Bài Thang đo nhiệt độ KFC

- Với đầu vào là độ F, yêu cầu của đề bài là chuyển sang độ C và K. Chúng ta chỉ cần sử dụng 2 công thức đơn giản là: $c = (f - 32)/1.8$ và $k = c + 273.15$
- Tuy nhiên, một vấn đề gây khó khăn cho bài này đó là độ chính xác (**precision**). Lý do chính ở đây là testcase của bài này được generate từ code C++ với kiểu dữ liệu float có khả năng biểu diễn số thực với precision từ 6-9 chữ số (**significant digits**).
- Thêm vào đó, câu lệnh xuất của C++ (e.g.: cout) có precision mặc định là 6 chữ số. Tức là, nó giả định rằng toàn bộ biến kiểu số thực chỉ quan trọng đến 6 significant digits và sẽ cắt bỏ toàn bộ chữ số còn lại phía sau.
- Trong khi đó, số thực trong Python lại có khả năng biểu diễn với precision cao hơn.
- Có thể dùng package decimal để đặt lại precision thành 6 significant digits.
- Ở đây, đối với các trường hợp in thừa các số không phía sau (e.g.: 1.23400), ta dùng **normalize()** để bỏ đi các số không thừa này (e.g.: 1.234).
- Lệnh **input()** nhập một dòng dữ liệu từ thiết bị nhập chuẩn cho kết quả là xâu ký tự
- Lệnh **print(*e*)** xuất dữ liệu từ biểu thức *e* ra thiết bị xuất chuẩn

```
# làm quen với Decimal
from decimal import *

# vì sao kết quả trả về là False?
print((1.1 + 2.2) == 3.3)
print(0.1 + 0.1 + 0.1 - 0.3 == 0)

# lí do là biểu diễn số thực liên quan đến số chữ số thập phân
print("Float representation: ", 1.1 + 2.2)
print("Float representation: ", 0.1 + 0.1 + 0.1 - 0.3)

# in ra context
print(getcontext())

print("Float val:", float(1/7))

# The significance of a new Decimal is determined solely by the number of digits input.
# Context precision and rounding only come into play during arithmetic operations.

# đặt số chữ số thập phân về 6
getcontext().prec = 6

# prec không được áp dụng ở đây
print("Decimal object construction:", Decimal(float(1/7)))

# khi tính toán - prec sẽ được áp dụng
print("Decimal object normalize:", Decimal(float(1/7)).normalize())
print("Decimal object arithmetic operation:", Decimal(float(1/7))+ Decimal(0))
print("Using decimal evaluation:", Decimal(1) / Decimal(7))

# hãy thử lại đoạn code trên với giá trị prec lớn, ví dụ 30
# để thấy dùng Decimal trước hay sau sẽ có kết quả khác nhau
```

```
#getcontext().prec = 30
```

```
from decimal import *

f = float(input())
c = (f - 32) /1.8
k = c + 273.15

getcontext().prec = 6

print(Decimal(c).normalize(), Decimal(k).normalize())
```

Bài Độ đo áp suất SI/PSI

- Sử dụng phép toán $pound * (0.453592/2.54^2)$ để đổi từ đơn vị $pound/inch^2$ sang đơn vị kg/cm^2
- Việc xử lý precision sẽ tượng tự câu trên

```
from decimal import *

psi = float(input())

kgscm = (psi*0.453592) / (2.54 * 2.54)

getcontext().prec = 6

print(Decimal(kgscm).normalize())
```

```
15
1.05460
```

Bài Gà và chó

Gọi x, y lần lượt là số con gà, chó.

Ta có xxx là tổng số con $\Rightarrow x + y = xxx$.

yyy là tổng số chân $\Rightarrow 2x + 4y = yyy$.

Giải hệ phương trình ta được:

$x = (4xxx - yyy)/2$

$y = xxx - x$

Lưu ý:

1. phép chia "/" trong python sẽ trả về kiểu float. Do đó, để có kết quả là số nguyên ta sử dụng phép chia nguyên "//" .
2. Sử dụng phương thức **split()** để tách riêng các dữ liệu được nhập trên cùng dòng.

3. Phương thức **map(*type, obj*)** sẽ chuyển kiểu dữ liệu của tất cả thành phần trong *obj* về kiểu

```
xxx,yyy = map(int, input().split())
x = (4*xxx - yyy) // 2
y = xxx - x
print(x, y)
```

```
36 100
22 14
```

Bài Phân loại tam giác

Gọi a, b, c lần lượt là độ dài 3 cạnh.

- Điều kiện để là một tam giác: Tổng độ dài 2 cạnh bất kì phải lớn hơn độ dài cạnh còn lại. Hay $a < b + c$ và $b < a + c$ và $c < a + b$.
- Điều kiện là một tam giác vuông. Áp dụng định lí Pytago ta có: $a^2 = b^2 + c^2$ hoặc $b^2 = a^2 + c^2$ hoặc $c^2 = a^2 + b^2$.
- Điều kiện là một tam giác đều: $a = b = c$.
- Điều kiện là một tam giác cân: $a = b$ hoặc $b = c$ hoặc $a = c$.

Lưu ý: Do tam giác đều là trường hợp đặc biệt của tam giác cân nên ta kiểm tra tam giác đều trước.

Để tính diện tích tam giác khi biết độ dài 3 cạnh, ta áp dụng công thức Heron:
$S = \sqrt{p(p-a)(p-b)(p-c)}$

Với p là nửa chu vi của tam giác: $p = \frac{(a+b+c)}{2}$

*Lưu ý: Sử dụng hàm **sqrt()** trong thư viện math để tính căn bậc hai. Để ý trường hợp khi diện tích ra số nguyên nhưng hàm làm tròn **round()** sẽ trả về số thực.*

```
from  math import sqrt
a = int(input())
b = int(input())
c = int(input())

if a<b+c and b<a+c and c<a+b:
    #Tinh dien tich
    p = (a+b+c)/2
    s = round(sqrt(p*(p-a)*(p-b)*(p-c)),2)
    if s == int(s):
        s = int(s)
    #Kiem tra tam giac
    if a*a==b*b+c*c or b*b==a*a+c*c or c*c== a*a+b*b:
        print("Tam giac vuong, dien tich = {}".format(s))
    elif a==b==c:
        print("Tam giac deu, dien tich = {}".format(s))
    elif a==b or b==c or a==c:
        print("Tam giac can, dien tich = {}".format(s))
    else:
```

```
        print("Tam giac thuong, dien tich = {}".format(s))
else:
    print("Khong phai tam giac")
```

Bài Bài kiểm tra

Để dễ xử lý, ta giả định các bàn trong lớp học được xếp cạnh nhau theo chiều ngang và mỗi bàn có 2 **chỗ ngồi**, các chỗ ngồi này sẽ được đánh số bắt đầu từ 0 và tăng dần. Với trường hợp bài kiểm tra có k (e.g.: $k = 2$) đề khác nhau, thì lớp học sẽ được biểu diễn như hình dưới:

Chỉ số chỗ ngồi	0	1	2	3	4	5	6	7	...
Mã đề	1	2	1	2	1	2	1	2	...

Giả sử, Alice đang ngồi ở chỗ có chỉ số 4, thì 2 chỗ ngồi có cùng đề và gần Alice nhất là 2 (tức $4 - k$) và 6 (tức $4 + k$).

Bây giờ, việc ta cần làm là chuyển cách tổ chức bàn học trong đề bài sang cách tổ chức nói trên. Gọi p là dãy bàn mà Alice đang ngồi, q (bằng 1 hoặc 2) là vị trí bàn, k là chỉ số chỗ ngồi tương ứng theo cách tổ chức mới. Lúc này, $k = (p - 1) * 2 + q$. Ngược lại, $p = k\ div\ 2 + 1$ và $q = k\ mod\ 2 + 1$.

```
n = int(input())
k = int(input())
p = int(input())
q = int(input())

ith = (p-1) * 2 + (q-1)

before = ith - k
after = ith + k

if before >= 0:
    print(before//2+1, before%2+1)
elif after <= n-1:
    print(after//2+1, after%2+1)
else:
    print(-1)
```

Bài Đường sắt trên cao

Đối với bài tập này, cần chú ý đến việc đoàn tàu có 2 hướng chạy: hướng đi và hướng về. Để xác định đoàn tàu đang chạy theo hướng đi hay hướng về, ta sẽ kiểm tra tính chẵn lẻ của phép toán $t\ div\ k$. Nếu $t\ div\ k$ chẵn suy ra đoàn tàu đang chạy theo hướng đi, ngược lại đang theo hướng về. Nếu đoàn tàu đang chạy theo hướng đi thì ga hiện tại sẽ bằng $t\ mod\ k$, ngược lại sẽ là $k - t\ mod\ k$

Ví dụ này sử dụng một cách khác, không sử dụng hàm **map**, để chuyển tất cả kiểu các thành phần dữ liệu của một đối tượng

```
k, t = [int(x) for x in input().split()]

if (t // k) % 2 == 0:
    print(t % k)
else:
    print(k - t % k)
```

Bài Tổng ước số

Để tính tổng ước số của n. Ta lần lượt kiểm tra các ứng viên ước số i bằng cách xem xét n/i (i từ 1 ➜ n), nếu n chia hết cho i thì ta sẽ cộng vào tổng. Ngược lại thì không.

Mà n không chia hết cho các số thuộc nửa đoạn $[n/2+1, n)$ nên chỉ cần xem xét i từ 1 ➜ $n/2$.

```
n = int(input())
s = 0
for i in range(1,n//2 + 1):
    if not n%i:
        s = s + i
print(s)
```

Bài Số Fibonacci

Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng 1 và 1, sau đó các số tiếp theo sẽ bằng tổng của 2 số liền trước nó.

Công thức truy hồi của dãy Fibonacci:

$$\boldsymbol{F(n) := \begin{cases} 1, & \text{khi } n = 1; \\ 1, & \text{khi } n = 2; \\ F(n-1) + F(n-2) & \text{khi } n > 2. \end{cases}}$$

Cụ thể, các số đầu tiên của dãy Fibonacci là 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610...

Gọi f là số fibonacci thứ n cần tìm. Đặt $f_1 = 1$, $f_2 = 1$ lần lượt là số fibonacci thứ 1 và 2.

Số fibonacci thứ 3: $f = f_1 + f_2 = 2$

Số fibonacci thứ 4: $f = f_2 + f_3$. Nếu ta gán $f_1 = f_2$ và $f_2 = f$ thì $f = f_1 + f_2 = 3$.

Vậy, để tìm số fibonacci thứ n ta lặp $n - 2$ lần.

$f = f_1 + f_2$

$f_1 = f_2$

$f_2 = f$

Sau khi kết thúc vòng lặp, f là kết quả cần tìm.

```
n = int(input())
f1 = f2 = f= 1
if 1 <= n <= 30:
    for i in range(n-2):
        f = f1 + f2
        f1 = f2
        f2 = f
    print(f)
else:
    print("So {} khong nam trong khoang [1,30].".format(n))
```