ruby的变量有局部变量，全局变量，实例变量，类变量，常量。

1、局部变量

局部变量以一个小写字母开头或下划线开头

局部变量有局部作用域限制（比如一个block内），它的作用域起始于声明处，结束于该声明所在的块、方法定义、类/模块定义的结尾。大家在写代码时经常这样写：

irb(main):001:0> i=123　---------- 这里的i就是局部变量

=>123

irb(main):002:0> s="hi"　---------- 这里的s就是局部变量

=>"hi"ruby的变量是动态变量，某个变量在前一刻是数字型，在后一刻可以是字符型：

irb(main):003:0> x=321

=>321

irb(main):004:0> x="hello"

=>"hello"

ruby是动态变量，但却是强类型，例如字符和数字不能直接相加：

irb(main):005:0> x=10

=>10

irb(main):006:0> y="hi"

=>"hi"

irb(main):007:0> x+y

TypeError: String can't be coerced into Fixnum

from（irb):7:in '+'

from（irb):7from : 0

必须手工进行转换： irb(main):008:0> x.to_s + y

=> "10hi"

2、全局变量

ruby的全局变量以$开头，例如：$x $y。全局变量可以在程序的任何地方加以引用。全局变量无需变量声明。引用尚未初始化的全局变量时，其值为 nil

ruby有内置的全局变量，应该是从perl哪里抄来的，例如 $！记录最近一次产生的错误，$. 表示行号等。良好的编程实际，是不使用全局变量，他们危险而难以跟踪。

3、实例变量

ruby的实例变量以@开头，是指实例化后的对象，才绑定的变量。实例变量属于特定的对象。例如：

irb(main):016:0> class Myclass

irb(main):017:0> def initialize(name,gender,age)

irb(main):018:0> @name=name

irb(main):019:0> @gender=gender

irb(main):020:2> @age=age

irb(main):021:0> end

irb(main):022:0> end

=> nil ----------

@name， @gender， @age都是实例变量。可以在类或子类的方法中引用实例变量。若引用尚未被初始化的实例变量的话，其值为

nil。

irb(main):023:0> x=Myclass.new("john")

=> #<Myclass:0x7f2e15a7dad8 @name="john">

Myclass类，他的构造器接收一个name参数，然后把该参数赋值给实例变量@name。

x是Myclass的实例，她拥有实例变量@name。

只有在类被实例化时，实例变量才产生和存在。但是，实例对象并不能直接访问实例变量：

irb(main):022:0> x.@name

SyntaxError: compile error

(irb):22: syntax error, unexpected tIVAR

from (irb):22

from :0

这样是错误的。必须在；类里面，定义get方法，来访问实例变量：

irb(main):023:0> class Myclass

irb(main):024:1> def name

irb(main):025:2> @name

irb(main):026:2> end

irb(main):027:1> end

```
=> nil

irb(main):028:0> x.name

=> "john"
```

当然，也可以定义set方法，来设置实例变量：

```
irb(main):029:0> class Myclass

irb(main):030:1>   def name=(value)

irb(main):031:2>     @name=value

irb(main):032:2>   end

irb(main):033:1> end

=> nil

irb(main):034:0> x.name="jean"

=> "jean"

irb(main):035:0> x.name

=> "jean"
```

这个set和get，可以通过ruby的元编程来实现，例如：

```
irb(main):036:0> class Myclass

irb(main):037:1>   attr_accessor :age

irb(main):038:1> end

=> nil

irb(main):039:0> x.age=20
```

=> 20

irb(main):040:0> x.age

=> 20

只要设置attr_accessor就够了，他会对@age这个实例变量，创建set和get方法。

irb(main):041:0> x=> #<Myclass:0x7f2e15a7dad8 @name="jean", @age=20>

对应的还有attr_reader只设置get方法，attr_writer只设置set方法。

4、类变量

ruby的类变量以@@开头，例如在类里申明的@@x @@y 等，一般很少使用。

类变量在类的定义中定义，可以在类的特殊方法、实例方法等处对类变量进行赋值和引用。类变量被类，类的子类和他们的实例对象共享。

class Person

@@number = 0 #使用前必须有初值

def initialize(name, gender, age)

@age = age

@@number += 1

end

end

类变量是私有的，在类外无法直接访问，你只能通过实例方法和类方法去访问它。可以把类变量看作一种被类、子类以及它们的实例所共享的全局变量。

模块中定义的类变量(模块变量)被所有包含该模块的类所共享。

```
module TestModule

@@foo = 10

end

class Klass

include Foo

p @@foo += 1          # => 11

end

class Base

include Foo

p @@foo += 2          # => 12

end
```

5、常量

ruby的常量以大写字母开头，常数的定义和初始化由赋值过程完成。

```
irb(main):048:0>Pi=3.14          ----------Pi就是一个常量

=>3.14
```

然而，ruby的常量是可以改变的。若对已定义的常数进行赋值的话，会出现警告信息：

```
irb(main):049:0> Pi=3.15

(irb):49: warning: already initialized constant Pi

=> 3.15
```

irb(main):050:0> Pi

=> 3.15

尽管触发警告，但常量的确被改变了。

注意：1）若引用未定义的常数会引发 NameError异常。

2）常量可以定义在类和模块中，不能定义在方法中

通常在类里设置常量：

irb(main):051:0> class Myclass

irb(main):052:1> Pi=3.1415

irb(main):053:1> end

=> 3.1415

3）若想在外部访问类或模块中的常数时，要使用“：：”操作符.

从类的外部访问这个常量：irb(main):055:0>Myclass::Pi

=>3.1415

模块也一样，例如访问ruby内置的Math模块的PI常量： irb(main):056:0>Math::PI

=>3.14159265358979

4)在类定义表达式生成类对象的同时，还会将类对象赋值给一个与该类同名的常数，引用类名也就是引用该常数。

class Test

end

p Test.class #Class

p Test #test

若想访问Object类中的常数(顶层的常数)时，也需要也使用"::"操作符，但操作符左边为空。