Temas 4 - Funciones Fecha de entrega 29/11/21 23:55h

Actividades a entregar [1]

IMPORTANTE: Una parte de la calificación (15%) valorará los comentarios (documentación de la función y aclaraciones dentro del código).

1. Codificar una función:

   Nombre: `mas_larga()`
   Parámetros: dos cadenas (sólo contienen letras)
   Devuelve: una nueva cadena **ordenada**
   Propósito: concatena las cadenas, eliminando letras repetidas

   sorted(iterable) ordena un *iterable*, es decir, una secuencia o colección.

   La función `assert (condicion)` [2] comprueba si se cumple la condición; si es así devuelve `True`, en caso contrario el programa se para y devuelve `AssertionError` [3]. Para probar el comportamiento de la función del ejercicio añadid, después de la declaración de la función, cinco llamadas con diferentes cadenas y comprobad si se obtiene el resultado esperado. Podría ser algo así:

```
cad1 = "xyaabbccccceffwwz"
cad2 = "aaaxxxxyyyyklmopq"
assert (mas_larga(cad1, cad2) == "abcefklmopqwxyz")

cad1 = "aacceeggffddbb"
assert (mas_larga(cad1, cad1) == "abcdefg")

cad1 = "xyaabbc"
cad2 = "aaaxxxxyyyyklmopq"
assert (mas_larga(cad1, cad2) != "abcefklmopqwxyz")

cad1 = "zzzzzzzzzxxxxxa"
cad2 = "zzzzzzzzzxxxxxb"
assert (mas_larga(cad1, cad2) != "bxz")
```

   En los casos 1 y 2 comparamos el valor de retorno de la función con un resultado esperado. En los casos 3 y 4 comprobamos que *no devuelve* dicho resultado.

   Procurad que las pruebas evalúen situaciones especiales.

---

[1] Para la resolución sólo es posible usar conceptos del tema actual y los temas previos (y lo que establezca el propio enunciado del problema)
[2] + info: https://www.programiz.com/python-programming/assert-statement
[3] Probad con la sentencia `assert (1 == 2)`

2. Cada proteína consiste en una cadena de aminoácidos diferentes cuyo recuento manual es una tarea tediosa. Codificar una función:

   Nombre: `porcentaje_amino()`
   Parámetros: una secuencia proteínica y una lista de aminoácidos (ambas correctas). Definir la lista de aminoácidos *por defecto* como ['A', 'V', 'L', 'I', 'M', 'P', 'F', 'W'].
   Devuelve: número entero[4]
   Propósito: calcular el porcentaje de dichos aminoácidos en la secuencia

   Proponed al menos 5 pruebas con sentencias `assert`. Procurad que las pruebas evalúen situaciones especiales.

3. Codificar una función:

   Nombre: `valida_secuencia()`
   Parámetros: una secuencia de bases del conjunto {AGTCU} (en mayúsculas) y, opcionalmente, un tipo de secuencia: 'ADN', 'ARN' o 'ADN/ARN'. La *opción por defecto* será 'ADN'.
   Devuelve: True o False
   Propósito: determinar si la secuencia de bases corresponde al tipo especificado.

   Proponed al menos 5 pruebas con sentencias `assert`. Procurad que las pruebas evalúen situaciones especiales.

   No se podrá utilizar el método `count`. La secuencia será siempre correcta y no contendrá a la vez la T y la U.

[4] Usar la función `trunc` o int del módulo `math`

4. Codificar una función que a partir de dos cadenas de bases de la misma longitud y un **número variable** (mayor que cero) de parámetros enteros, que corresponden a posiciones de ambas cadenas, compruebe que en las dichas posiciones de ambas cadenas hay bases diferentes, insertando un * en la cadena de salida, y que las bases son iguales, insertando un = .

   Proponed al menos 5 pruebas con sentencias `assert`. Procurad que las pruebas evalúen situaciones especiales.

   Documentad la función siguiendo el mismo formato que se ilustra en los ejercicios previos.

5. Se puede modelar una mutación de genes simplemente reemplazando una base de una cadena ADN por una letra aleatoria del alfabeto {A, C, G y T}.
   a. Construir un **programa** que a partir de una secuencia ADN realice un número de mutaciones (puede ser fijo o solicitarlo al usuario) y que imprima las frecuencias de cada base antes y después de la mutación (puede usarse el código ya desarrollado para actividades de temas previos).
   b. Desarrollar **4 funciones** para: introducir por teclado una cadena correcta ADN, realizar una mutación aleatoria, calcular la frecuencia de las bases e imprimir dichas frecuencias. La función que realiza la mutación deberá elegir aleatoriamente qué posición de la cadena mutar, y elegir aleatoriamente por qué base sustituirla.
   c. Colocar las funciones en un módulo `utilsADN.py` que importará el programa (usad `import` o `from`)
   d. Documentar las funciones siguiendo el mismo formato que se ilustra en los ejercicios previos. Comentar el código.