Chapter 1

Introduction

Chapter 1
Introduction to Python

In This Chapter:

1. Say "Hello, World!" With Python
2. Reading Raw Input
3. Python If-Else
4. Arithmetic Operators
5. Python: Division
6. Loops
7. Write a function
8. Print Function

1.1 Say "Hello, World!" With Python

Here is a sample line of code that can be executed in Python:

```
print("Hello, World!")
```

You can just as easily store a string as a variable and then print it to stdout:

```
my_string = "Hello, World!"
print(my_string)
```

The above code will print `Hello, World!` on your screen. Try it yourself in the editor below!

Input Format

You do not need to read any input in this challenge.

Output Format

Print `Hello, World!` to stdout.

Sample Output

```
Hello, World!
```

Code

```
if __name__ == '__main__':
    print "Hello, World!"
```

1.2 Reading Raw

Read a line of input from stdin and save it to a variable, s. Then print the contents of s to stdout.

Input Format

A single line containing sentence.

Constraints

$1 \leq |s| \leq 500$

Output Format

Print the contents of s to stdout.

Sample Input

How many chickens does it take to cross the road?

Sample Output

How many chickens does it take to cross the road?

Code

```
def read():
    s = raw_input()
    return s
```

1.3 Python If-Else

Given an integer,n, perform the following conditional actions:

- If n is odd, print Weird
- If n is even and in the inclusive range of 2 to 5, print Not Weird
- If n is even and in the inclusive range of 6 to 20, print Weird
- If n is even and greater than 20, print Not Weird

Input Format

A single line containing a positive integer,n .

Constraints

$1 \leq n \leq 100$

Output Format

Print Weird if the number is weird; otherwise, print Not Weird.

Sample Input 0

3

Sample Output 0

Weird

Sample Input 1

24

Sample Output 1

Not Weird

Code

```
if __name__ == '__main__':
    n = int(raw_input())

    if n % 2 == 1:
        print 'Weird'
    elif n in range(2,5+1):
        print 'Not Weird'
    elif n in range(6,20+1):
        print 'Weird'
    else:
        print 'Not Weird'
```

1.4 Arithmetic Operators

Read two integers from STDIN and print three lines where:

1. The first line contains the sum of the two numbers.
2. The second line contains the difference of the two numbers (first - second).
3. The third line contains the product of the two numbers.

Input Format

The first line contains the first integer,a . The second line contains the second integer,b .

Constraints

$1 \leq a \leq 10^{10}$

$1 \leq b \leq 10^{10}$

Output Format

Print the three lines as explained above.

Sample Input

```
3
2
```

Sample Output

```
5
1
6
```

Code

```
# Enter your code here. Read input from STDIN.

a=int(raw_input())
b=int(raw_input())
print a+b
print a-b
print a*b
```

1.5 Python: Division

Read two integers and print two lines. The first line should contain integer division, a // b . The second line should contain float division, a / b.

You don't need to perform any rounding or formatting operations.

Input Format

The first line contains the first integer,a . The second line contains the second integer, b.

Output Format

Print the two lines as described above.

Sample Input

```
4
3
```

sample Output

```
1
1.3333333333333333
```

Code

```
from __future__ import division
if __name__ == '__main__':
    a = int(raw_input())
    b = int(raw_input())
    print (a//b)
    print (a / b)
```

1.6 Loops

Read an integer N. For all non-negative integers i < N, print i^2. See the sample for details.

Input Format

The first and only line contains the integer,N .

Constraints

$1 \leq N \leq 20$

Output Format

Print N lines, one corresponding to each i.

Sample Input

```
5
```

Sample Output

```
0
1
4
9
16
```

Code

```
# Enter your code here. Read input from STDIN.
N=int(raw_input())
for i in range(0,N):
    print pow(i,2)
```

1.7 Write a function

We add a Leap Day on February 29, almost every four years. The leap day is an extra, or intercalary, day and we add it to the shortest month of the year, February.
In the Gregorian calendar three criteria must be taken into account to identify leap years:

- The year can be evenly divided by 4;
- If the year can be evenly divided by 100, it is NOT a leap year, unless;
- The year is also evenly divisible by 400. Then it is a leap year.

This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years.

You are given the year, and you have to write a function to check if the year is leap or not.

Input Format

Read y, the year that needs to be checked.

Constraints

$1900 \leq y \leq 10^5$

Output Format

Output is taken care of by the template. Your function must return a boolean value (True/False)

Sample Input

```
1990
```

Sample Output

```
False
```

Code

```
def is_leap(year):
    leap = False
    if year % 400 == 0:
        leap = True
    elif year % 100 == 0:
        leap = False
    elif year % 4 == 0:
        leap = True
    return leap


year = int(raw_input())
print is_leap(year)
```

1.8 Print Function

In Python 2, the default print is a simple IO method that doesn't give many options to play around with.

The following two examples will summarize it.

Example 1:

```
var, var1, var2 = 1,2,3
print var
print var1, var2
```

Prints two lines and, in the second line, var1 and var2 are separated by a single space.

Example 2:

```
for i in xrange(10):
    print i,
```

Prints each element separated by space on a single line. Removing the comma at the end will print each element on a new line.
Let's import the advanced print_function from the __future__ module.

Its method signature is below:

```
print(*values, sep=' ', end='\n', file=sys.stdout)
print(value1, value2, value3, sep=' ', end='\n', file=sys.stdout)
```

Here, values is an array and *values means array is unpacked, you can add values separated by a comma too. The arguments *sep*, *end*, and *file* are optional, but they can prove helpful in formatting output without taking help from a string module.
The argument definitions are below:
sep defines the delimiter between the values.
end defines what to print after the values.
file defines the output stream.
in Python 2 print_function is much faster than the default print
Your task is to read an integer N.

Without using any string methods, try to print the following:

123…N

Note that "…" represents the values in between.

Input Format

The first line contains an integer N.

Output Format

Output the answer as explained in the task.

Sample Input

```
3
```

Sample Output

```
123
```

Code

```
# Enter your code here. Read input from STDIN. Print output to STDOUT

from __future__ import print_function

[print(x, end='') for x in xrange(1, int(raw_input()) + 1)]
```