Tarea 8 - Estructuras de datos: Tuplas

Curso de `Python`

Ejercicio 1

Pide al usuario el número de números enteros que va a introducir por teclado. Para cada uno de esos números, crea una tupla donde la primera entrada sea el número entero y, la segunda, la palabra "par" o "impar" según la paridad del número entero. Muestra la tupla recién creada al usuario.

Solución

```
n = int(input("¿Cuántos números enteros vas a introducir? "))
nums = []

for _ in range(n):
  isEven = False
  num = int(input())
  if num % 2 == 0:
    isEven = True
  nums.append((num, "par" if isEven else "impar"))

print(nums)
```

Ejercicio 2

Dado un año proporcionado por el usuario, crea una tupla de dos elementos cuya primera entrada sea el año y, la segunda entrada, el horóscopo chino correspondiente.

Solución

```
year = int(input("Introduce un año "))

horoscopes = ["Mono", "Gallo", "Perro", "Cerdo", "Rata", "Buey",
              "Tigre", "Conejo", "Dragón", "Serpiente", "Caballo", "Cabra"]

horoscope = horoscopes[year % 12]

result = (year, horoscope)
print(result)
```

Ejercicio 3

Dada una frase proporcionada por el usuario, crea una lista de tuplas indicando palabra, longitud de cada palabra, letra inicial y posición que ocupan dentro de la frase.

Solución

```
s = input()
s = s.lower()
words = s.split(" ")
info = []

for i in range(len(words)):
  info.append((words[i], len(words[i]), words[i][0], i))

print(info)
```

Ejercicio 4

Haz que el usuario introduzca palabras hasta que introduzca una palabra vacía. Guarda todas las palabras en una tupla y muestra la primera y la última introducidas haciendo uso del método unpacking.

PISTA: Para guardar los elementos de uno en uno vas a tener que utilizar un tipo de dato que no es tupla y luego transformarlo a tupla.

Solución

```
w = input()
words = []

while w != "":
  words.append(w)
  w = input()

words = tuple(words)

first, *_, last = words
print("Primera palabra:", first)
print("Última palabra:", last)
```

Ejercicio 5

Dada una lista de palabras, crea otra lista del mismo tamaño que guarde la longitud de cada palabra. Usa la función `zip()` para crear un diccionario con claves las palabras y valores, su longitud.

Solución

```
words = ["ola", "caracola", "piña", "playa"]
words_len = []

for w in words:
  words_len.append(len(w))

print(dict(zip(words, words_len)))
```

Ejercicio 6

Haz que el usuario introduzca palabras hasta que introduzca una palabra vacía. Guarda todas las palabras en una tupla y muestra la tupla y el número total de caracteres que ha introducido.

PISTA: Para guardar los elementos de uno en uno vas a tener que utilizar un tipo de dato que no es tupla y luego transformarlo a tupla.

Solución

```
w = input()
words = []

while w != "":
  words.append(w)
  w = input()

words = tuple(words)
total_chrs = 0

for w in words:
  total_chrs += len(w)

print("Tus palabras han sido {} y suman {} caracteres".format(words, total_chrs))
```

Ejercicio 7

Crea una lista de 20 tuplas de tamaño 2. La primera entrada será un número entero entre 1 y 20 y la segunda entrada contendrá una lista con los 10 primeros múltiplos del número entero correspondiente. Por último, muestra las tablas de multiplicar del 1 al 20 con el formato "1 x 1 = 1".

Solución

```
multiplication_tables = []

for i in range(1, 21):
  multiplication_tables.append((i, [i * j for j in range(1, 11)]))

for item in multiplication_tables:
  print("\n=== TABLA DEL {}===".format(item[0]))
```

```
  for multiple in item[1]:
    print("{} x {} = {}".format(item[0], item[1].index(multiple) + 1, multiple))
```

Ejercicio 8

Pide al usuario dos números enteros por teclado. Asegúrate de que el primero es mayor o igual al segundo. Realiza la división entera y guarda en una tupla el dividendo, el divisor, el cociente y el resto de la divisón entera realizada y muéstrale al usuario el resultado por pantalla.

Solución

```
dividend = int(input("Introduce un número entero: "))
divisor = int(input("Introduce un número entero menor o igual al anterior: "))

if divisor > dividend:
  print("El segundo número entero que introduzcas debe ser menor o igual al primero")
else:
  print((dividend, divisor,  dividend // divisor, dividend % divisor))
```

Ejercicio 9

Pide al usuario un número entre 0 y 360. Para dicho número, crea una tupla y muéstrala donde la primera entrada sea dicho número y, la segunda, la medida angular correspondiente en radianes. Recuerda, $360° = 2\pi$rad. En este caso, utiliza $\pi = 3.141592653589793$

Solución

```
degree = float(input("Introduce un número del intervalo [0, 360] "))

print((degree, degree * 2 * 3.141592653589793 / 360))
```

Ejercicio 10

Pide al usuario un número complejo. Para dicho número, crea una tupla donde la primera entrada sea dicho número complejo, la segunda, su opuesto y, la tercera, su conjugado.

Solución

```
z = complex(float(input("Introduce la parte real de un número complejo: ")),
            float(input("Introduce la parte imaginaria de un número complejo: ")))

print((z, -z, z.conjugate()))
```