Mathematics with Python and Ruby/Real numbers in Ruby

Since the apparition of the digits, numbers are in some way more concrete than were the fractions: Seeing $\frac{6}{5}$ one has the impression that there are two numbers instead of one, whereas 1.2 is a single number at first sight.

1 Decimal numbers

A number is decimal when it has a finite number of digits. Then the real numbers that are not decimal have an infinite decimal expansion. Such a number can be constructed on purpose like the Liouville numbers.

In *Ruby*, certain decimal numbers have an infinite representation yet: This apparent paradox is due to the fact that the internal representation of numbers in *Ruby* is binary and even such a simple number as 1.2 has an infinite binary expansion.

2 Fractions

A fraction is characterized by the fact that its (finite or infinite) expansion is eventually periodic. Examples:

puts(1.0/3) puts(1.0/9.0) puts(1/11.0) puts(1.0/7)

3 Irrational numbers

The first numbers known as not being fractions were the square roots and the famous Golden mean. Here are some of them:

puts(2**0.5) puts(Math.sqrt(2)) puts((1+5**0.5)/2) puts((Math.sqrt(5)+1)/2)

Two famous transcendental numbers are e and π ; they are properties of the *Math* object:

puts(Math::E) puts(Math::PI)

Champernowne's constant can be built from a string object:

c='0.' (1..40).collect { |n| c=c+n.to_s } puts(c.to_f) puts(c.to_r)

At first, *c* is a string, and *Ruby* knows it because it is written inside a pair of *quotes*. It represents already the decimal expansion of the constant, which begins with a *zero* followed by the decimal period. Then the object *(1..40)* (a list of integers) is created on the fly, and its *collect* method is called upon, with a small *Ruby* block, which has only one variable called *n* and only one instruction, asking to add the string representation of *n* to *c*. As *c* is a string, + denotes the string addition, which is concatenation. At the end, *c* is converted, either in a *float* or in a fraction.

4 Operations

The four operations are denoted by the classical +, -, * and /. As soon as one of the operand includes a period, it is recognized as a real number, and the float operation is applied. Euclidian's division is available too, which allows *Ruby* to compute the principal value of an angle (in radians):

puts(100%Math::PI)

The minus sign can also be unary, in which case it represents the negation of the following number. To add a number *h* to an other number *x*, one can, instead of writing *x=x+h*, write *x+=h* which is shorter.

To get an integer approximation to a real number, *floor*, *ceil* and *to_i* can be used. To get the absolute value of the number *x*, use *x.abs*. And its square root can be obtained with any of the following methods:

r=2**0.5 puts(r) r=Math.sqrt(2) puts(r)

5 Logarithms and hyperbolic functions

5.1 Logarithms

To compute the natural logarithm, decimal logarithm, and the arguments of the hyperbolic circular functions, one writes

puts(Math.log(0.5)) puts(Math.log10(0.5))
puts(Math.acosh(0.5)) puts(Math.asinh(0.5))

```
puts(Math.atanh(0.5))
```

5.2 Hyperbolic functions

The following script computes and displays the exponential and hyperbolic circular functions of 2:

```
puts(Math.exp(2))  puts(Math.cosh(2))
puts(Math.sinh(2)) puts(Math.tanh(2))
```

6 Circular functions

To get the cosine, sine and tangent of an angle which measures one radian:

```
puts(Math.cos(1)) puts(Math.sin(1)) puts(Math.tan(1))
```

To get the angle from one of the above numbers, one just has to add an extra *a* before the name of the function:

```
puts(Math.acos(0.5))  puts(Math.asin(0.5))
puts(Math.atan(0.5))
```

To know an angle from the sides of a right triangle, on can use *Math.atan(y/x)* or *Math.atan2(x,y)*. And to know the third side, one can use *Math.hypot(x,y)*. For example, if one knows that the sides of a right triangle measure 5' and 12', one can know the angles and hypotenuse with this script:

```
cdr=180/Math::PI a=12 b=5 puts(Math.atan2(a,b)*cdr)
puts(Math.atan2(b,a)*cdr) puts(Math.hypot(a,b))
```

7 Text and image sources, contributors, and licenses

7.1 Text

- **Mathematics with Python and Ruby/Real numbers in Ruby** *Source:* https://en.wikibooks.org/wiki/Mathematics_with_Python_and_Ruby/Real_numbers_in_Ruby?oldid=2289953 *Contributors:* Recent Runes and Alain Busser

7.2 Images

7.3 Content license