Ejercitación № 7: Tuplas

Cátedra Redictado Programación II Abril 2022

1. Ejercicios Simples con Tuplas

Desarrollar cada uno de los siguientes ejercicios de forma completa, detallando cada una de las etapas involucradas en la construcción de un programa, tal cual las presentamos en la receta. Para la etapa de testing de nuestro programa se deberá emplear el módulo `pytest`. Tener en cuenta que deberá crear funciones que permitan el pasaje de argumentos para posteriormente testear el programa contra los datos de testing.

Ejercicio 1. Las cartas como tuplas

1. Proponga una representación con tuplas para las cartas de la baraja francesa.
2. Escriba una función de creación de un elemento del tipo ***Carta***, la cual tome dos datos, el valor y el palo, de una carta, y retorne un elemento de tipo ***Carta*** válido.
3. Escriba una función **es_poker** que reciba cinco cartas de la baraja francesa e informe (devuelva el valor lógico correspondiente) si esas cartas forman o no un *poker* (es decir si hay cuatro cartas con el mismo número).

Ejercicio 2. El tiempo como tuplas

1. Proponer una representación con tuplas para modelar el tiempo: `hh:mm:ss`.
2. Escriba una función de creación de un elemento del tipo ***Tiempo***, el cual tome tres datos, la hora, los minutos, y los segundos, y retorne un elemento de tipo ***Tiempo*** válido.
3. Escribir una función **suma_Tiempo** que reciba dos tiempos dados y devuelva la suma de éstos.
4. Escribir una función **resta_Tiempo** que reciba dos tiempos dados y devuelva la resta de éstos.

Ejercicio 3. La fecha como tuplas

1. Proponer una representación con tuplas para modelar una fecha: `dd:mm:aaaa`.
2. Escriba una función de creación de un elemento del tipo ***Fecha***, el cual toma tres datos, el día, el mes, y el año; y retornando un elemento de tipo ***Fecha*** válido. Los valores serán enteros en este caso.
3. Escribir una función llamada **diaAnteriorNum** que dada una fecha expresada como la terna *(Día, Mes, Año)* (donde Día, Mes y Año son números enteros) calcule el día anterior al dado, en el mismo formato.
4. Escriba una función **diaAnteriorTex** que dada una fecha expresada como la terna *(Día, Mes, Año)* (donde Día y Año son números enteros, y Mes es el texto: Ene, Feb, ..., Dic, según corresponda) calcule el día anterior al dado, en el mismo formato.

Ejercicio 4. El dominó como tupla

1. Proponer una representación con tuplas para modelar una ficha de dominó.
2. Escriba una función de creación de un elemento del tipo ***Dominó***, el cual toma dos enteros, y retorna una ficha de ***Dominó*** válido. Los valores serán enteros en este caso.

3. Escriba una función que indique si dos fichas de dominó encajan o no. Las fichas son recibidas en dos tuplas, por ejemplo: $(3,4)$ y $(5,4)$.
4. Escriba una función que indique si tres fichas de dominó encajan o no. Utilice las funciones anteriores.

Ejercicio 5. Una persona como tupla

1. Proponer una representación con tuplas para modelar una persona.
2. Escriba una función de creación de un elemento del tipo ***Persona***, el cual toma cinco datos de una persona, como por ejemplo, nombre, apellido, dni, teléfono, y edad, y retorna una elemnto de tipo ***Persona*** válido.
3. Escriba una función que indique si dos ***Personas*** son familiares. Una persona es familiar de otra, si tiene el mismo apellido, y dirección.
4. Diseñe por lo menos dos tipos tupla que contenga al tipo ***Persona*** como dato.

Ejercicio 6. Varias tuplas....

Diseñar mediante el uso de tuplas los tipos correspondientes a cada uno de las siguientes entidades o abstracciones del mundo real.

1. Productos de supermercado.
2. Animales de un zoológico.
3. Libros de una biblioteca.
4. Tickets de una cafetería.
5. Pacientes de un hospital.
6. Juegos de una ludoteca.

Elegir tres diseños de las entidades anteriores, y para cada uno de ellos agregar:

1. una *función de creación* del tipo definido de forma tal que permita la creación de elementos validos del tipo,
2. *funciones de seteo* o actualización para cada campo de la tupla.
3. *funciones de impresión* para cada tipo de forma tal que los datos se muestren presentables, indentados, explicados, separados, etc, embellecidos podríamos decir: pretty-print.

2. Reveemos ejercicios de Racket en Python

Las estructuras que vimos en Racket, son equivalentes al concepto de tuplas de Python. En esta sección daremos una nueva versión de los problemas resueltos en Racket, pero ahora programándolos en Pyhton. Reutilizar la receta de los ejercicios, responder en cada problema, cuáles fueron las ventajas y las desventajas de cada solución propuesta.

Ejercicio 7. En este ejercicio representaremos una persona mediante una estructura con 5 campos:

- 1er campo: el nombre y apellido
- 2do campo: el valor numérico de su peso
- 3er campo: un string que representa la unidad en la cual está dado el peso (valores posibles: “G” o“Kg”)
- 4to campo: el valor numérico de la estatura

- 5to campo: un string que representa la unidad en la cual está dada la estatura (valores posibles: “Mt” o “Cm”)

Teniendo en cuenta esto se pide:

Diseñe una estructura persona que le permita representar a cualquier persona con su peso y estatura.

Diseñe una función IMC que tome como entrada un valor de tipo persona y calcule su índice de masa corporal. En caso que no reciba como entrada una estructura de tipo persona deberá mostrar el siguiente mensaje de error: "Tipo de dato inválido".

Ayuda 1: Recuerde que cada estructura que define viene acompañada de un predicado que determina si un objeto es o no una estructura de ese tipo. Ayuda 2: Por si no lo sabe, el índice de masa corporal (IMC) de una persona se calcula según la siguiente fórmula:

$$\frac{peso}{altura^2}$$

Ayuda 3: Le puede servir conocer las siguientes equivalencias:

1 Kg = 1000 G

1 Mts = 100 Cms

Ejercicio 8. En este ejercicio representaremos un auto con una tupla con los siguientes campos:

- 1er campo: Modelo.
- 2do campo: Año.
- 3er campo: Patente.
- 4to campo: Tipo de combustible (diesel o nafta).
- 5to campo: Rendimiento óptimo, expresado en kilómetros por litro.

Teniendo en cuenta esto se pide:

1. Diseñe una tupla auto que contenga los campos descriptos más arriba.
2. Diseñe una función `costo-viaje` que tome como entrada un valor de tipo auto y un número de kilómetros a recorrer y calcule el costo del viaje. Para esto debe tener en cuenta:
   - Cantidad de combustible: el número de litros necesarios para recorrer m kilómetros con un auto nuevo está determinado por su rendimiento óptimo. Sin embargo, con el correr de los años, el rendimiento disminuye. Se estima que si el auto tiene:
     - Entre 1 y 5 años, el rendimiento disminuye 2 %
     - Entre 6 y 10 años, el rendimiento disminuye 6 %
     - Entre 10 y 15 años, el rendimiento disminuye 10 %
     - Más de 15 años, el rendimiento disminuye 15 %

     Es decir, si un auto 0km rinde 13km/litro, después de un año rendirá 12.74 km/litro, y después de 12 años 11,7 km/litro.
   - Peajes: por cada 100 kilómetros recorridos, se debe pagar un peaje de $50.
   - Precio combustible: el precio actual del litro de nafta es $19 y el litro de diesel $17.

   Ejemplo: Un gol naftero 2013 de rendimiento óptimo 13km/litro, debido a sus 3 años de antigüedad tendrá un rendimiento de 12,74 km/litros. Por lo tanto, recorrer 450 kms tendrá un costo de: $(450/12,74) * 19\$ + 200\$$

Ejercicio 9. En este ejercicio representaremos un alumno con una estructura con los siguientes campos:

- 1er campo: Nombre del alumno.
- 2do campo: Promedio de sus calificaciones (un valor entre 0 y 10).
- 3er campo: Porcentaje de asistencia a clases (un valor entre 0 y 100).

Teniendo en cuenta esto se pide:

1. Diseñe una tupla `alumno` que contenga los campos descriptos más arriba.
2. Diseñe una función `condición` que tome como entrada un valor de tipo alumno, y devuelva un string indicando su condición. Las condiciones posibles son: *"Libre"*, *Regular"* y *"Promovido"*. Para calcular la condición del alumno deben tenerse en cuenta las siguientes reglas:
   - Si el alumno tiene un porcentaje de inasistencia mayor al 40 % queda automáticamente *libre*, sin importar el promedio de sus calificaciones.
   - Si el alumno tiene una asistencia mayor o igual al 60 %, y tiene una nota inferior a 6, también se considera *libre*.
   - Si el alumno tiene una asistencia mayor o igual al 60 %, y una nota mayor o igual a 6 y menor estricta que 8, se considera *regular*.
   - Si el alumno tiene una asistencia mayor o igual al 60 %, y una nota mayor o igual a 8, se considera *promovido*.

   En el caso de que la función `condición` reciba como entrada un dato que no corresponda a una tuplaalumno deberá responder con un mensaje de error (como por ejemplo: "Tipo de dato inválido"), y retornar una tupla designada para tal efecto.
3. Modifique el diseño de datos alumno, para agregar la información de la condición de cada alumno. Luego utilizando la función `condición`, diseñe una función llamada, `genera-condición`, la cual recibe como datos un alumno, y actualiza el `campo condición`, con la evaluación de la función `condición`.

Referencias

[1] Think Python: How to Think Like a Computer Scientist, Allen B. Downey, 2nd Edition, Version 2.2.18.

[2] Algoritmos y Programación I, Aprendiendo a programar usando Phyton como herramienta, Rosita Wachenchauzer et.al., 2016, (sin publicar).