公式サイトから資料を
DLできます。
パスワード: pk_py1130
puzzleknot.wp.xdomain.jp

puzzleknot
Tohoku University Programming Club

やってみよう 1の解答例

```
def TriangleArea(x1, y1, x2, y2, x3, y3):
    S = (1/2) * abs((x1 - x3)*(y2 - y3) - (x2 - x3)*(y1 - y3))
    return S
```

2

やってみよう 2の解答例

```
def TriangleArea(x1, y1, x2, y2, x3, y3):
    S = (1/2) * abs((x1 - x3)*(y2 - y3) - (x2 - x3)*(y1 - y3))
    return S


P = [5, 7, 3, 4, 1, 2]
S = TriangleArea(P[0], P[1], P[2], P[3], P[4], P[5])
print(S)
```

3

やってみよう 3の解答例

```
def TriangleArea(x1, y1, x2, y2, x3, y3):
    S = (1/2) * abs((x1 - x3)*(y2 - y3) - (x2 - x3)*(y1 - y3))
    return S

Points = [(5, 7, 3, 4, 1, 2),
          (3, 1, 4, 1, 5, 9),
          (2, 6, 5, 3, 5, 8),
          (9, 7, 9, 3, 2, 3)]

for P in Points:
    S.append(TriangleArea(P[0], P[1], P[2], P[3], P[4], P[5]))

print(S)
```

4

補足: listの入れ子

- listやtuple、dict、setなどはその要素としてlistなどを持つことが可能です

```
Points = [(5, 7, 3, 4, 1, 2),
          (3, 1, 4, 1, 5, 9),
          (2, 6, 5, 3, 5, 8),
          (9, 7, 9, 3, 2, 3)]
```

- これはlistの中にtupleが入れ子になっています

5

補足: listの入れ子

```
Points = [(5, 7, 3, 4, 1, 2),
          (3, 1, 4, 1, 5, 9),
          (2, 6, 5, 3, 5, 8),
          (9, 7, 9, 3, 2, 3)]
```

- 各要素には Points[1][2] のような形でアクセスできます
- なぜでしょう？

6

補足: listの入れ子

```
Points = [(5, 7, 3, 4, 1, 2),
          (3, 1, 4, 1, 5, 9),
          (2, 6, 5, 3, 5, 8),
          (9, 7, 9, 3, 2, 3)]
```

Points[1][2]

- Points[1]は(3, 1, 4, 1, 5, 9) というtuple

7

補足: listの入れ子

```
Points = [(5, 7, 3, 4, 1, 2),
          (3, 1, 4, 1, 5, 9),
          (2, 6, 5, 3, 5, 8),
          (9, 7, 9, 3, 2, 3)]
```

Points[1][2]

- この後ろについた[2]は、その2番目の要素を指すことになる
  （0から数えることに注意）

8

補足: listの入れ子

- Twitterのbotを作る際、dictの中にlistが入っていたり、その逆のものが登場したりします
- この感覚に慣れておきましょう

9

やってみよう 4の解答例

```
import math

x = []
i = 0
while(i <= 2*math.pi):
    x.append(i)
    i += 0.1

for ang in x:
    print(math.sin(ang))
```

10

補足: 小数の演算

- コンピュータで扱われる小数は、浮動小数点数と呼ばれるものです
  （興味があれば調べてみてください）
- コンピュータはすべてのデータを2進数で扱っているので、当然10進数の小数は2進数になることになります

11

補足: 小数の演算

- 10進数では循環しない小数も、2進数では循環することがあります
  (e.g. 0.1は
  0.0001 1001 1001 1001 1001 1001…
  となります)
- このため、このような数は正確に表現することができません

12

補足: 小数の演算

- 解答例のコードによって生成された list xは以下のように微妙に誤差があります

```
[0, 0.1, 0.2, 0.3000000000000004, 0.4, 0.5, 0.6, 0.7,
0.7999999999999999, 0.8999999999999999, 0.9999999999999999,
1.0999999999999999, 1.2, 1.3, 1.4000000000000001,
1.5000000000000002, 1.6000000000000003, 1.7000000000000004,
1.8000000000000005, 1.9000000000000006, 2.0000000000000004,
2.1000000000000005, 2.2000000000000006, 2.3000000000000007,
2.40000000000000001, 2.50000000000000001, 2.60000000000000001,
2.70000000000000001, 2.80000000000000001, 2.90000000000000012,
3.0000000000000013, 3.1000000000000014, …]
```

- 小数の扱いには気をつけましょう

13

補足: 小数の演算

- [高度] decimal モジュールをimportすることで、限りなく正確に近い演算が可能です

```
[0, Decimal('0.1'), Decimal('0.2'), Decimal('0.3'),
Decimal('0.4'), Decimal('0.5'), Decimal('0.6'),
Decimal('0.7'), Decimal('0.8'), Decimal('0.9'),
Decimal('1.0'), Decimal('1.1'), Decimal('1.2'),
Decimal('1.3'), Decimal('1.4'), Decimal('1.5'),
Decimal('1.6'), Decimal('1.7'), Decimal('1.8'),
Decimal('1.9'), Decimal('2.0'), Decimal('2.1'),
Decimal('2.2'), Decimal('2.3'), Decimal('2.4'),
Decimal('2.5'), Decimal('2.6'), Decimal('2.7'),
Decimal('2.8'), Decimal('2.9'), Decimal('3.0'),
Decimal('3.1'), Decimal('3.2'),…
```

14

補足: 小数の演算

- [高度] コード例

```
import math
import decimal

x = []
i = 0
while(i <= 2*math.pi):
    x.append(i)
    i += decimal.Decimal('0.1')

for ang in x:
    print(ang, end=" ")
    print(math.sin(ang))
```

15

補足: モジュールの作成

- モジュールは自分で作成が可能です
- 関数名や定数をまとめたファイルを作成し、(作りたいモジュール名).pyで保存します
- 自作モジュールは基本的に、実行したいスクリプトと同じ場所に置きましょう

16

補足: モジュールの作成

- 例: Areaというモジュールを作成
- Area.py （モジュール）

```
def TriangleArea(x1, y1, x2, y2, x3, y3):
    S = (1/2) * abs((x1 - x3)*(y2 - y3) - (x2 - x3)*(y1 - y3))
    return S
```

17

補足: モジュールの作成

- 例: Areaというモジュールを作成
- 呼び出す側のスクリプト

```
import Area

S = Area.TriangleArea(5, 7, 3, 4, 1, 2)
print(S)
```

18

補足: モジュールの作成

- あるスクリプトの名前として、既存のモジュール名を利用し、別の場所でそのモジュールをimportしようとすると、代わりにそのスクリプトがimportされてしまいます
- モジュールをimportする際には、呼び出す側のスクリプト名にも気を遣ってください
(**importするモジュールの名前にしない**)

19

numpyを利用した行列の計算例

```
import numpy as np

a = numpy.asarray([1, 2, 3], [4, 5, 6])
b = numpy.asarray([1, 2], [3, 4], [5, 6])

print(a)
print(b)
print(a.dot(b)) # a・b

c = np.asarray([[-1, 2, 3], [4, 5, 6], [7, 8, 9]])

print(c)
print(np.linalg.det(c)) # det(c)
```

20

補足: モジュールに別名をつける

- importするモジュールには、別名(エイリアス) をつけることが可能です
- **as** を使います

```
import numpy as np
```

- numpy.asarray と書いていた部分は np.asarray と書き換えられます (逆に、numpyのままではエラーになります)

21