0,t) &= 0 \\
u(1,t) &= 0
\end{aligned}$$

をシミュレーションしたい．

1. 初期値 $u(x,0)$ を matplotlib を用いて plot せよ
2. 微分方程式を $t = 2$ までシミュレーションし，結果を matplotlib を用いて plot せよ

なお，この方程式は手でも解ける．任意の一変数関数 $g(y)$ に対し，

$$u(x,t) = g(x+2t)$$

とすれば，

$$\frac{\partial u}{\partial t} = 2\frac{\partial u}{\partial x}$$

が満たされる．$t = 0$ を代入すると，

$$u(x,0) = g(x)$$

すなわち，g は初期値そのものである．つまり，$u(x,t)$ は，初期値を $-2t$ だけ平行移動したものにすぎない．すなわち左方向に速さ 2 で平行移動していくだけである．

問題 8-2　場の方程式．2 次元

u は 2 次元の座標 (x,y) と時刻 t の関数で以下を満たす．

$$\begin{aligned}
\frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \quad (0 < x < 1,\, 0 < y < 1) \\
u(x,y,0) &= \text{適当 ([0,1] の乱数)} \quad (0 < x < 1,\, 0 < y < 1) \\
u(x,0,t) &= x \\
u(1,y,t) &= 1 \\
u(x,1,t) &= 1 \\
u(0,y,t) &= y
\end{aligned}$$

をシミュレーションしたい．

1. 初期状態を matplotlib を使って色で可視化 (pcolor) せよ
2. 十分大きな t までシミュレートした結果を，を matplotlib を使って可視化せよ