Algorithm

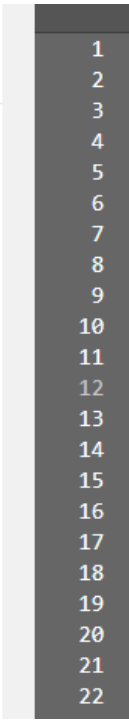
Algorithm overview

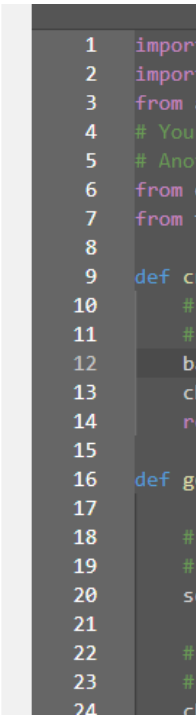
1. Create the base population We create a random initial population. Each individual is is defined by its genetic material. We create a new individual with the previously declared `create_chromosome(size)` function.
2. Evaluation Each individual is scored on its fitting to the problem. This is done in the beginning of the selection.
3. Selection Each individual has a chance to be retained proportional to the way it fits the problem. We only keep the selected individuals returned by the `selection(population)` function.
4. Crossover / reproduction Random couples are formed in the selected population. Each couple produces a new individual. The number of individuals in the population can either be constant or vary over time.

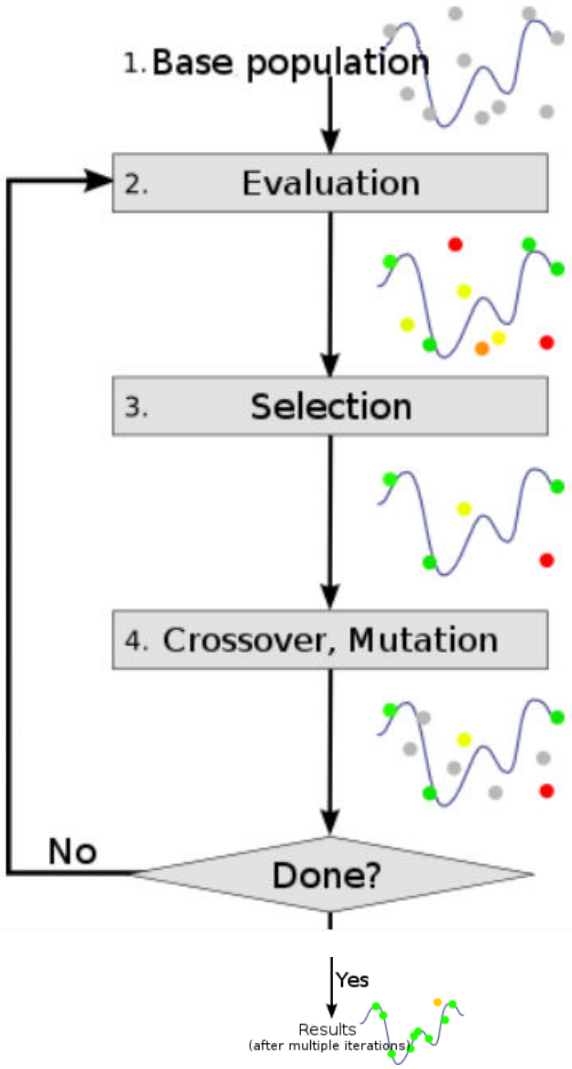
On each reproduction :

- Crossover The genetic material of a child is a combination of the parents' (generally 50% of each parent's genetic material). Once the parents have been chosen the ``crossover(parent1, parent2)` function allows the creation of the child.
- Mutation Probability : from 0.1% to 1% Each child have a chance to have a randomly modified gene thanks to the `mutation(chromosome)` function.

Finally, the `is_answer(chromosome)` function checks if the individual is solution to the problem (100% score). If there is no solution, we go to the next generation (phase 2).







*algorithm.py

```
import random
import sys
from answer imp
# You can redef
# Another imple
from encoding i
from tools impo

def create_popu
    # use the p
    # TODO: cre
    ba
    chrom = cre
    return ???

def generation(

    # selection
    # use the s
    select = se

    # reproduct
    # As long a
    children =
    # TODO: imp
    while len(c
```

The goal of this exercise is to find the secret sentence using a genetic algorithm and the tools we created earlier.

Genetic algorithm

Run

```
import random
import sys
from answer import is_answer, get_mean_score
# You can redefine these functions with the ones you wrote previously.
# Another implementation is provided here.
from encoding import create_chromosome
from tools import selection, crossover, mutation

def create_population(pop_size, chrom_size):
    # use the previously defined create_chromosome(size) function
    # TODO: create the base population
    ba
    chrom = create_chromosome(chrom_size)
    return ???

def generation(population):

    # selection
```

Genetic algorithm

Success!

Standard Error

```
....
----------------------------------------------------------------------
Ran 4 tests in 5.100s

OK
```

Standard Output

```
Well done !
This sentence is harder.
You REALLY understood GENETIC ALGORITHMS !! Congratulations !
jNxbthOwkccnaiqpfooJmxfIBqZNuZJYMouIbHLbDRFAcaZhgExowygcfNonxmNUGYdITZ
JQXnqxgAEZHkaljGHGadgAxRIWArGV
```

algorithm

```
1  import
2  import
3  import
4  from an
5  #·You·c
6  #·Anoth
7  #·from·
8  from to
9
10 def get
11     ret
12
13 def cre
14     #·T
15     str
16     ret
17
18 def sco
19     #·f
20     #·W
21     ret
22
23 def sel
```