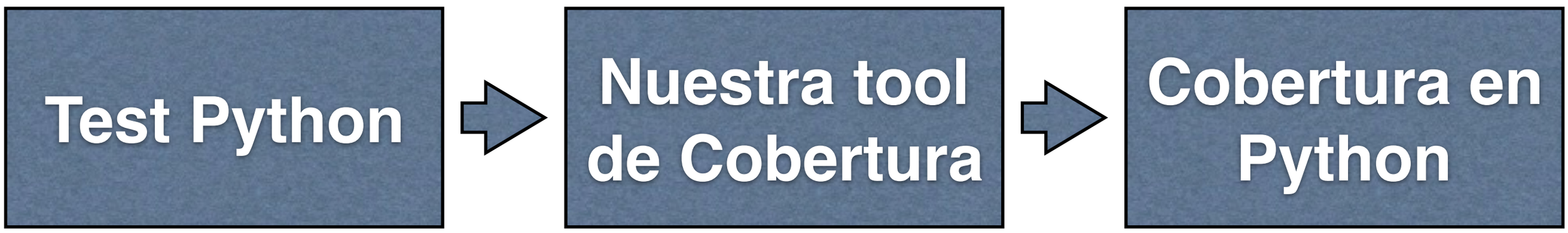
TP2: EvoPy

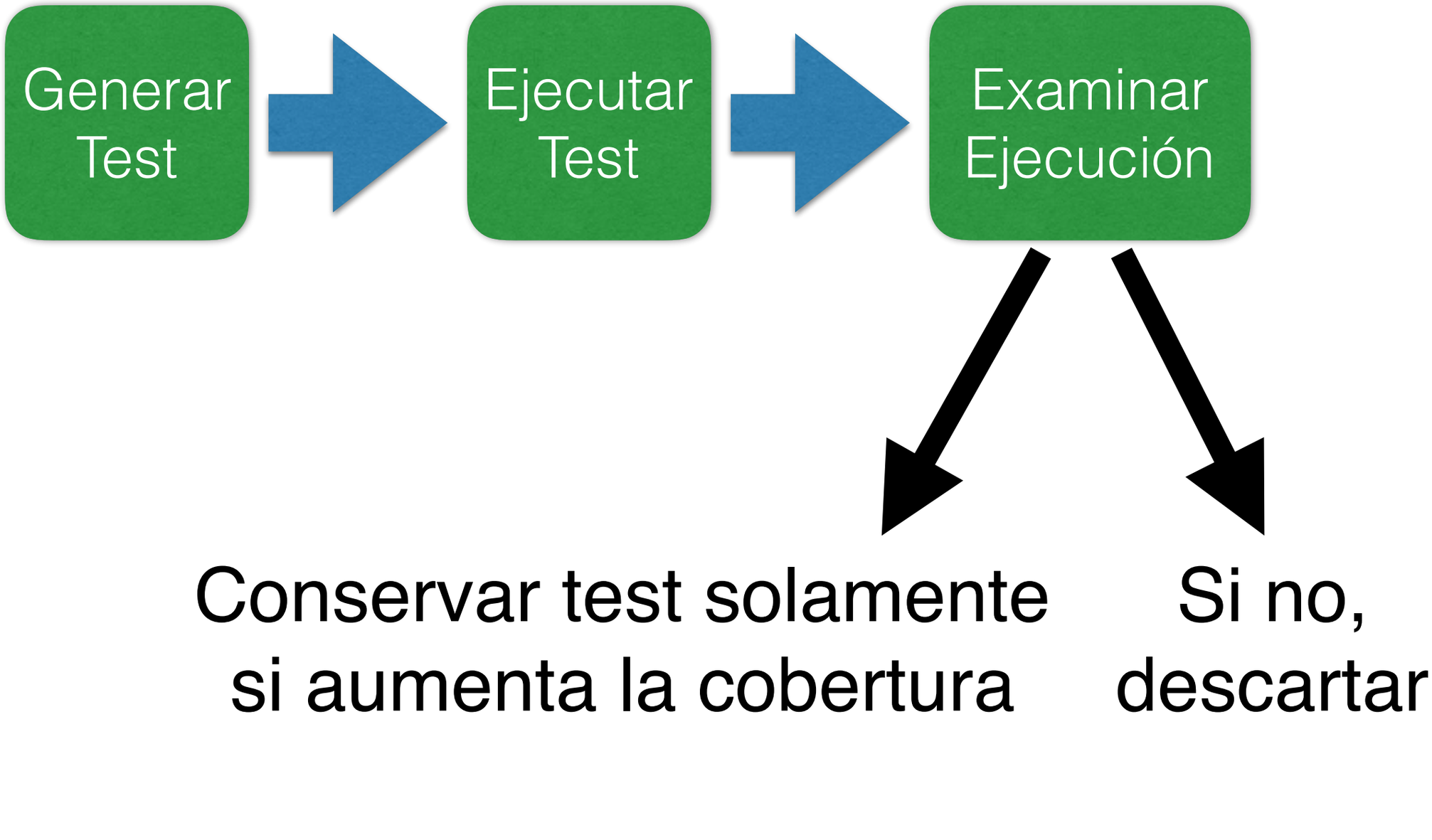
Generación Automática de Casos de Test - 2016

Infraestructura: Taller #1

- Tenemos la posibilidad de Medir la Cobertura de un Test Python



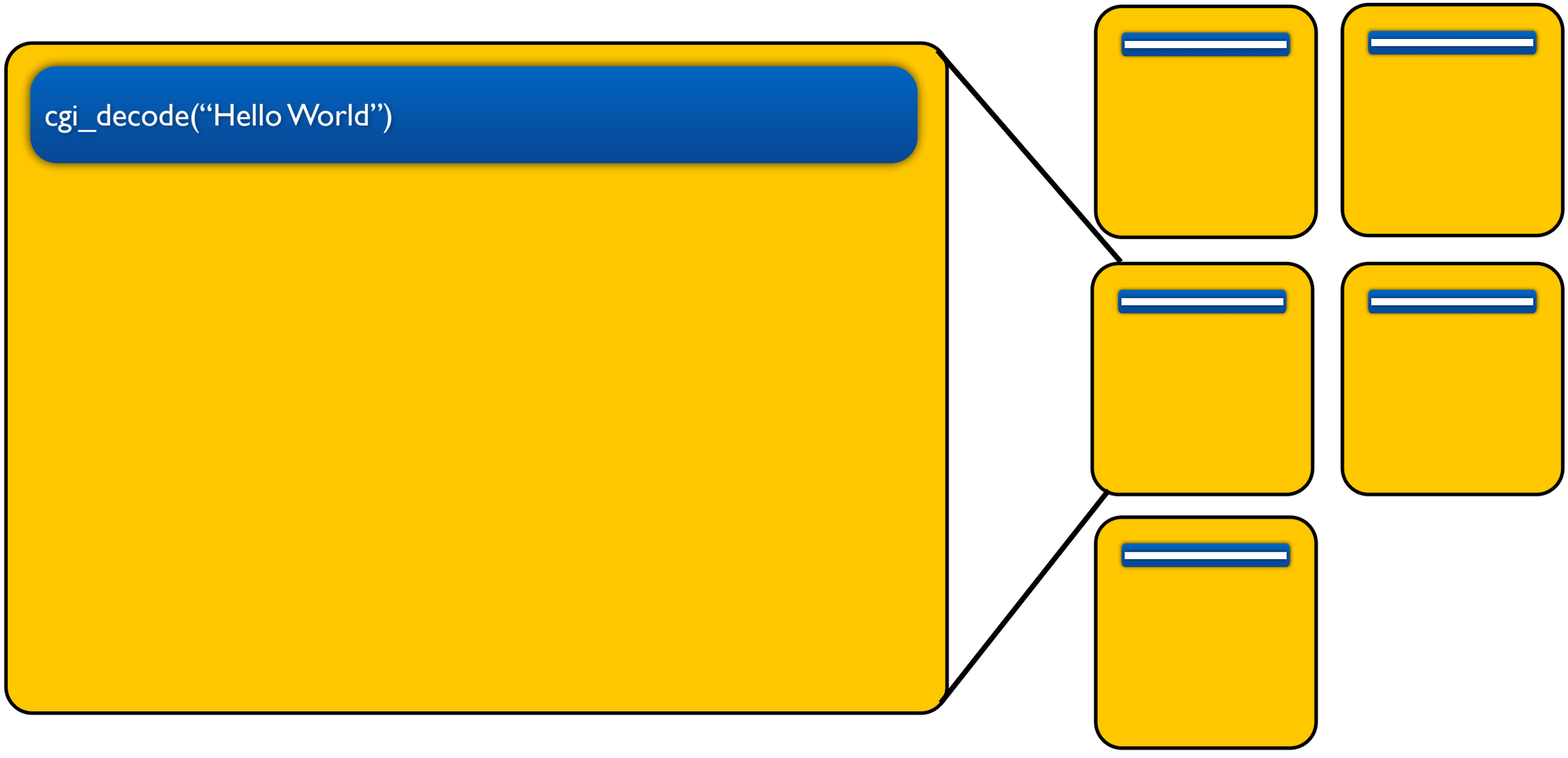
TP1: Random Testing



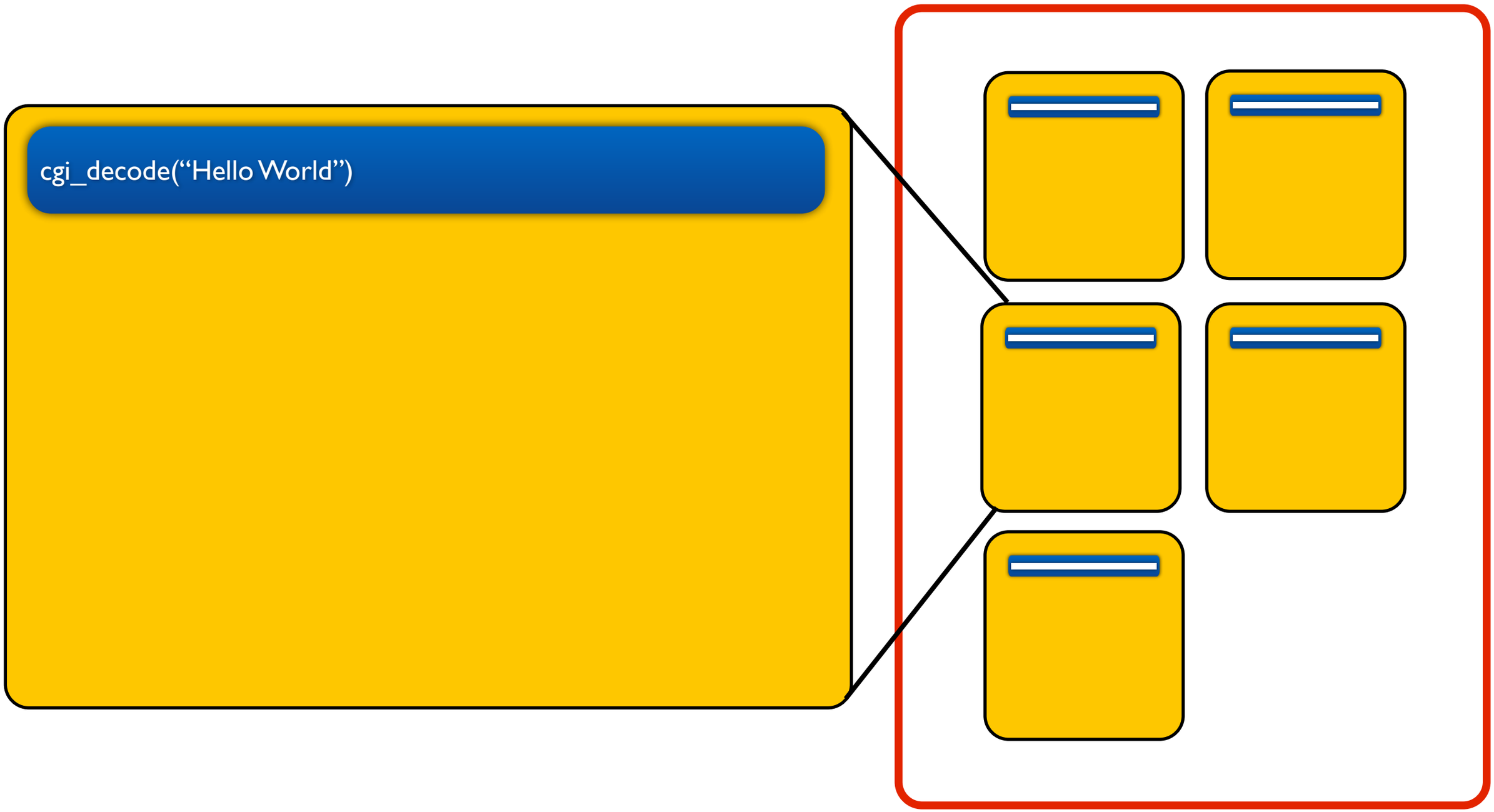
TP2: Enunciado

- Desarrollar un whole-test suite search-based generator para funciones Python que reciban sólo tipos primitivos:
  - Ejemplo: cgi_decode(“Hello World”)
- Utilizar pyntch para inferir automáticamente el tipo de cada argumento de las funciones
  - Ignorar funciones “privadas” (empiezan con “_”)
- Completar la clase EvoPy

EvoPy: Whole-Test Suite Generation

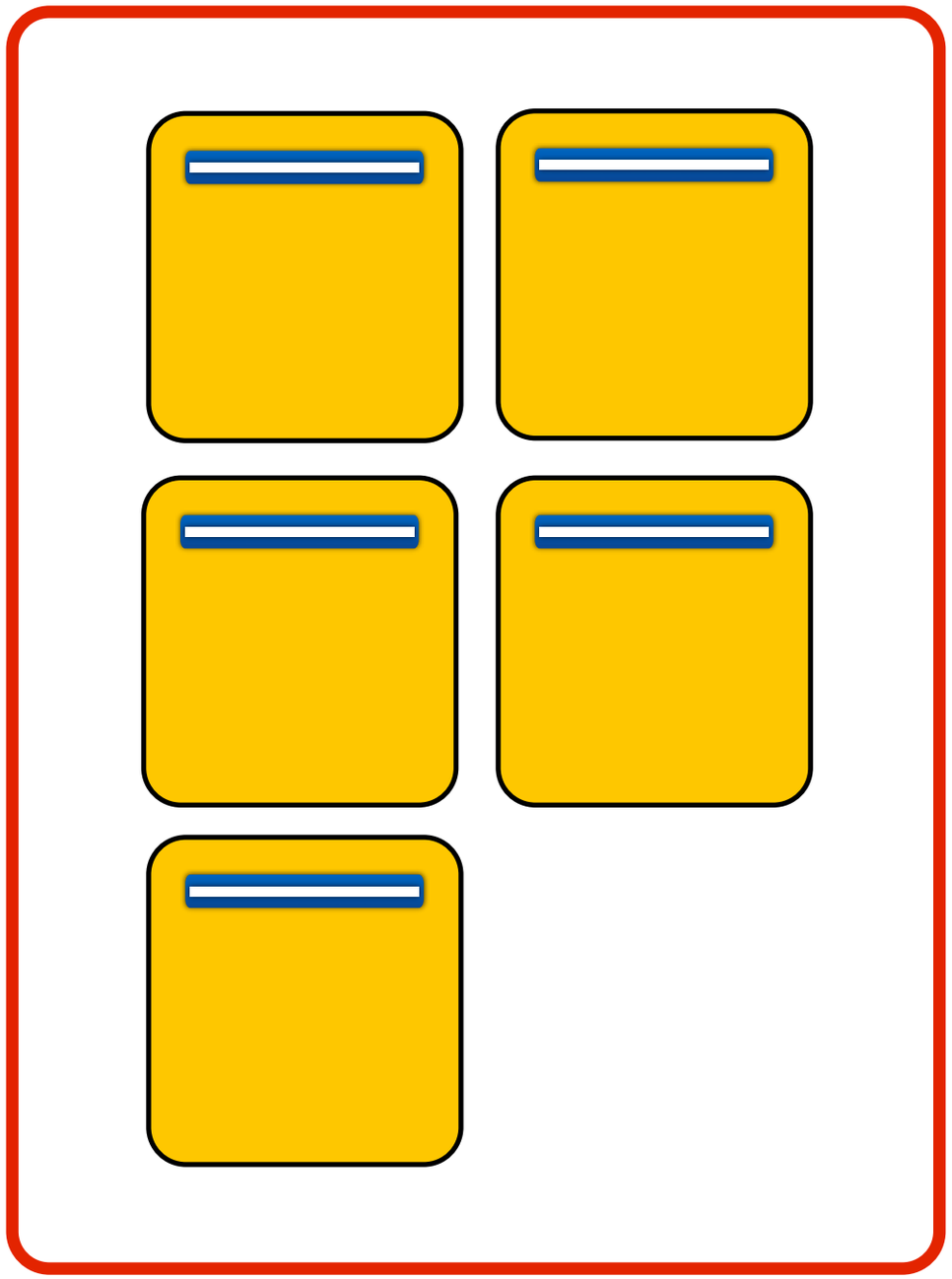


EvoPy: Whole-Test Suite Generation



EvoPy: Whole-Test Suite Generation

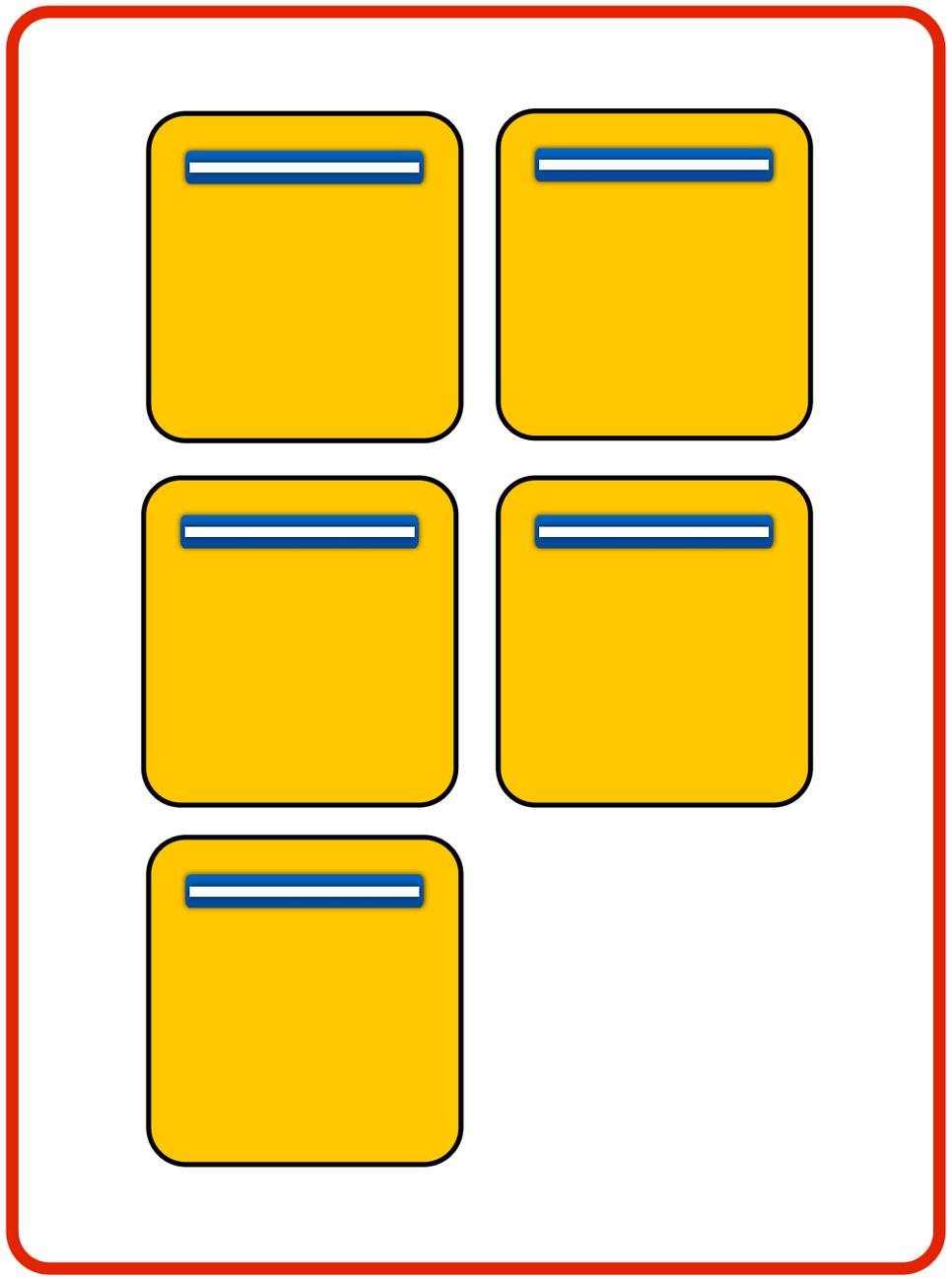
- Optimize entire test suite at once towards
- Ordering of coverage goals no longer an issue
- Infeasibility of individual coverage goals does not affect search.



EvoPy: Fitness Function

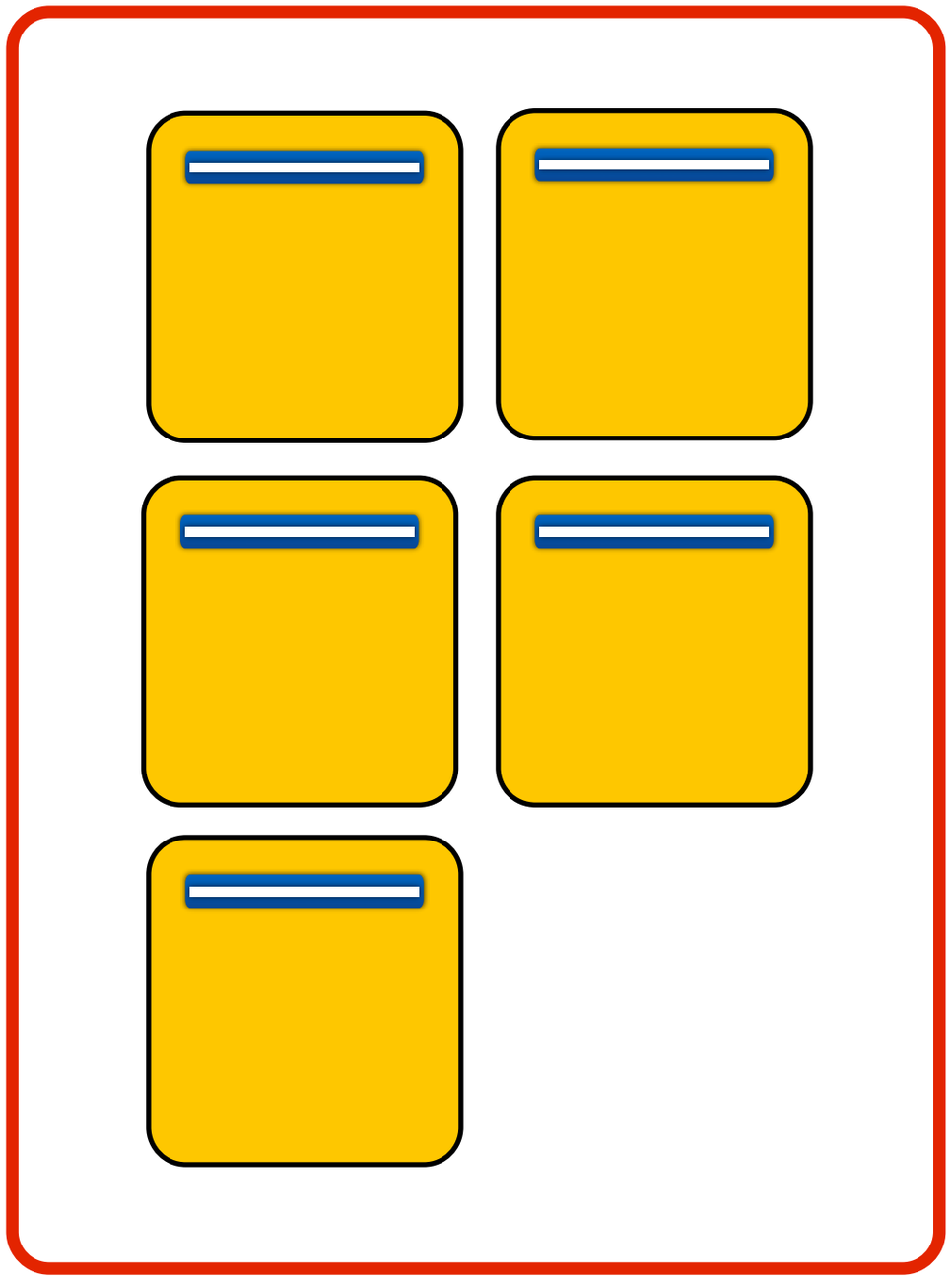
- Cantidad de líneas y brances cubiertos por un individuo
- Usar la misma tool de cobertura desarrollada para el TP1 :)

Genetic Algorithm (GA)



Genetic Algorithm (GA)

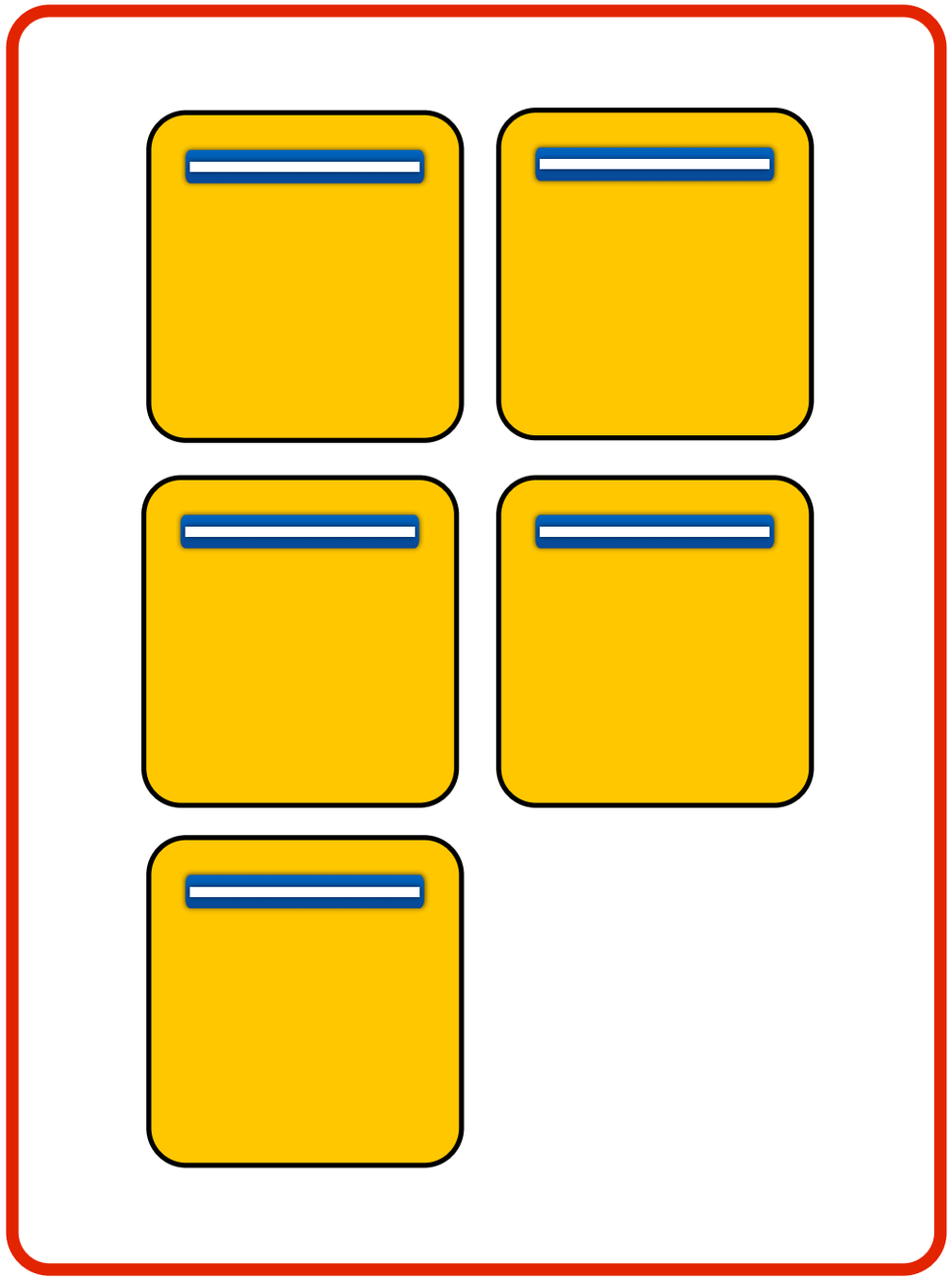
Initialize Population



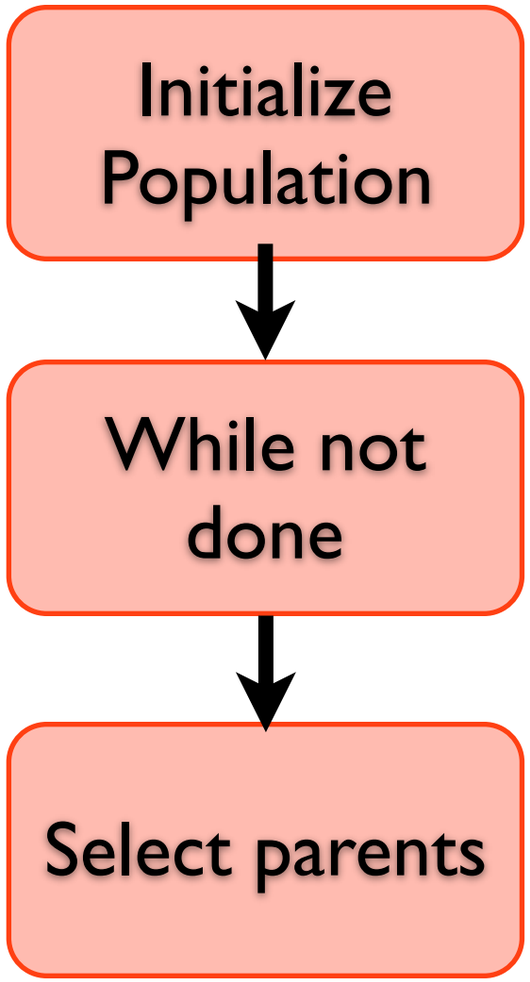
Genetic Algorithm (GA)

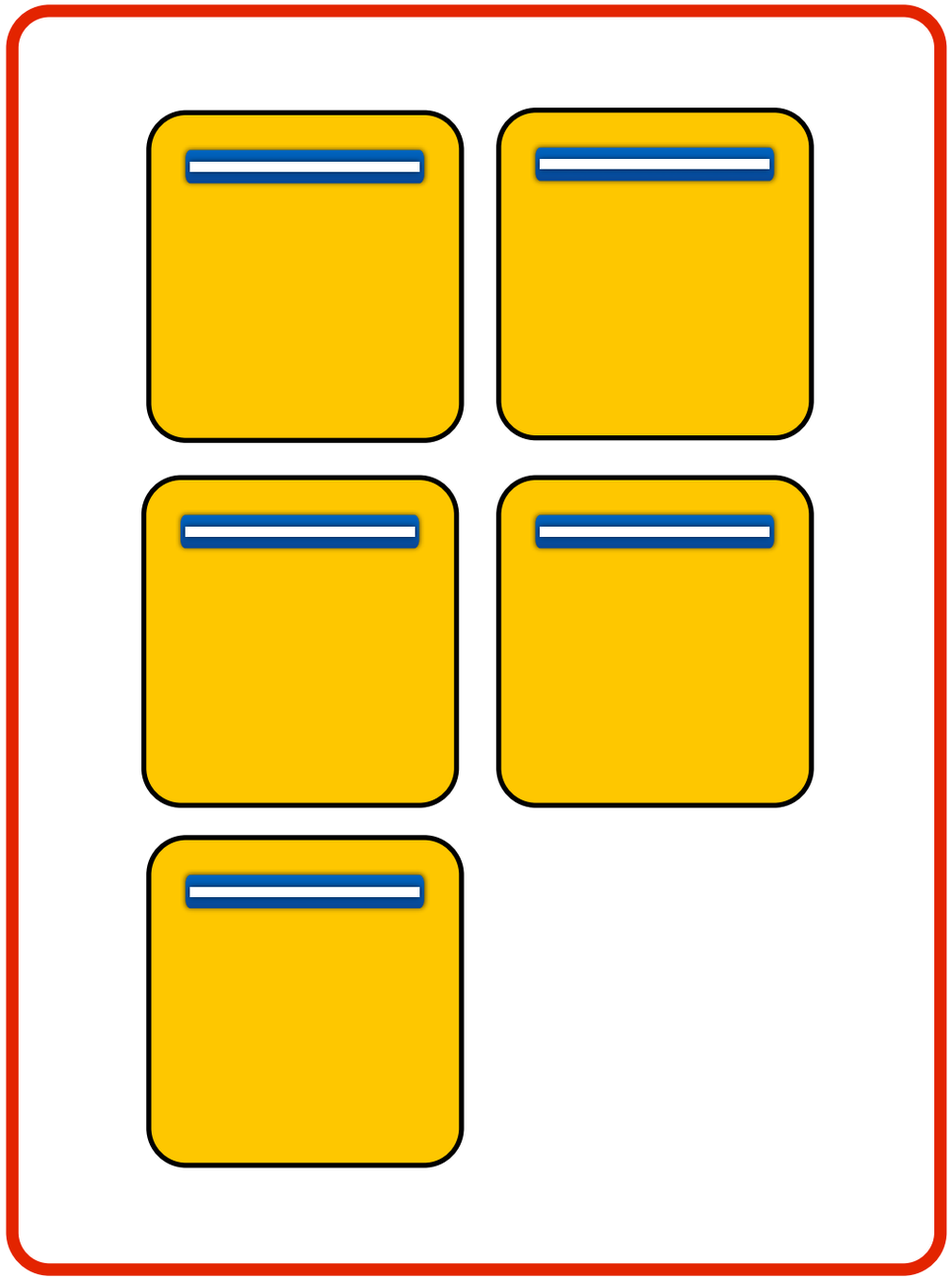
Initialize Population

While not done

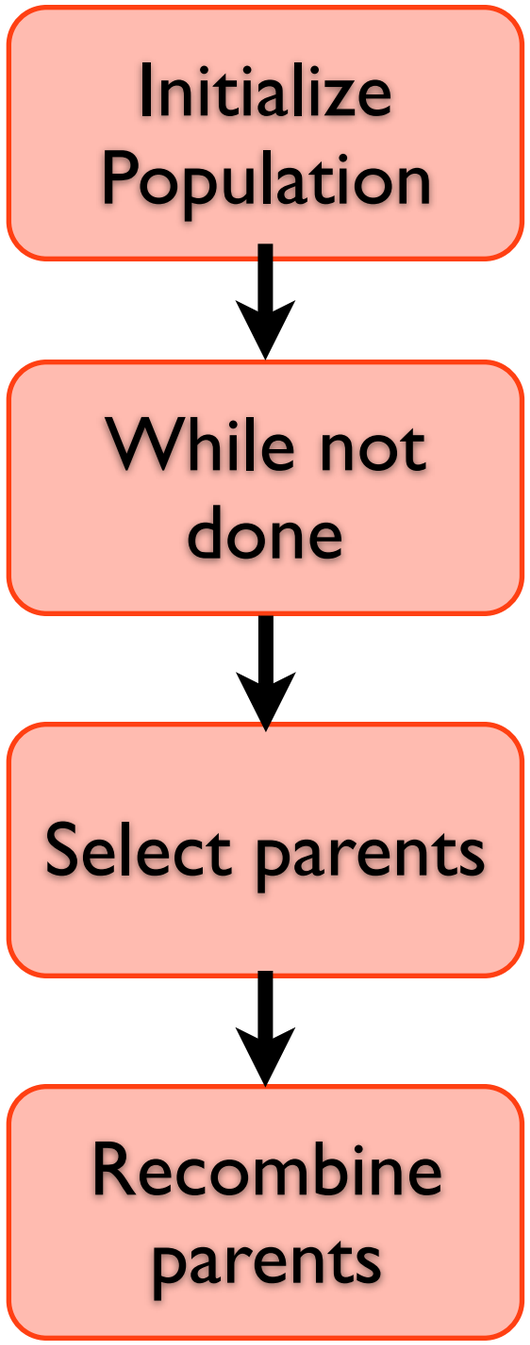


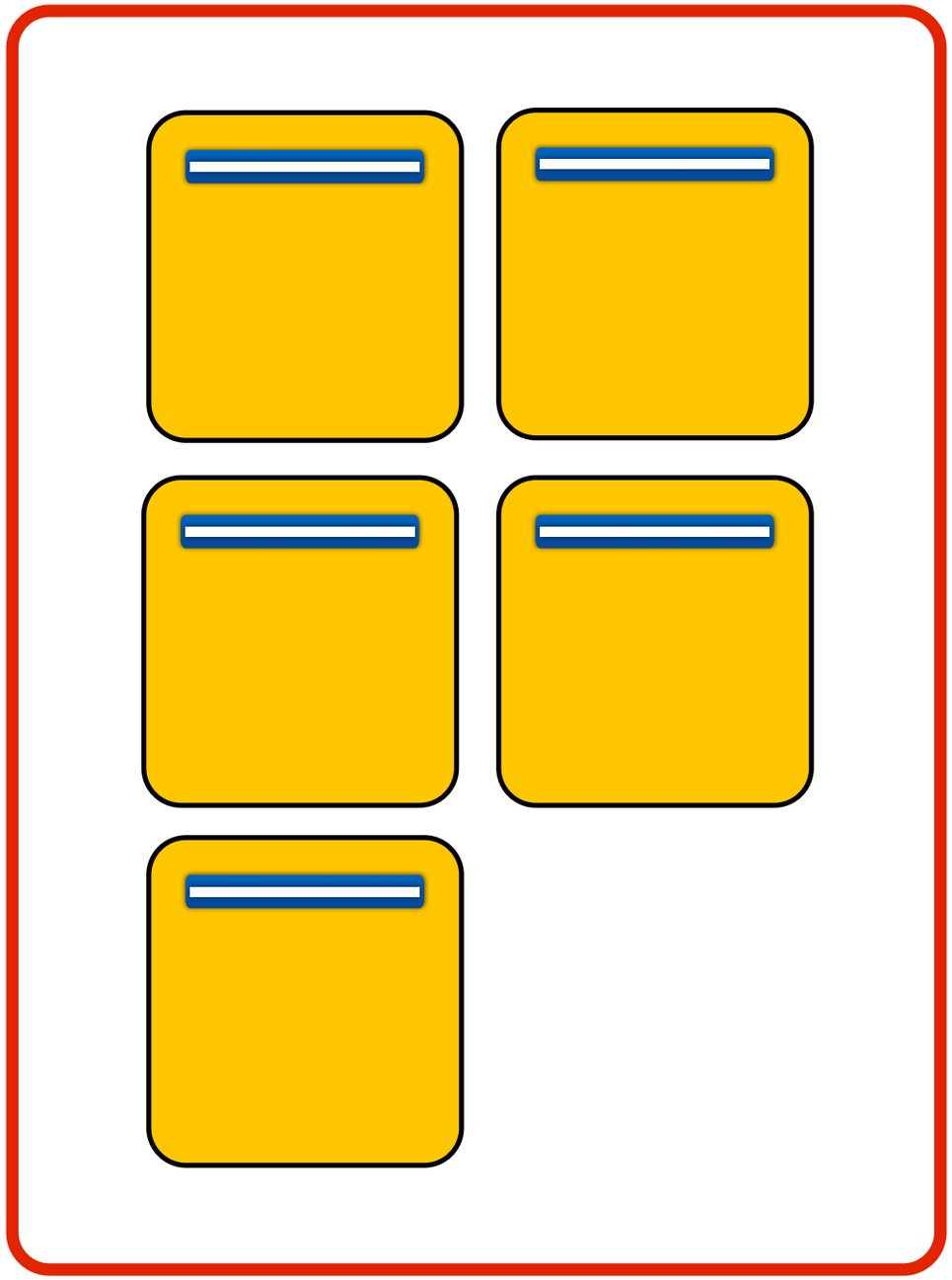
Genetic Algorithm (GA)



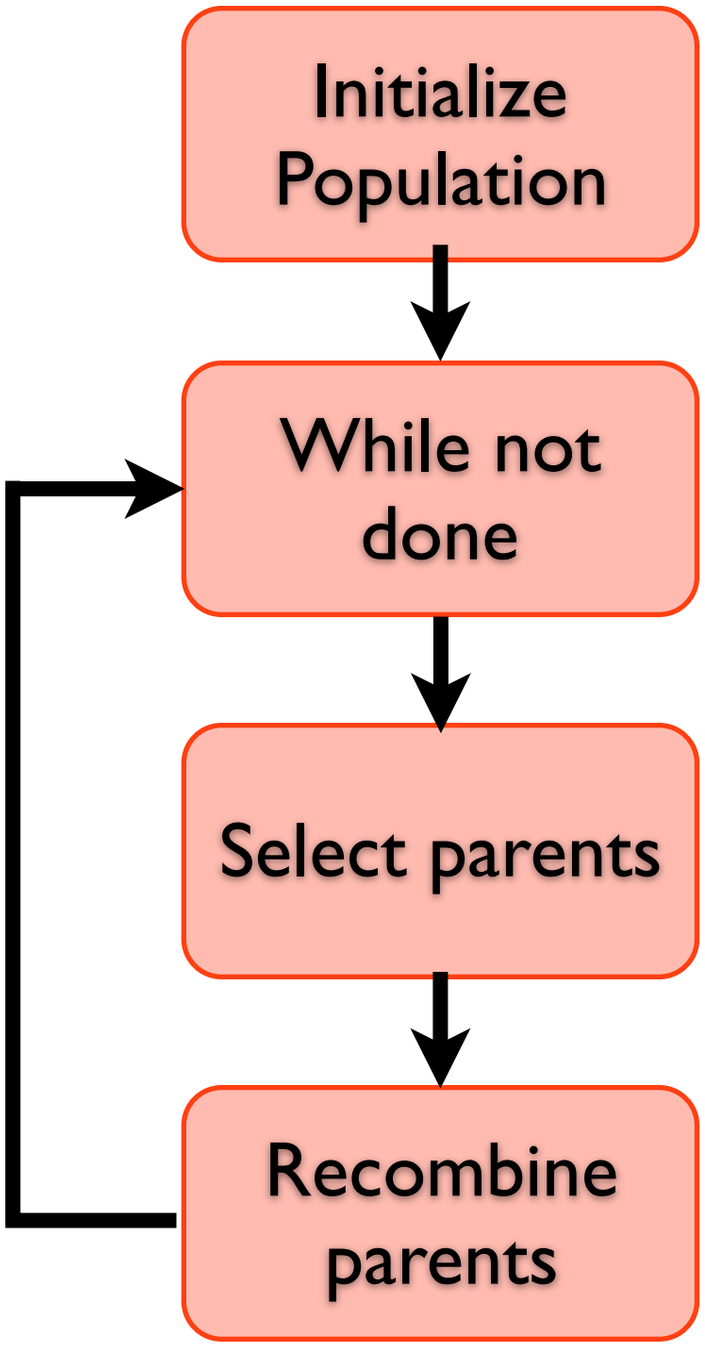


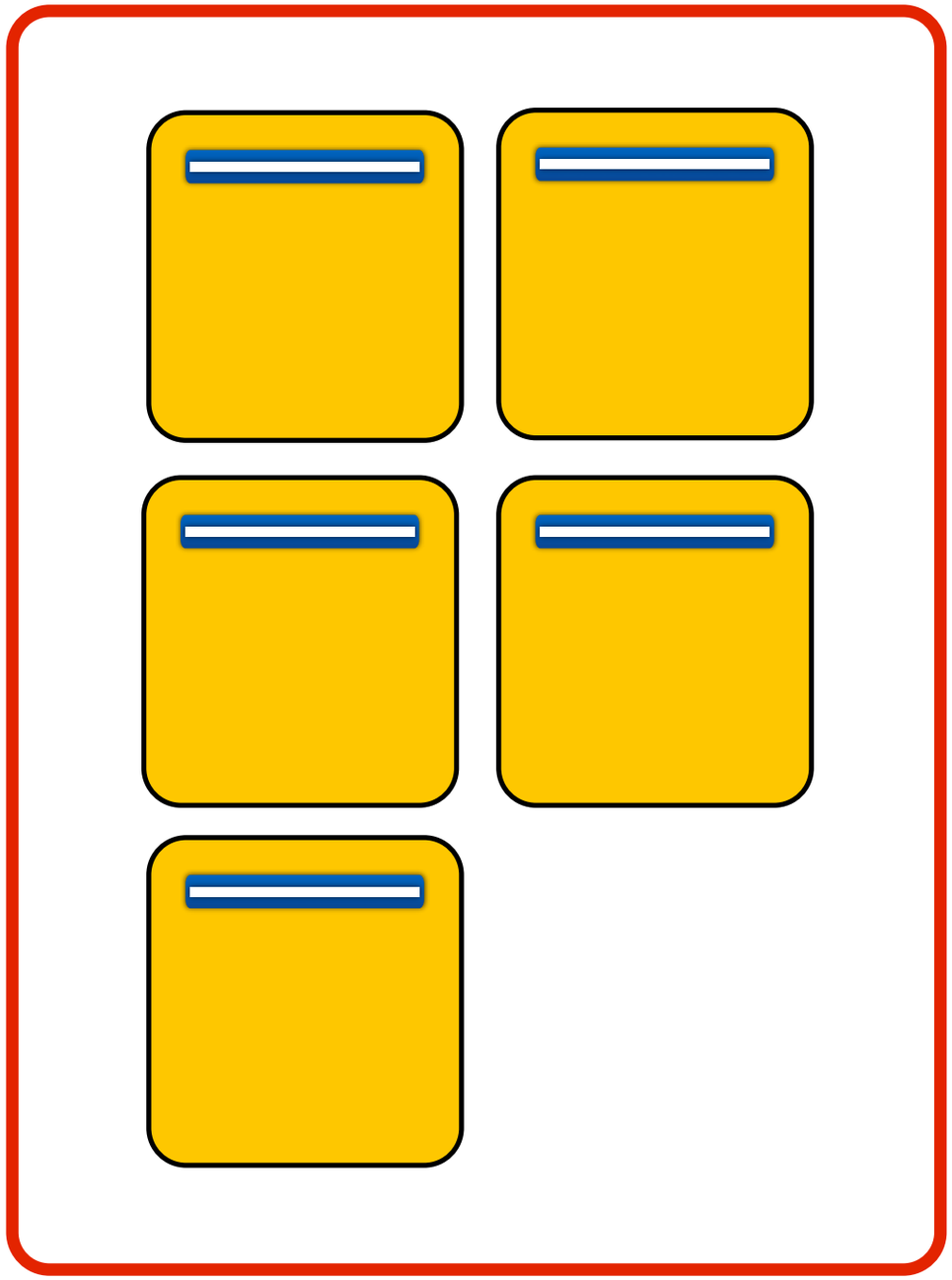
Genetic Algorithm (GA)



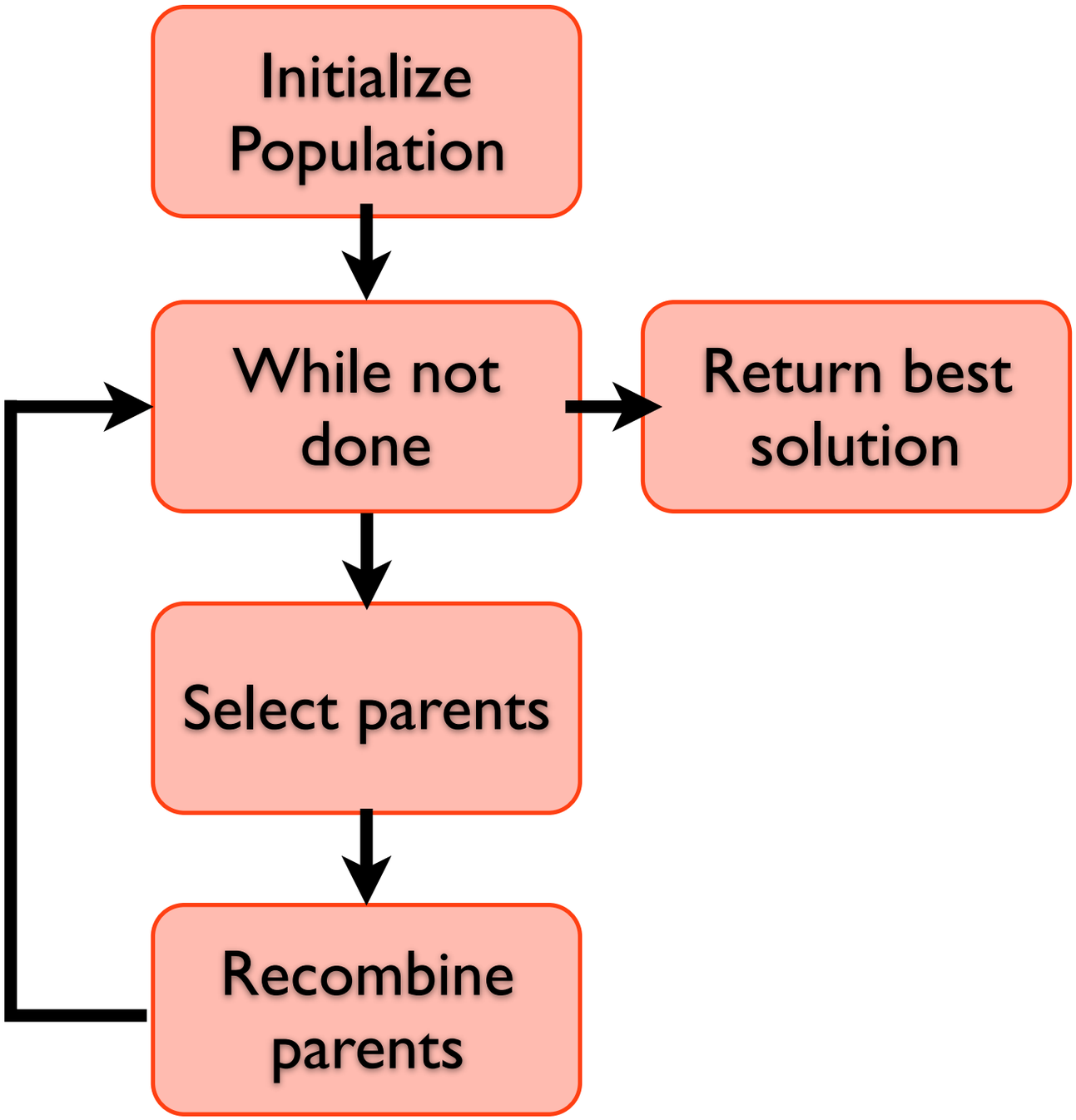


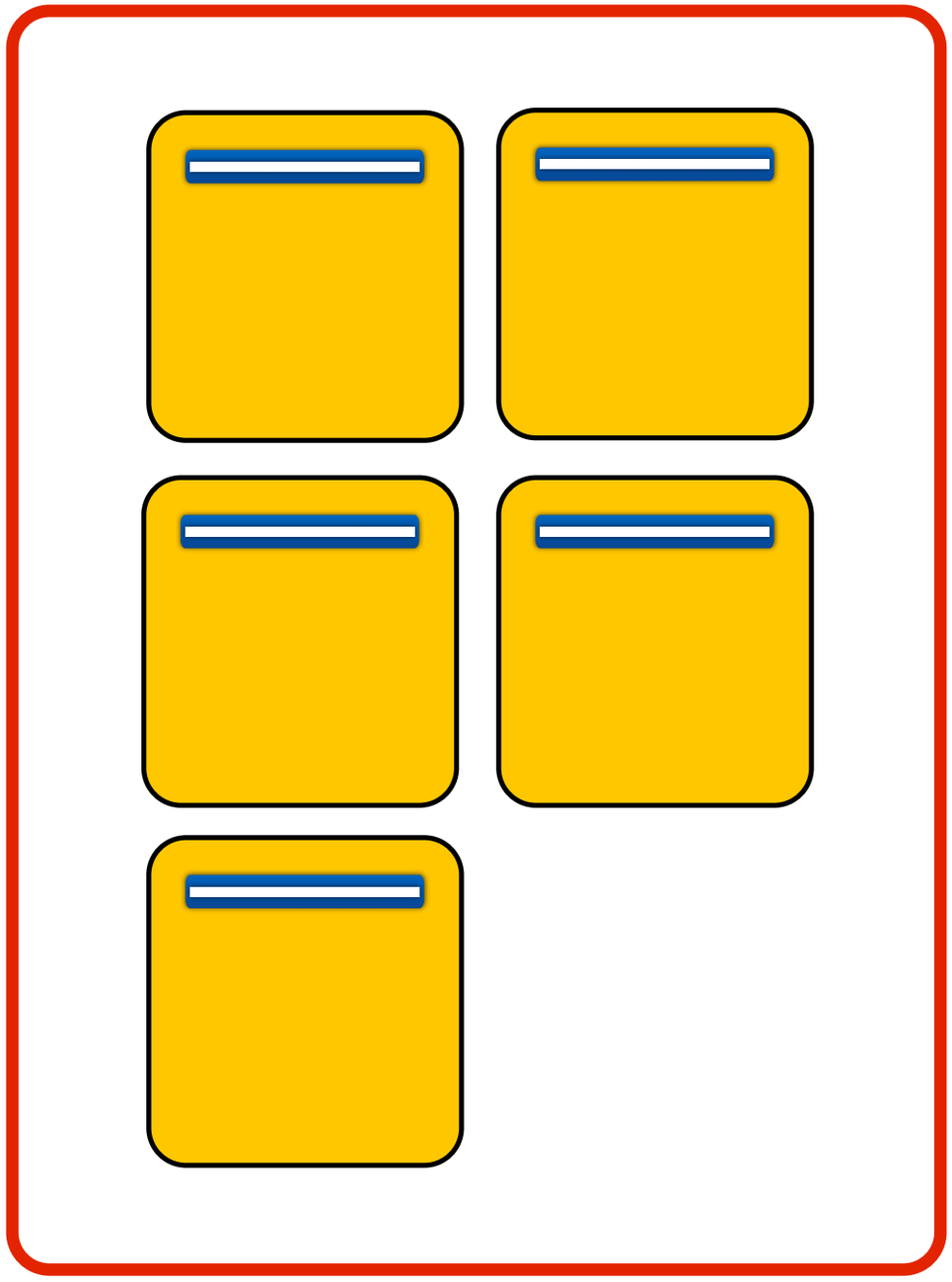
Genetic Algorithm (GA)



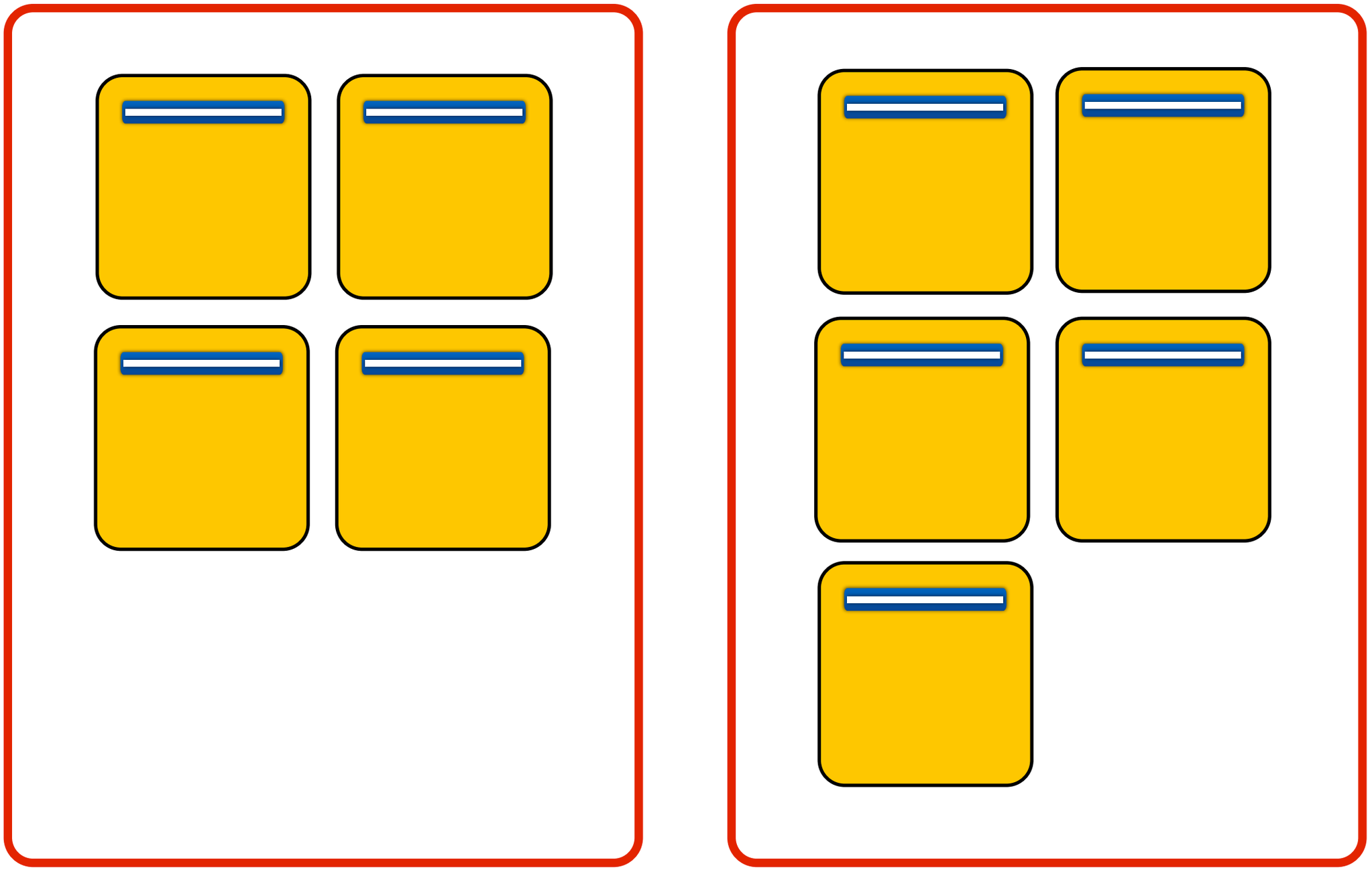


Genetic Algorithm (GA)

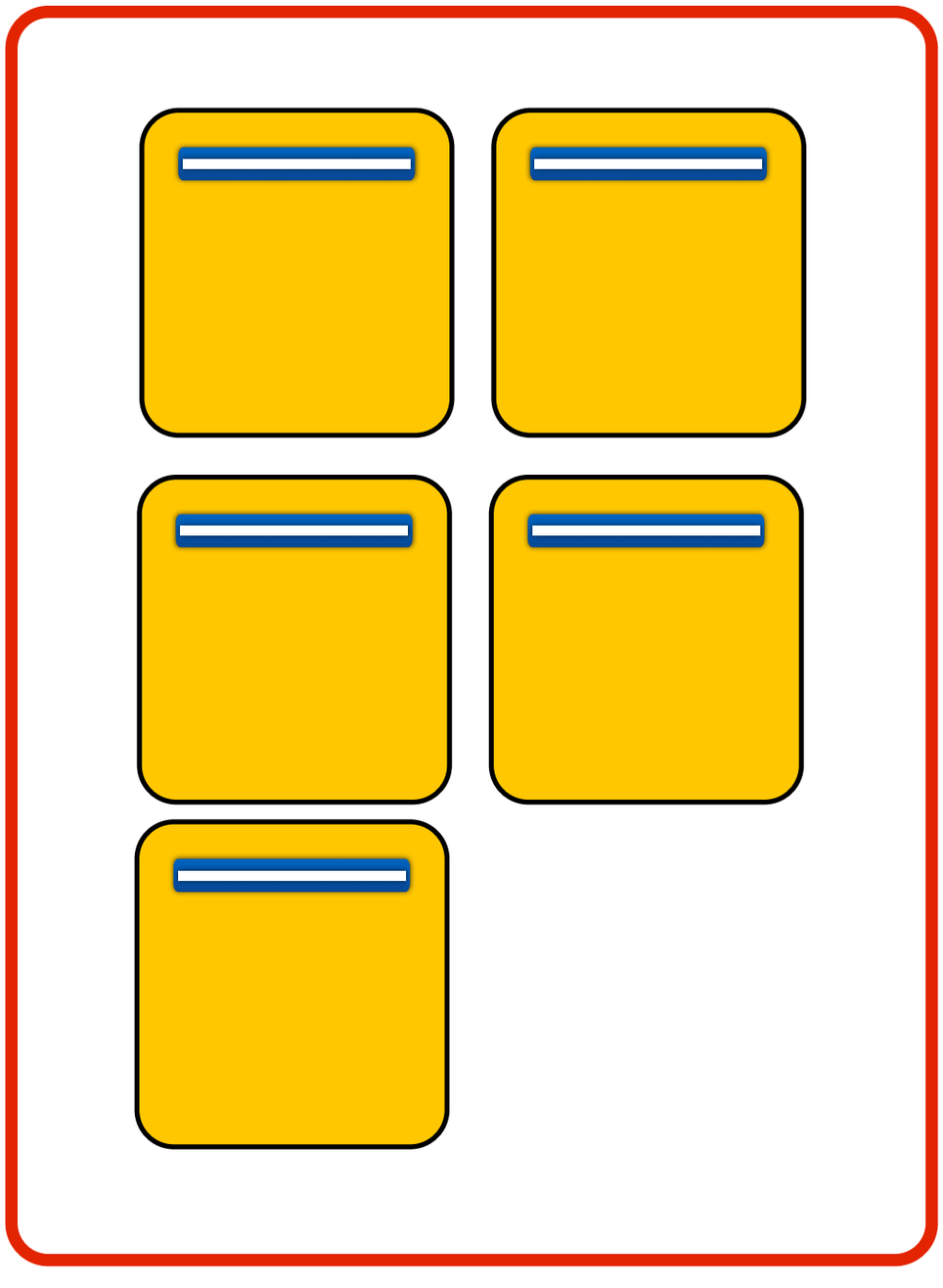


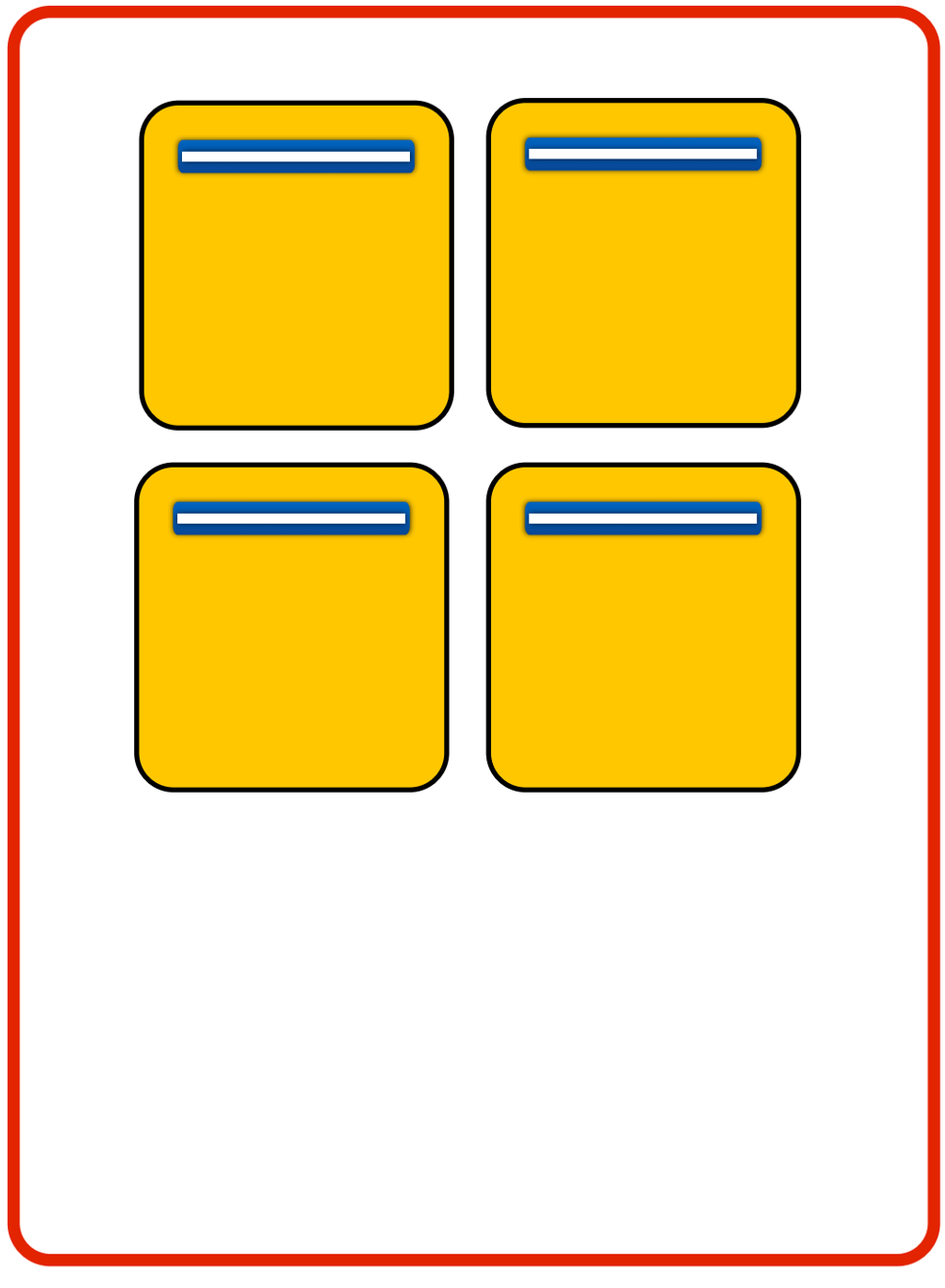


Whole-Test Suite Generation

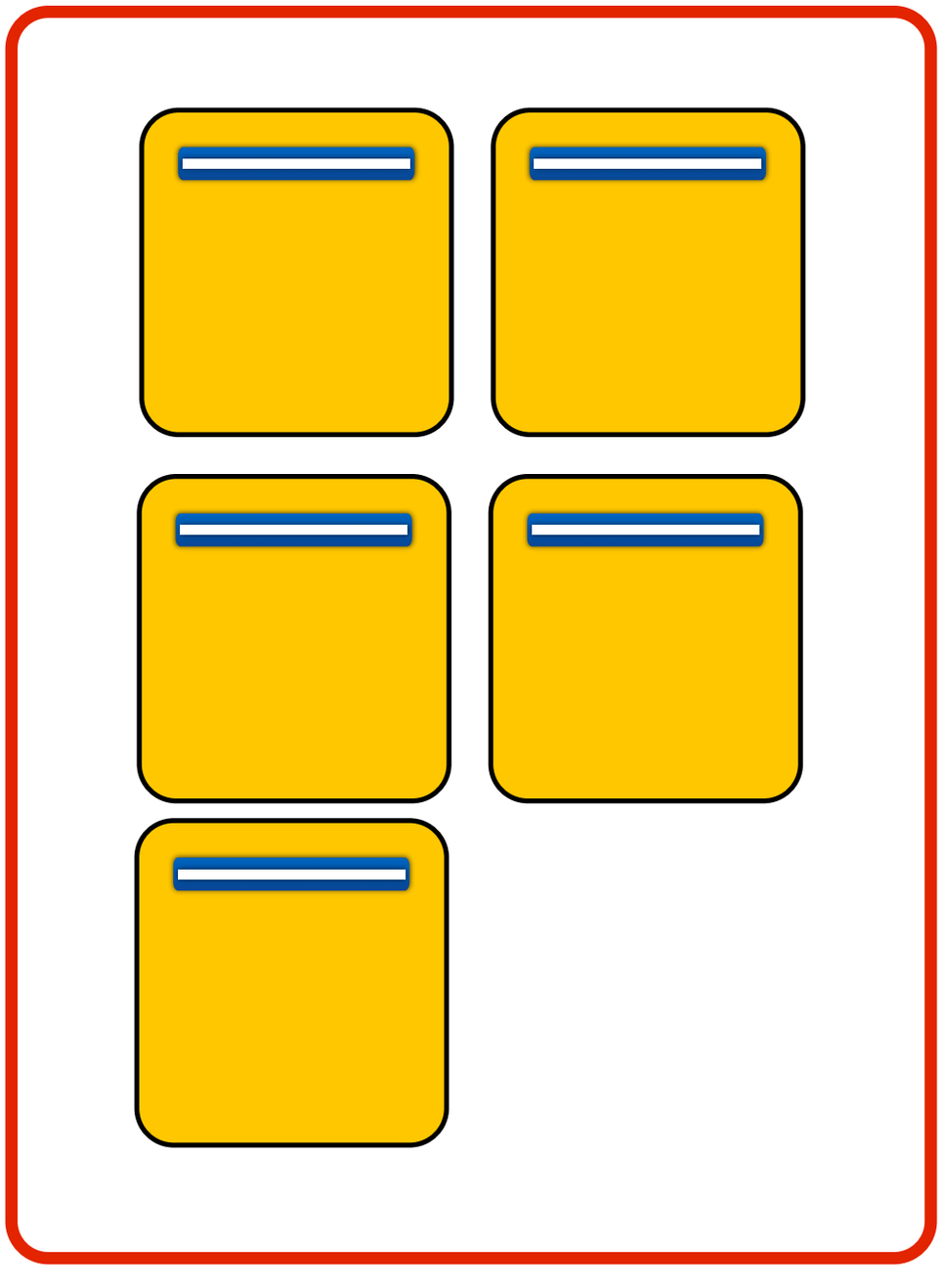


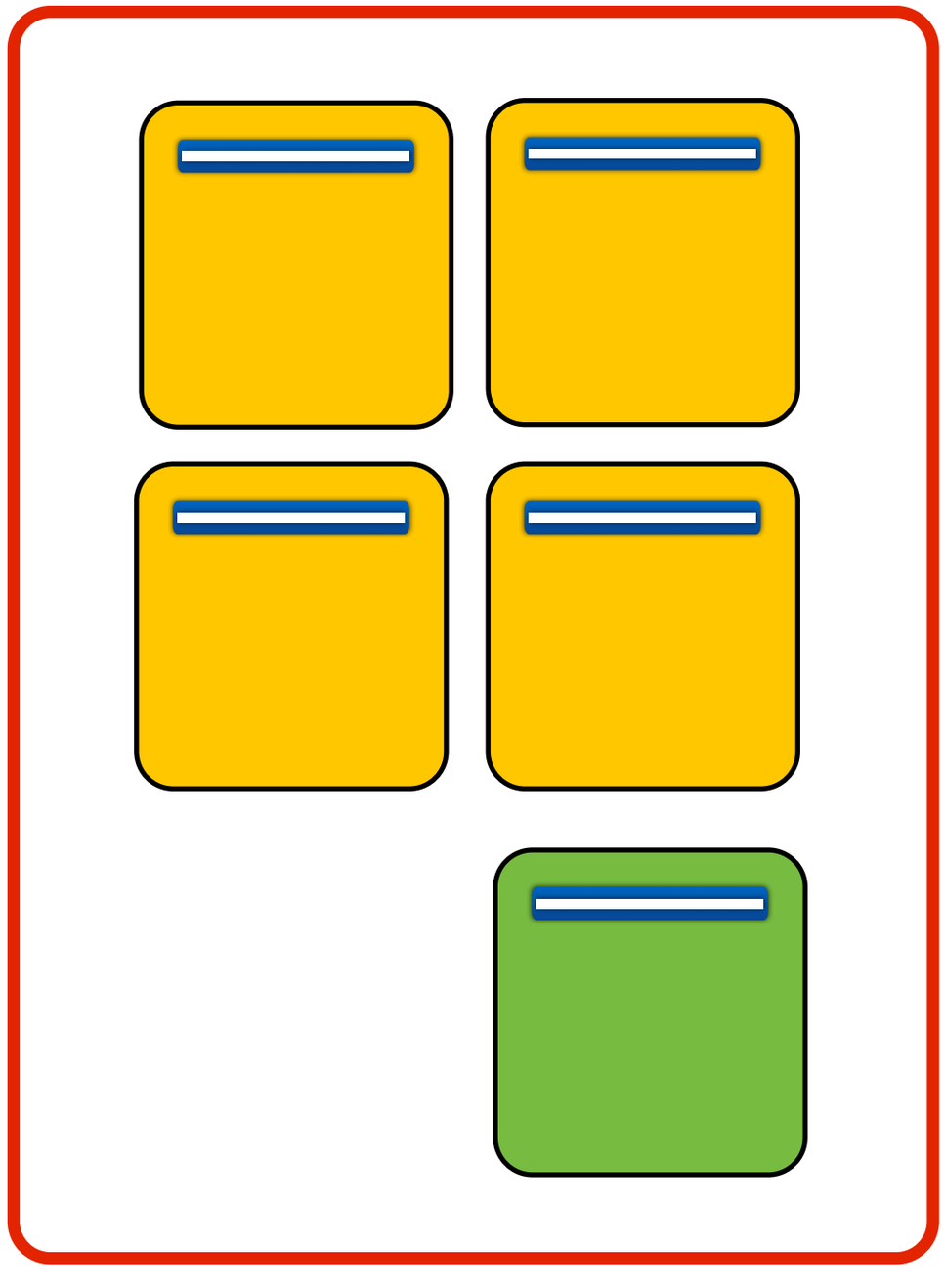
Crossover



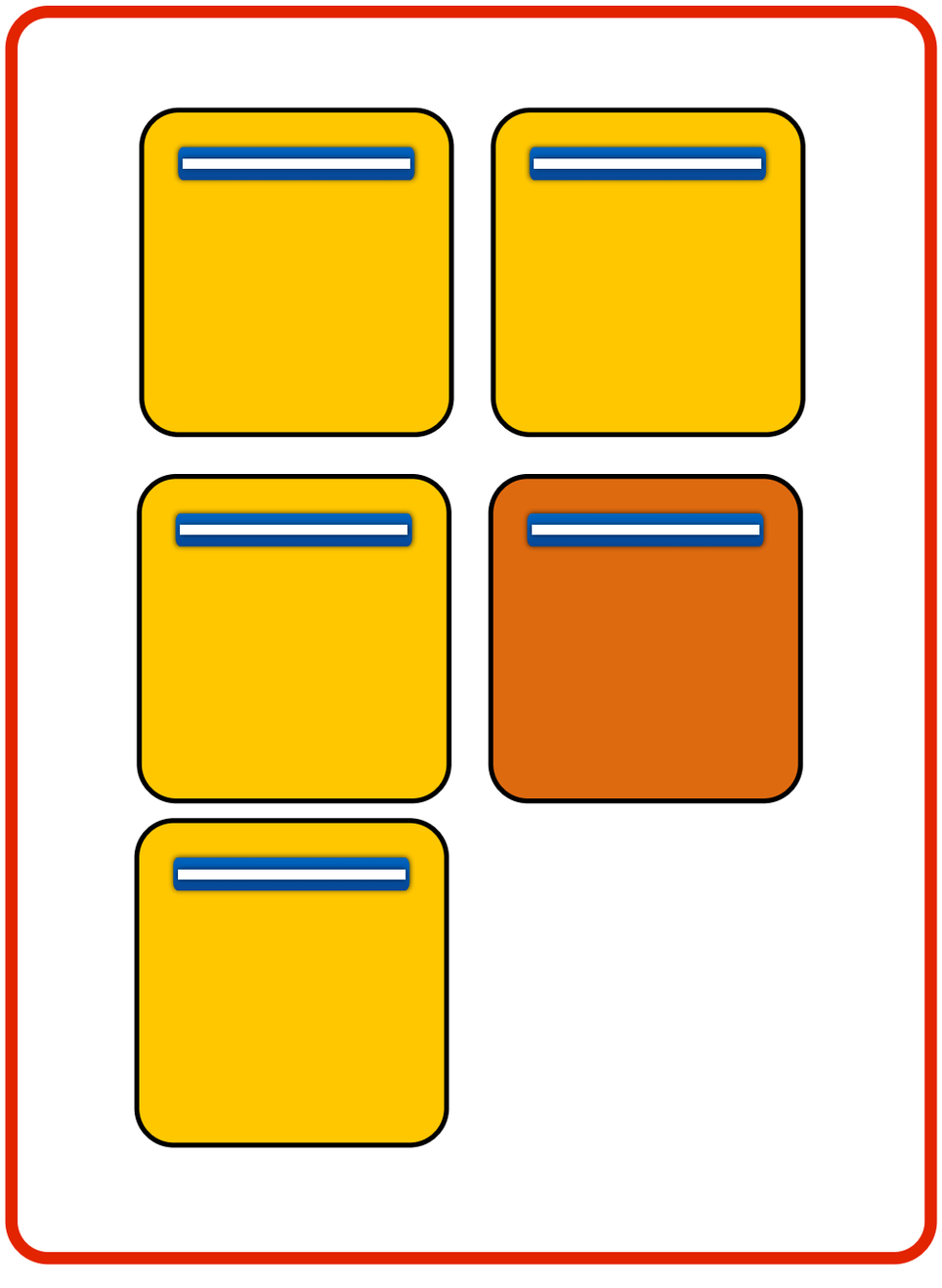


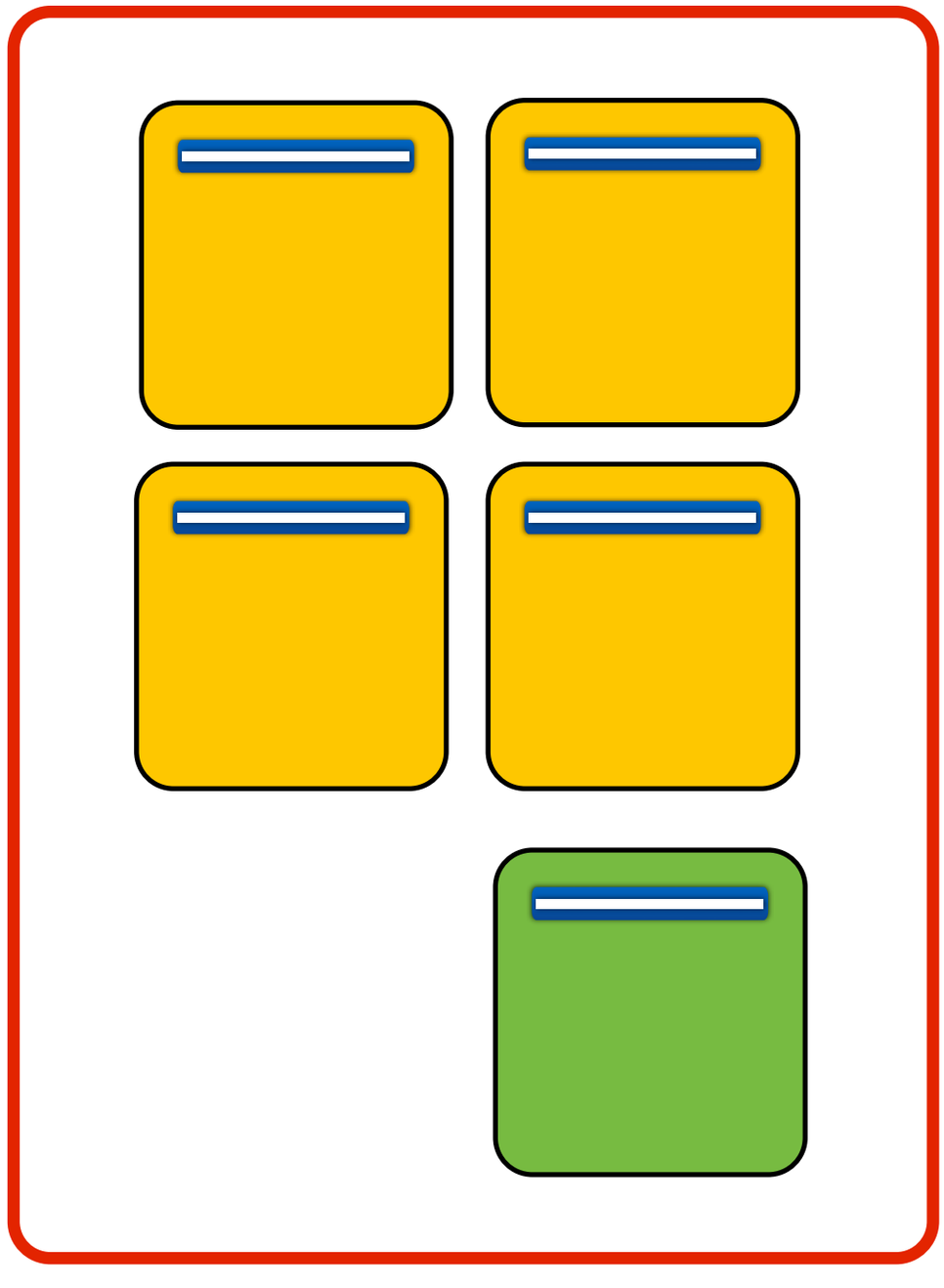
Mutation - Add Test



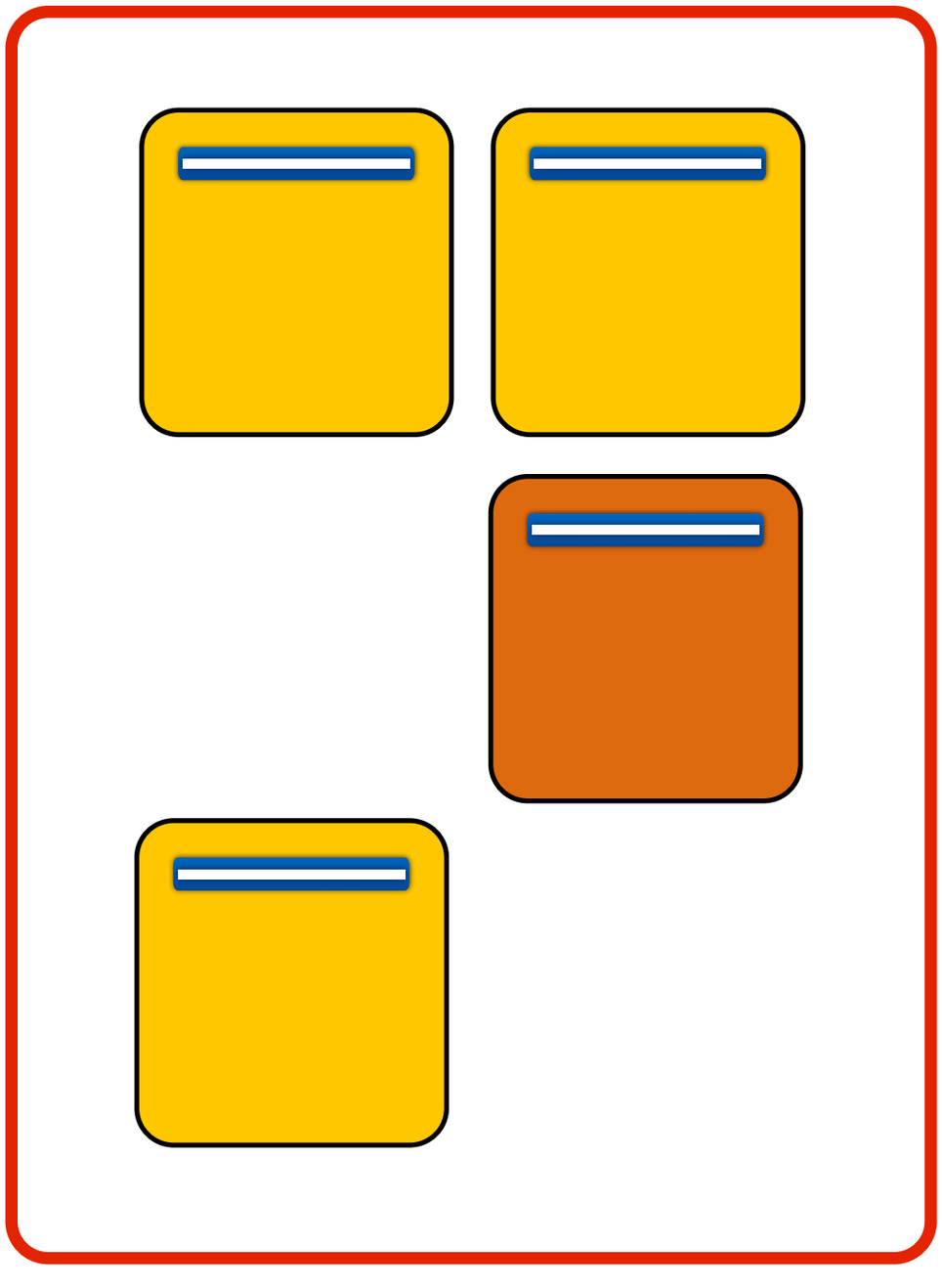


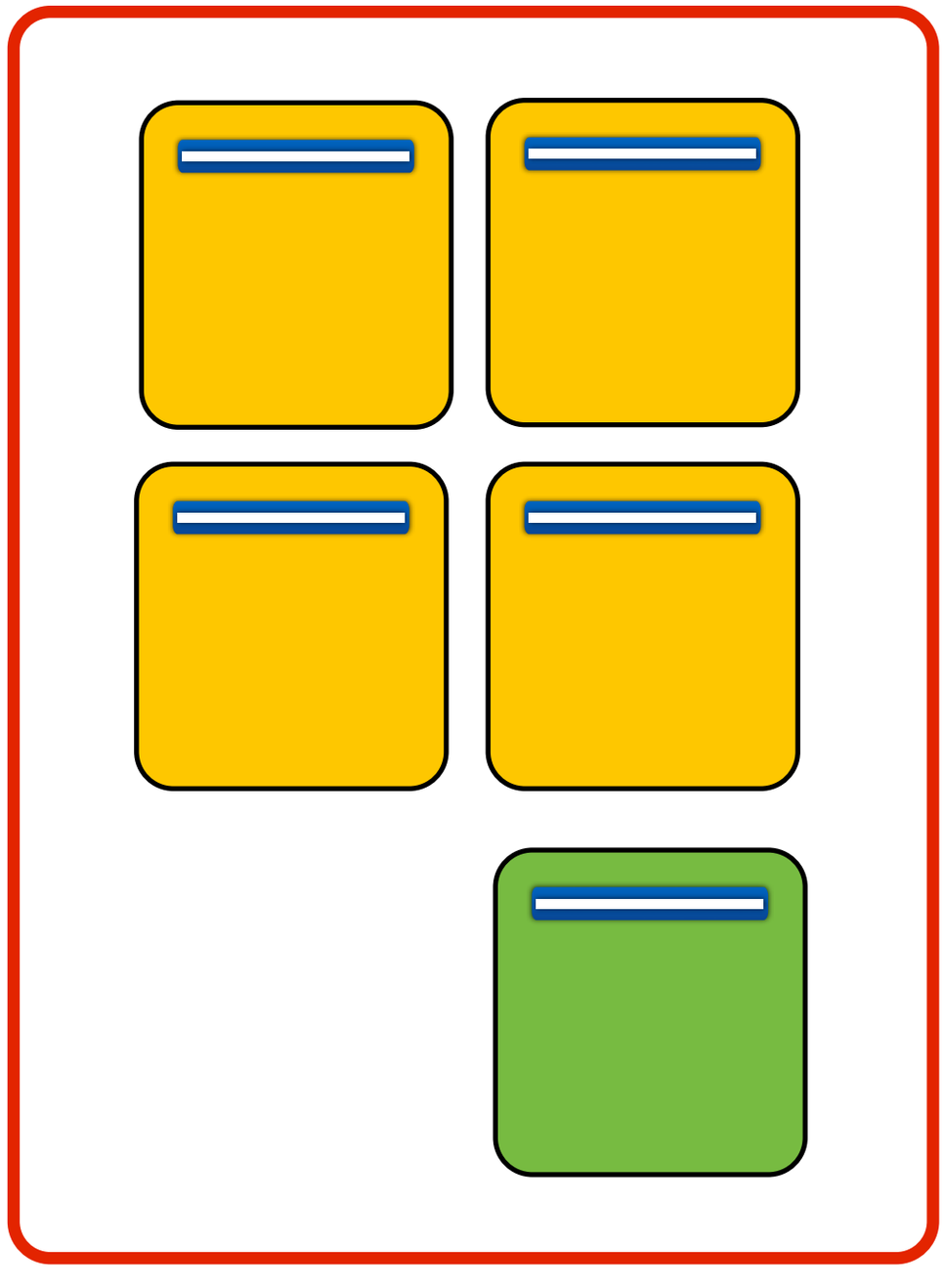
Mutation - Modify Test





Mutation - Remove Test





EvoPy - Argumentos

- **targetPyFile**: el archivo .py con el módulo target (str)
- **maxGenerations**: la cantidad máxima de generaciones del algoritmo (int). El valor -1 significa que se ignora.
- **maxTime**: la cantidad de segundos máximo a utilizar en la generación (int). El valor -1 significa que se ignora.
- **output_dir**: la carpeta donde se almacenarán los tests generados (str)
- **random_seed**: la semilla de aleatoriedad para tener reproducibilidad (int)

EvoPy - Parámetros

- **populationSize:** cantidad de individuos en la población (default 50)
- **eliteSize**: tamaño de la elite (default 1)
- **testSuiteLength**: tamaño máximo de un test suite (default 40)

EvoPy - Parámetros

- **crossoverProbability**: [0,1] probabilidad de aplicar crossover al seleccionar individuos (default 0.75)
- **mutationProbability**: [0,1] probabilidad de aplicar mutation en el nuevo individuo (default 0.75)
- **addNewTestProbability**: [0,1] probabilidad de insertar un nuevo test generado aleatoriamente (default 0.33)
- **modifyExistingTestProbability**: [0,1] probabilidad de modificar un test existente aleatoriamente (default 0.33)
- **removeExistingTestProbability**: [0,1] probabilidad de eliminar un test existente (default 0.33)

Algorithm 1 The genetic algorithm applied in EvoSuite

```
1  current_population ← generate random population
2  repeat
3     Z ← elite of current_population
4     while |Z| ≠ |current_population| do
5        P1,P2 ← select two parents with rank selection
6        if crossover probability then
7           O1,O2 ← crossover P1,P2
8        else
9           O1,O2 ← P1,P2
10       mutate O1 and O2
11       f_P = min(fitness(P1),fitness(P2))
12       f_O = min(fitness(O1),fitness(O2))
13       l_P = length(P1) + length(P2)
14       l_O = length(O1) + length(O2)
15       T_B = best individual of current_population
16       if f_O < f_P ∨ (f_O = f_P ∧ l_O ≤ l_P) then
17          for O in {O1,O2} do
18             if length(O) ≤ 2 × length(T_B) then
19                Z ← Z ∪ {O}
20             else
21                Z ← Z ∪ {P1 or P2}
22       else
23          Z ← Z ∪ {P1,P2}
24    current_population ← Z
25 until solution found or maximum resources spent
```

TP1: TestCall

- Es una invocación a una función bajo test usando una lista de argumentos
- Por ejemplo:
  - `cgi_decode(“Hello World”)`

TP1: TestArgument

- Es un valor para ser usado como argumento en un **TestCall**
- Se compone de:
  - Un tipo: ‘constant’, ‘list’, ‘tuple’, ‘dict’, ‘set’
  - El valor: 0, 1, [], [0], {“Juan”:0} que coincide con el valor

TP1: TestExecutor

- Esta clase se encarga de ejecutar un TestCall
- Crea un TestExecutor para cgi_decode.py:

```
exec = TestExecutor(“cgi_decode.py”)
```

- Permite ejecutar un TestCall usando funciones de cgi_decode.py:

```
err_code exec.execute(test_call)
```

- Los valores posibles de err_code son “OK”, “Exception” o “TypeError”

TP1: TestWriter

- Esta clase permite escribir un TestCall como un archivo .py
- Crea un nuevo TestWriter que escribirá el archivo como un test para el módulo cgi_decode en el directorio “output”

```
w = TestWriter(“cgi_decode”,”output”)
```

- Escribe un nuevo archivo que contendrá el test_call con el sufijo 10

```
w.writeToFile(test_call,10)
```

TP1: TestGenerator

- Es la clase encargada de generar aleatoriamente un TestCall
- Crear un nuevo TestGenerator para el módulo “cgi_decode” usando la semilla 0

```
g = TestGenerator(“cgi_decode”,0)
```

- Genera aleatoriamente un nuevo test call para el módulo indicado en el constructor

```
t = g.generate_new_test_call()
```

TP2: EvoPy

```
class EvoPy:

    def generate_tests(self, targetPyFile,
    maxGenerations, maxTime, output_dir,
    random_seed):

        # COMPLETAR

        return None
```

TP2: Enunciado

- Completar la clase **EvoPy** para tener un whole-test search based generator basado en algoritmos genéticos
- En la carpeta “examples” está el benchmark a utilizar
- **EvoPy** debe funcionar correctamente (ie no crashear) con todos los examples y usando distintas semillas de aleatoriedad

TP2: Evaluación

- Ejecutar 10 veces EvoPy sobre cada uno de los módulos de la carpeta "examples" con maxTime=120
- Reportar el promedio de cobertura de líneas y de branches

Example	Avg. Line Cov.	Avg. Branch Cov.
arrays.py		
cgi_decode.py		
convexhull.py		
coord.py		
encryption.py		
levenshtein.py		
persons.py		
sets.py		
sorting.py		
triangle.py		
tuples.py		
years.py		

TP2: Requisitos

- Haber terminado el Taller #2!
- Haber terminado el TP1!
- Python >=3.4
- Python 2.7 + Pyntch
- Eclipse (>= 4.5.0) + PyDev

Entrega: Miércoles 9 de Noviembre