Chapter 3: `if` statements

Author: Vincent Delecroix

Comparisons and `if`

The comparison signs in Python and many other programming languages are as follows

==	equality
!=	difference
<	less than
>	greater than
<=	lesser than or equal to
>=	greater than or equal to

Exercise 3.1

Which number is the largest 1000^{1001} or 1001^{1000}?

```
sage: 1000^1001 > 1001^1000
True
```

Exercise 3.2

Let us consider the following code:

```
sage: a = # enter a value for a
....: if a != 2:
....:     print('lost')
....: elif a == 3:
....:     print('an instant, please')
....: else:
....:     print('you win')
```

What is the above program doing

- when the variable a is 1?

```
sage: a = 1# enter a value for a
....: if a != 2:
....:     print('lost')
....: elif a == 3:
....:     print('an instant, please')
....: else:
....:     print('you win')
```

```
lost
```

- when the variable a is 2?

```
sage: a = 2# enter a value for a
....: if a != 2:
....:     print('lost')
....: elif a == 3:
....:     print('an instant, please')
....: else:
....:     print('you win')
you win
```

- when the variable a is 3?

```
sage: a = 3# enter a value for a
....: if a != 2:
....:     print('lost')
....: elif a == 3:
....:     print('an instant, please')
....: else:
....:     print('you win')
lost
```

- when the variable a is 15?

```
sage: a = 15# enter a value for a
....: if a != 2:
....:     print('lost')
....: elif a == 3:
....:     print('an instant, please')
....: else:
....:     print('you win')
lost
```

Exercise 3.3

Two prime numbers p and q are said *twin* if $q = p+2$. Find all twin prime numbers below 10000.

```
sage: TwinPrimeNumbers = [];
....: for p in prime_range(2,10000):
....:     if (p+2).is_prime():
....:         TwinPrimeNumbers.append((p,p+2))
....: TwinPrimeNumbers
```

```
sage: len(TwinPrimeNumbers)
205
```

Exercise 3.4

Find the smallest and largest integers in the set

$$\{a^b - b^a : a \in \{1,2,\dots,5\}, b \in \{1,2,\dots,5\}\}$$

```
sage: min([a^b-b^a for a in range(6) for b in range(6)])
-399
```

```
sage: max([a^b-b^a for a in range(6) for b in range(6)])
399
```

Exercise 3.5

Recall that the method `digits` of an integer returns the list of its digits:

```
sage: 1527.digits()
[7, 2, 5, 1]
```

Solve Euler problem 56 by finding the maximal sum of digits of numbers of the form a^b with both a and b lesser than 100

```
sage: MaximalSumDigits = 0;
....: for a in srange(100):
....:     for b in srange(100):
....:         Sum = sum((a^b).digits())
....:         if MaximalSumDigits < Sum:
....:             MaximalSumDigits = Sum
....: MaximalSumDigits
972
```

Exercise 3.6

Solve Euler problem 4 about palindromes. A palindromic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is $9009 = 91 \times 99$.

Find the largest palindrome made from the product of two 3-digit numbers.

```
sage: LargestPalindrome = 0;
....: for a in srange(100,1000):
....:     for b in srange(a,1000):
....:         if Word((a*b).digits()).is_palindrome():
....:             LargestPalindrome = max(LargestPalindrome,a*b)
....: LargestPalindrome
906609
```

Exercise 3.7

Let us consider the following list of integers:

```
sage: l = [123, 414, 264, 18, 689, 21, 5571, 28, 589, 12, 111, 231,
....: 158, 551, 250, 68, 5728, 2222, 4198, 571, 28, 518, 999, 444,
....: 112, 689, 672, 334, 680, 273]
```

Construct two lists `leven` and `lodd` that contain respectively the even and odd elements of `l`.

```
sage: leven = []; lodd=[]
....: for x in l:
....:     if x%2 == 0:
....:         leven.append(x)
....:     else:
....:         lodd.append(x)
```

Using `in` and `not in`

The condition of an `if` or `elif` statement is not necessarily a comparison. Basically, any Python object would fit!

```
sage: a = 5
sage: if a:
....:     print('I am not zero')
I am not zero
```

What happens under the hood is that the object `a` (here an integer) is converted to a boolean value (`True` or `False`). You can see the boolean value of an object by using `bool`

```
sage: bool(5)
True
```

```
sage: bool(0)
False
```

```
sage: bool([])
False
```

```
sage: bool([0])
True
```

A useful construction is obtained with the keyword `in`: the result of `a in b` is whether `a` belongs to the object `b`. For example:

```
sage: 2 in ZZ
True
```

```
sage: 2/3 in ZZ
False
```

```
sage: 2/3 in QQ
True
```

```
sage: 1 in [3, 5, 2, 1, 2, 8]
True
```

```
sage: 'a' in 'Saint-Flour'
True
```

```
sage: 'z' in 'Saint-Flour'
False
```

To check that an element is not in a given object use `a not in b`:

```
sage: 10 not in Primes()
True
```

```
sage: 5/2 not in ZZ
True
```

Exercise 3.8

Using an `if` statement involving `in` inside a `for` loop, count the number of vowels in the string:

```
sage: s = 'How many vowels are present in this sentence?'
```

```
sage: VowelsCount = 0
....: for i in range(0,len(s)):
....:     if s[i] in 'aeiouy':
....:         VowelsCount = VowelsCount + 1
....: VowelsCount
14
```

Count the number of consonant in the string:

```
sage: s = 'How many consonants are present in this sentence?'
```

```
sage: ConsonantsCount = 0
....: for i in range(0,len(s)):
....:     if s[i] not in 'aeiou ?':
....:         ConsonantsCount = ConsonantsCount + 1
....: ConsonantsCount
27
```

Exercise 3.9 (Pythagorean triples)

A Pythagorean triple is a triple `(a, b, c)` of positive integers so that $a^2+b^2=c^2$. An example is $3^2+4^2=5^2$. How many Pythagorean triples are there with a, b and c smaller than 100?

```
sage: PythagoreanTripleCount = 0
....: for a in range(100):
....:     for b in range(a,100):
....:         for c in range(b,100):
....:             if a^2+b^2 == c^2:
....:                 PythagoreanTripleCount += 1
....: PythagoreanTripleCount
150
```

Solve Euler problem by finding the unique Pythagorean triple so that $a+b+c=1000$

```
sage: for a in range(1000):
....:     for b in range(a,1000):
....:         for c in range(b,1000):
....:             if a^2+b^2 == c^2 and a+b+c==1000:
....:                 PythagoreanTriple = (a,b,c)
....: PythagoreanTriple
(200, 375, 425)
```

Combining conditions `or`, `and` and `not`

To make even more complicated tests you can combine them. The main operators for this are `or`, `and`.

```
sage: n = 17
sage: if n.is_prime() and (n+2).is_prime():
....:     print('a twin number!')
a twin number!
```

Exercise 3.10

Let us call a positive integer `n` a triple twin if all of `n`, `n+2` and `n+6` are primes. How many triple twins are there smaller than 10000?

```
sage: TripleTwins = 0;
....: for p in prime_range(2,10000):
....:     if (p+2).is_prime() and (p+6).is_prime():
....:         TripleTwins += 1
....: TripleTwins
55
```

The operator `not` is used for negation of a condition.

```
sage: not True
False
```

```
sage: not False
True
```

More exercises

For more exercises in the same veine you can challenge yourself with

- Euler problem 30 (sum of certain numbers)
- Euler problem 33 (digit cancelling fractions)
- Euler problem 34 (numbers which are sum of factorials of their digits)
- Euler problem 35 (circular primes)
- Euler problem 36 (integers palindromic in base 2 and 10)
- Euler problem 37 (truncatable primes)
- Euler problem 38 (integer right triangles, aka pythagorean triples)
- Euler problem 39 (binomials greater than a milion)
- Euler problem 40 (continued fractions)