gsrubylib

require '**debuglog**' unless $gs\_nodebuglog

require '**pry**' unless $gs\_nopry

require '**contracts**'

include Contracts

|  |  |
| --- | --- |
| if object.**in?** collection  if object.**not\_in?** collection |  |
| if object.**not\_nil?** |  |
| str = object.**pp\_s** |  |
| o.**define\_method**(:add) do |x,y| x + y end |  |
| squares = (1..10).**build\_hash** { |n| [n, n\*n] } | graph |
| squares.values.**mapf**(:to\_s) | collectf |
| h = squares.**apply\_keys** { |k| k.to\_s } |  |
| h = squares.**apply\_values** { |k| k.to\_s } |  |
| “foo”.**indent**(4) |  |
| “bar”.**tabto**(4) |  |
| USAGE = %{  | usage: prog [-o dir] -h file...  | where  | -o dir outputs to DIR  | -h prints this message  }.**trim**("|") |  |
| **StringIO.string** { |o| o.puts “Hi…” } |  |
| class Person  **attr\_predicate** :young  **attr\_predicate\_rw** :successful  end |  |

|  |  |
| --- | --- |
| Labels  Result = GS::Label.create(:win, :lose,  :draw)  result = Result.lose  result.to\_s / to\_sym / symbol / inspect  result == Result[:lose]  Labels are safer than symbols because they guard against misspellings. They also “inspect” nicely. | Values  Person =  GS::Value.new(name: String, age: Nat, married: Bool)  .default(married: false)  .create  p = Person[name: 'John', age: 25] *or Person.new(…)*  *or Person[‘John’, 25]*  p.name; p.age; p.married; p.married?  p[:name], p[:age] *etc.*  p.with(age: 26, married: true)  p.attributes  p.values  p.values(:name, :married)  e = p.upgrade(Employee, title: 'Nurse', salary: 58400)  p = e.downgrade(Person)  Person.info # "Person[name: String, …]"  Values are read-only structs with Contracts built-in, default values, predicate methods, copy-constructors (*with*), transformers (*upgrade*, *downgrade*).  They combine type safety, state safety and convenience. |
|  |  |