Descripción

Este programa implementa un algoritmo de evolución genética para encontrar una cadena de caracteres que coincida con una cadena objetivo dada. Utiliza un conjunto de genes disponibles para generar cadenas aleatorias y luego las muta para mejorar su aptitud hasta que se encuentra la cadena objetivo.

Variables

- **geneSet**: `str`
  - Conjunto de caracteres disponibles para generar y mutar cadenas.
  - Valor: `' abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!.'`
- **target**: `str`
  - Cadena objetivo que el algoritmo intenta encontrar.
- **starRate**: `datetime.datetime`
  - Tiempo de inicio del programa para medir el tiempo transcurrido.
- **bestParent**: `str`
  - Mejor cadena generada hasta el momento.
- **bestFitness**: `int`
  - Aptitud de la mejor cadena generada hasta el momento.
- **child**: `str`
  - Última cadena mutada generada.
- **childFitness**: `int`
  - Aptitud de la última cadena mutada generada.

Funciones

- **generate_parent(length)**
  - Genera una cadena aleatoria de una longitud dada usando los genes disponibles.
  - Parámetros:
    - `length ( int )`: Longitud de la cadena a generar.
  - Retorna:
    - `str` : Cadena generada aleatoriamente.
- **get_fitness(guess)**
  - Calcula la aptitud de una cadena comparándola con el objetivo.
  - Parámetros:
    - `guess ( str )`: Cadena a evaluar.
  - Retorna:
    - `int` : Aptitud de la cadena.
- **mutate(parent)**
  - Muta una cadena cambiando un carácter aleatorio.
  - Parámetros:
    - `parent ( str )`: Cadena a mutar.
  - Retorna:
    - `str` : Cadena mutada.
- **display(guess)**
  - Muestra la cadena, su aptitud y el tiempo transcurrido.

- Parámetros:
  - `guess` (`str`): Cadena a mostrar.

Algoritmo

1. Se genera una cadena aleatoria inicial (`bestParent`) y se calcula su aptitud (`bestFitness`).
2. En un bucle, se generan nuevas cadenas mutadas (`child`) a partir de la mejor cadena actual.
3. Se calcula la aptitud de la nueva cadena mutada (`childFitness`).
4. Si la nueva cadena mutada tiene una aptitud mayor que la mejor cadena actual, se actualiza la mejor cadena y su aptitud.
5. El proceso se repite hasta que se encuentra una cadena que coincida con la cadena objetivo.

Ejecución

El programa se ejecuta generando una cadena aleatoria inicial y luego mutándola en un bucle hasta que se encuentra la cadena objetivo. En cada iteración, se muestra la cadena generada, su aptitud y el tiempo transcurrido.

In [1]:
```
import datetime
import random
import os
```

In [2]:
```
geneSet = " abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTUVWXYZ!."
target = input("Introduce la cadena a adivinar: ")

# random.seed(2)  # Fija la semilla para el generador de números aleatorios para reproducibi
starRate = datetime.datetime.now()  # Guarda el tiempo de inicio

def generate_parent(length):
  """Genera una cadena aleatoria de una longitud dada usando los genes disponibles."""
  genes = []
  while len(genes) < length:

    sampleSize = min(length - len(genes), len(geneSet))
    genes.extend(random.sample(geneSet, sampleSize))
  return ''.join(genes)

def get_fitness(guess):
  """Calcula la aptitud de una cadena comparándola con el objetivo."""
  return sum(1 for expected, actual in zip(target, guess) if expected == actual)

def mutate(parent):
  """Muta una cadena cambiando un carácter aleatorio."""
  index = random.randrange(0, len(parent))
  childGenes = list(parent)
  newGene, alternate = random.sample(geneSet, 2)
  childGenes[index] = alternate if newGene == childGenes[index] else newGene
  return ''.join(childGenes)

def display(guess):
  """Muestra la cadena, su aptitud y el tiempo transcurrido."""
  timeDiff = datetime.datetime.now() - starRate
  fitness = get_fitness(guess)
  text="{0}\t{1}\t{2}".format( fitness, str(timeDiff),guess)
  print(text)



# Genera el primer padre aleatorio y calcula su aptitud
bestParent = generate_parent(len(target))
bestFitness = get_fitness(bestParent)
display(bestParent)

# Bucle de evolución
while True:
  child = mutate(bestParent)  # Genera un hijo mutado
```

```
  childFitness = get_fitness(child)  # Calcula la aptitud del hijo

  if bestFitness >= childFitness:
    #display(child)
    continue  # Si el hijo no es mejor, continúa con el siguiente ciclo

  display(child)  # Muestra el hijo

  if childFitness >= len(bestParent):
    break  # Si el hijo es perfecto, termina el bucle

  bestFitness = childFitness  # Actualiza la mejor aptitud
  bestParent = child  # Actualiza el mejor padre
  display(bestParent)
```

```
0       0:00:00.001354  KwZñ
1       0:00:00.002323  Kwlñ
1       0:00:00.002394  Kwlñ
2       0:00:00.003725  Kolñ
2       0:00:00.003781  Kolñ
3       0:00:00.005641  holñ
3       0:00:00.005691  holñ
4       0:00:00.007486  hola
```