IWI 131- PROGRAMACIÓN

Ayudantía 10

Ayudante: Anastasiia Fedorova

Paralelo: 212

Fecha: 17.06.2020

¿TUPLAS?

Una estructura, llamada tuple o tupla, es un conjunto de elementos, que poseen cierto orden. Es una estructura **inmutable.**

A veces las tuplas de n elementos las llaman **n-tuplas** – es bueno aprendar esta terminología, pues la van a escuchar, por ejemplo, en ramos relacionados con el análisis vectorial (ej. mat22, mat23, etc).

```
l = [1, 2, 3, 4]
```

Lista

```
t = (1, 2, 3, 4)
```

Tupla

USOS RECOMENDADOS

Una tupla sirve para agrupar elementos, que deben ir juntos. Es un tipo inmutable – una vez que creamos la tupla, no podemos volver a modificala, por lo nos que sirve para guardar valores que deben permanecer constantes.

CREACIÓN

Como en listas, podemos crear una tupla vacía de dos maneras:

```
t = tuple()
```

```
t = ()
```

Además, podemos crear una tupla con elementos definidos también de dos maneras, ambas equivalentes:

```
t = (1, 2, 3)
```

```
t = 1, 2, 3
```

DESEMPAQUETAMIENTO

Para obtener los elementos de la tupla, la podemos desempaquetar. Muy comunmente algunos valores no nos van a interesar – pero, como no podemos desempaquetar la tupla parcialmente, se usa el _ (underscore, don`t care) para ignorar el valor.

```
t = ("Peter", "Parker")
nombre, apellido = t
print(nombre)
>>> Peter
print(apellido)
>>> Parker
```

```
t = ("Monty", "Python", "Los
caballeros de la mesa cuadrada")
nombre, apellido, _ = t
print(nombre,apellido)
>>> Monty Python
```

ACCESO CON ÍNDICE

No siempre queremos desempaquetar la tupla entera (pues, puede ser que sea muy larga o incluso nos interese solo 1 valor) y por ello las tuplas también soportan el acceso por índice.

```
t = ("Monty", "Python", ”El sentido de vida")
nombre, apellido, _ = t
print(nombre == t[0])
>>> True
```

COMPARACIÓN ENTRE LAS TUPLAS

Dos tuplas se dicen iguales, si:

- Tienen el mismo largo
- Cada uno de sus elementos correspondientes tienen el mismo valor

```
(1, 2) == (3 - 2, 1 + 1)
>>> True
(9, 1) == (9, 2)
>>> False
(9, 1) == (9, 1, 12, 4)
>>> False
```

COMPARACIÓN ENTRE LAS TUPLAS

Para determinar si una tupla es menor que otra, se utiliza lo que se denomina **orden lexicográfico**. Si los elementos en la primera posición de ambas tuplas son distintos, ellos determinan el ordenamiento de las tuplas. Si los elementos respectivos siguen siendo iguales, entonces se sigue probando con los siguientes uno por uno, hasta encontrar dos distintos. Si a una tupla se le acaban los elementos para comparar antes que a la otra, entonces es considerada menor que la otra.

```
(1, 4, 7) < (2, 0, 0, 1)
>>> True

(2, 1, 8) < (2, 2, 8)
>>> True

"pera" < "peras”
>>> True
```

CONCATENACIÓN Y REPETICIÓN

Las tuplas se pueden concatenar y repetir – en este caso, no se cambia la tupla original si no se obtiene una nueva.

Repetición:

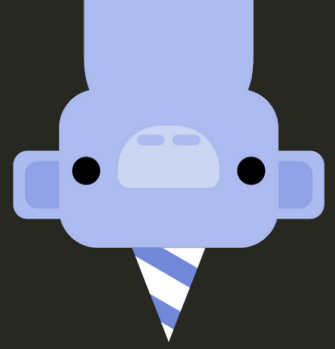
```
t = ("manzana", "pera",
"kiwi")
newt = t * 2
print(newt)

>>> ("manzana", "pera",
"kiwi", "manzana", "pera",
"kiwi")
```

Concatenación:

```
t1 = ("manzana", "pera", "kiwi")
t2 = ("mango", "durazno")
newt = t1 + t2
print(newt)
>>> ("manzana", "pera", "kiwi",
"mango", "durazno")
```

EJERCICIO 1



La plataforma de social network Datcord alamcena los datos de los usuarios en una lista de tuplas. Las tuplas contienen el nombre del usuario, la lista de servers a cuales este pertenece y la lista de sus amigos:

Escriba las siguientes funciones:

```
servers_comunes(usuarios, user1, user2)
son_amigos(usuarios, user1, user2)
usuario_social(usuarios)
```

```
usuarios = ("ayudante_supreme",
["progcomp", "iwi131", "hurones"],
["mario", "poyo", "delfin", "zapato",
"anihcanom"]), (...), ...]
```

Donde la primera retorna una lista de servers en común de 2 usuarios. La segunda, True si son amigos y False si no. El usuario más social es aquel que tiene más amigos. Asumir que no puede pasar que user1 sea amigo de user2, pero no al revés.

SOLUCIÓN

```
def servers_comunes(usuarios, user1, user2):
    lista1 = []
    lista2 = []
    res = []
    for t in usuarios:
        name, servers, _ = t
        if name == user1:
            lista1 = list(servers)
        elif name == user2:
            lista2 = list(servers)
    largo1 = len(lista1)
    largo2 = len(lista2)
    if largo1 > largo2:
        for s in lista1:
            if s in lista2:
                res.append(s)
    else:
        for s in lista2:
            if s in lista1:
                res.append(s)
    return res
```

SOLUCIÓN

```python
def son_amigos(usuarios, user1, user2):
    for t in usuarios:
        name, _, amigos = t
        if name == user1:
            if user2 in amigos:
                return True
    return False
```

```python
def usuario_social(usuarios):
    max_amigos = -1
    social = ""
    for t in usuarios:
        user, _, amigos = t
        if len(amigos) > max_amigos:
            max_amigos = len(amigos)
            social = user
    return social
```

EJERCICIO 2

Un supermercado utiliza tablas de datos para llevar la información de su inventario.

La lista productos tiene el código, el nombre, el precio y la cantidad de unidades del producto en bodega. La lista ventas contiene las ventas realizadas, representadas por el número de boleta, la fecha de la venta y el rut del cliente. El detalle de cada venta se encuentra en la lista itemes. Cada ítem tiene asociado un número de boleta, un código de producto y una cantidad.

Implementar las siguientes funciones.

```
producto_mas_caro(productos)
>>> 'Vuvuzela'

valor_total_bodega(productos)
>>> 1900570

ingreso_total_por_ventas(itemes, productos)
>>> 13944

fecha_ultima_venta_producto(47470, itemes,
ventas)
>>> (2010, 10, 13)

total_ventas_del_mes(2010, 10, itemes,
productos)
>>> 4160
```

EJERCICIO 1

```
productos = [
    (41419, 'Fideos',         450, 210),
    (70717, 'Cuaderno',       900, 119),
    (78714, 'Jabon',          730, 708),
    (30877, 'Desodorante',   2190,  79),
    (47470, 'Yogur',           99, 832),
    (50809, 'Palta',          500,  55),
    (75466, 'Galletas',       235,   0),
    (33692, 'Bebida',         700,  20),
    (89148, 'Arroz',          900, 121),
    (66194, 'Lapiz',          120, 900),
    (15982, 'Vuvuzela',     12990,  40),
    (41235, 'Chocolate',     3099,  48),
]
```

Código, nombre, costo, cantidad

```
ventas = [
    (1, (2010,  9, 12),  '8830268-0'),
    (2, (2010,  9, 19), '11652624-7'),
    (3, (2010,  9, 30),  '7547896-8'),
    (4, (2010, 10,  1),  '8830268-0'),
    (5, (2010, 10, 13),  '7547896-8'),
    (6, (2010, 11, 11), '11652624-7'),
]
```

Número de boleta, fecha, RUT

```
itemes = [
    (1, 89148,  3),
    (2, 50809,  4),
    (2, 33692,  2),
    (2, 47470,  6),
    (3, 30877,  1),
    (4, 89148,  1),
    (4, 75466,  2),
    (5, 89148,  2),
    (5, 47470, 10),
    (6, 41419,  2),
]
```

Número de boleta, código de producto, cantidad

SOLUCIÓN

```python
def producto_mas_caro(productos):
    maxcosto = -1
    maxprod = ""
    for t in productos:
        if t[-2] > maxcosto:
            maxcosto = t[-2]
            maxprod = t[1]
    return maxprod
```

```python
def valor_total_bodega(productos):
    total = 0
    for t in productos:
        _, _, costo, cantidad = t
        total += costo * cantidad
    return total
```

SOLUCIÓN

```
def ingreso_total_por_ventas(itemes,
productos):
    total = 0
    for boleta in itemes:
        _, codigo, cantidad = boleta
        for t in productos:
            if t[0] == codigo:
                total += cantidad * t[-2]
    return total
```

```
def fecha_ultima_venta_producto(codigo,
itemes, productos):
    boletas = []
    ultima_fecha = ()
    for item in itemes:
        num, cod, _ = item
        if cod == codigo:
            boletas.append(num)
    for venta in ventas:
        if venta[0] in boletas:
            if venta[1] > ultima_fecha:
                ultima_fecha = venta[1]
    return ultima_fecha
```

SOLUCIÓN

```
def total_ventas_del_mes(a, mes, itemes, productos):
    total_mes = 0
    boletas = []
    for venta in ventas:
        # recolectamas boletas del mes
        if venta[1][0] == a and venta[1][1] == mes:
            boletas.append(venta[0])
    for item in itemes:
        num, cod, cantidad = item
        # si el numero de item esta en las boletas de mes
        if num in boletas:
            for producto in productos:
                cd, _, costo, _ = producto
                if cod == cd:
                    total_mes += costo * cantidad
    return total_mes
```