Clase Número: 11

Estructura de datos tipo diccionario

Hasta ahora hemos presentado dos estructuras fundamentales de datos en Python: listas y tuplas. Ahora presentaremos y comenzaremos a utilizar la estructura de datos tipo diccionario.
La estructura de datos tipo diccionario utiliza una clave para acceder a un valor. El subíndice puede ser un entero, un float, un string, una tupla etc. (en general cualquier tipo de dato inmutable)

Podemos relacionarlo con conceptos que conocemos:

- Un diccionario tradicional que conocemos podemos utilizar un diccionario de Python para representarlo. La clave sería la palabra y el valor sería la definición de dicha palabra.
- Una agenda personal también la podemos representar como un diccionario. La fecha sería la clave y las actividades de dicha fecha sería el valor.
- Un conjunto de usuarios de un sitio web podemos almacenarlo en un diccionario. El nombre de usuario sería la clave y como valor podríamos almacenar su mail, clave, fechas de login etc.

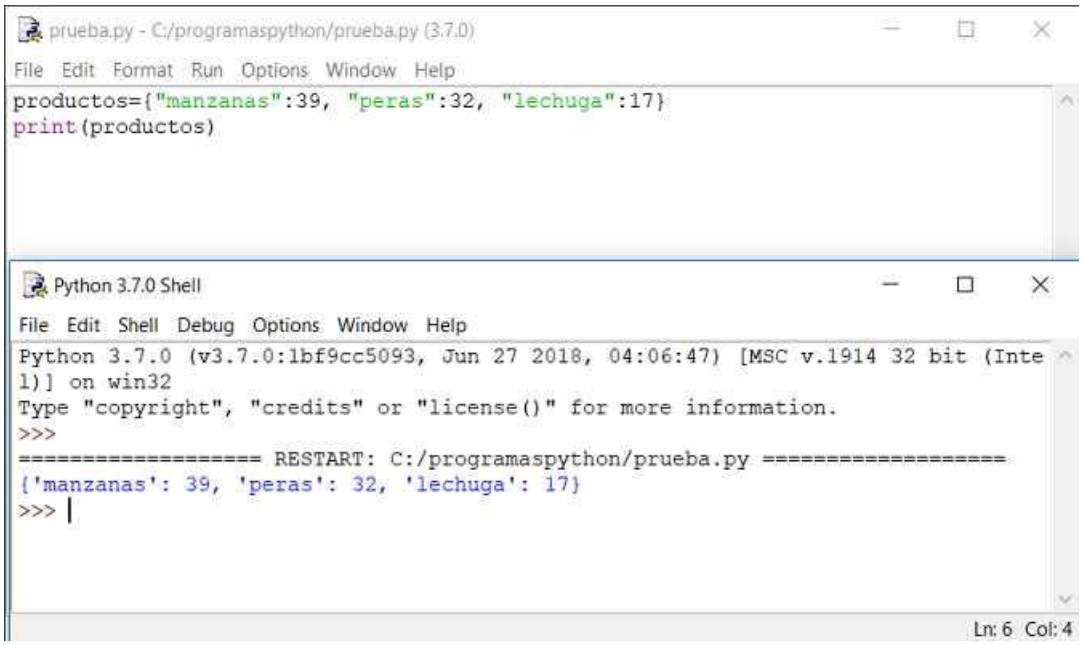
Hay muchos problemas de la realidad que se pueden representar mediante un diccionario de Python.

Recordemos que las listas son mutables y las tuplas inmutables. Un diccionario es una estructura de datos mutable es decir podemos agregar elementos, modificar y borrar.

Definición de un diccionario por asignación.

```
productos={"manzanas":39, "peras":32, "lechuga":17}
print(productos)
```

Como vemos debemos encerrar entre llaves los elementos separados por coma. A cada elementos debemos indicar del lado izquierdo del caracter : la clave y al lado derecho el valor asignado para dicha clave. Por ejemplo para la clave "peras" tenemos asociado el valor entero 32.



Problema 1:

En el bloque principal del programa definir un diccionario que almacene los nombres de paises como clave y como valor la cantidad de habitantes. Implementar una función para mostrar cada clave y valor.

Programa: ejercicio159.py

```
def imprimir(paises):
    for clave in paises:
        print(clave, paises[clave])


# bloque principal

paises={"argentina":40000000, "españa":46000000, "brasil":190000000,
"uruguay": 3400000}
imprimir(paises)
```

En el bloque principal de nuestro programa inicializamos un diccionario con cuatro elementos y lo pasamos a la función imprimir:

```
paises={"argentina":40000000, "españa":46000000, "brasil":190000000,
"uruguay": 3400000}
imprimir(paises)
```

Mediante el ciclo repetitivo for podemos acceder sucesivamente a las claves almacenadas en el diccionario y luego conociendo la clave podemos acceder al valor de forma similar a las listas indicando como subíndice la clave:

```
def imprimir(paises):
    for clave in paises:
        print(clave, paises[clave])
```

Problema 2:

Crear un diccionario que permita almacenar 5 artículos, utilizar como clave el nombre de productos y como valor el precio del mismo.
Desarrollar además las funciones de:
1) Imprimir en forma completa el diccionario
2) Imprimir solo los artículos con precio superior a 100.

Programa:

```
def cargar():
    productos={}
    for x in range(5):
        nombre=input("Ingrese el nombre del producto:")
        precio=int(input("Ingrese el precio:"))
        productos[nombre]=precio
    return productos


def imprimir(productos):
    print("Listado de todos los articulos")
    for nombre in productos:
        print(nombre, productos[nombre])


def imprimir_mayor100(productos):
    print("Listado de articulos con precios mayores a 100")
    for nombre in productos:
        if productos[nombre]>100:
            print(nombre)


# bloque principal

productos=cargar()
imprimir(productos)
imprimir_mayor100(productos)
```

Para agregar elementos a un diccionario procedemos a asignar el valor e indicamos como subíndice la clave:

```
        nombre=input("Ingrese el nombre del producto:")
        precio=int(input("Ingrese el precio:"))
        productos[nombre]=precio
```

Si ya existe el nombre de producto en el diccionario se modifica el valor para esa clave.

Operador in con diccionarios

Para consultar si una clave se encuentra en el diccionario podemos utilizar el operador in:

```
if clave in diccionario:
    print(diccionario[clave])
```

Esto muy conveniente hacerlo ya que si no existe la clave produce un error al tratar de accederlo:

```
print(diccionario[clave])
```

Problema 3:

Desarrollar una aplicación que nos permita crear un diccionario ingles/castellano. La clave es la palabra en ingles y el valor es la palabra en castellano.
Crear las siguientes funciones:
1) Cargar el diccionario.
2) Listado completo del diccionario.
3) Ingresar por teclado una palabra en ingles y si existe en el diccionario mostrar su traducción.

Programa:

```
def cargar():
    diccionario={}
    continua="s"
    while continua=="s":
        caste=input("Ingrese palabra en castellano:")
        ing=input("Ingrese palabra en ingles:")
        diccionario[ing]=caste
        continua=input("Quiere cargar otra palabra:[s/n]")
    return diccionario


def imprimir(diccionario):
    print("Listado completo del diccionario")
    for ingles in diccionario:
        print(ingles,diccionario[ingles])


def consulta_palabra(diccionario):
    pal=input("Ingrese la palabra en ingles a consultar:")
    if pal in diccionario:
            print("En castellano significa:",diccionario[pal])


# bloque principal

diccionario=cargar()
imprimir(diccionario)
consulta_palabra(diccionario)
```

La función de cargar crea el diccionario y va solicitando la palabra en castellano y su traducción. Luego de agregar un elementos se solicita el ingrese de una variable string pidiendo que confirme si quiere cargar otra palabra en el diccionario o finalizar:

```
def cargar():
    diccionario={}
    continua="s"
    while continua=="s":
        caste=input("Ingrese palabra en castellano:")
        ing=input("Ingrese palabra en ingles:")
        diccionario[ing]=caste
        continua=input("Quiere cargar otra palabra:[s/n]")
    return diccionario
```

La función imprimir muestra el diccionario en forma completa:

```
def imprimir(diccionario):
    print("Listado completo del diccionario")
    for ingles in diccionario:
        print(ingles,diccionario[ingles])
```

Lo nuevo aparece cuando queremos consultar la traducción de una palabra.
Se solicita al operador que ingrese la palabra en ingles que desconoce y mediante el operador in verificamos si dicha palabra se encuentra dentro del diccionario, en caso afirmativo procedemos a mostrar el valor del diccionario, es decir la palabra en castellano:

```
def consulta_palabra(diccionario):
    pal=input("Ingrese la palabra en ingles a consultar:")
    if pal in diccionario:
            print("En castellano significa:",diccionario[pal])
```

Problema propuesto

1 - Crear un diccionario en Python que defina como clave el número de documento de una persona y como valor un string con su nombre.
Desarrollar las siguientes funciones:
a) Cargar por teclado los datos de 4 personas.
b) Listado completo del diccionario.
c) Consulta del nombre de una persona ingresando su número de documento.

Solución al problema

```
def cargar():
    personas={}
    for x in range(4):
        numero=int(input("Ingrese el numero de documento:"))
        nombre=input("Ingrese el nombre:")
        personas[numero]=nombre
    return personas


def imprimir(personas):
    print("Listado del diccionario completo")
    for numero in personas:
        print(numero, personas[numero])


def consulta_por_numero(personas):
    nro=int(input("Ingrese el numero de documento a consultar:"))
    if nro in personas:
        print("Nombre de la persona:",personas[nro])
    else:
        print("No existe una persona con dicho numero de documento")


# bloque principal

personas=cargar()
imprimir(personas)
consulta_por_numero(personas)
```

Diccionarios: con valores de tipo listas, tuplas y diccionarios

Lo más poderoso que podemos encontrar en las estructuras de datos en Python es que podemos definir elementos que sean también estructuras de datos. En general se dice que podemos anidar una estructura de datos dentro de otra estructura de datos.
Ya vimos en conceptos anteriores que podemos definir elementos de una lista que sean también de tipo lista o de tipo tupla.

Hemos dicho que un diccionario consta de claves y valores para esas claves. Desarrollaremos problemas donde los valores para esas claves sean tuplas y o listas.

Problema 1:

Confeccionar un programa que permita cargar un código de producto como clave en un diccionario. Guardar para dicha clave el nombre del producto, su precio y cantidad en stock.
Implementar las siguientes actividades:
1) Carga de datos en el diccionario.
2) Listado completo de productos.
3) Consulta de un producto por su clave, mostrar el nombre, precio y stock.
4) Listado de todos los productos que tengan un stock con valor cero.

Programa:

```
def cargar():
    productos={}
    continua="s"
    while continua=="s":
        codigo=int(input("Ingrese el codigo del producto:"))
        descripcion=input("Ingrese la descripcion:")
        precio=float(input("Ingrese el precio:"))
        stock=int(input("Ingrese el stock actual:"))
        productos[codigo]=(descripcion,precio,stock)
        continua=input("Desea cargar otro producto[s/n]?")
    return productos


def imprimir(productos):
    print("Listado completo de productos:")
    for codigo in productos:

print(codigo,productos[codigo][0],productos[codigo][1],productos[codigo][2])


def consulta(productos):
    codigo=int(input("Ingrese el codigo de articulo a consultar:"))
    if codigo in productos:
        print(productos[codigo][0],productos[codigo][1],productos[codigo][2])


def listado_stock_cero(productos):
    print("Listado de articulos con stock en cero:")
    for codigo in productos:
        if productos[codigo][2]==0:

print(codigo,productos[codigo][0],productos[codigo][1],productos[codigo][2])
```

```
# bloque principal

productos=cargar()
imprimir(productos)
consulta(productos)
listado_stock_cero(productos)
```

La función de cargar crea un diccionario llamado "productos" y mediante una estructura repetitiva añadimos en cada vuelta en el diccionario una entrada. La clave del diccionario es el código del producto y el valor del diccionario es una tupla que almacena la descripción del producto, su precio y su stock:

```
def cargar():
    productos={}
    continua="s"
    while continua=="s":
        codigo=int(input("Ingrese el codigo del producto:"))
        descripcion=input("Ingrese la descripcion:")
        precio=float(input("Ingrese el precio:"))
        stock=int(input("Ingrese el stock actual:"))
        productos[codigo]=(descripcion,precio,stock)
        continua=input("Desea cargar otro producto[s/n]?")
    return productos
```

En la función de imprimir recorremos el diccionario mediante un for in y recuperamos en cada paso una clave y mediante esta accedemos al valor que como sabemos se trata de una tupla que contiene 3 elementos:

```
def imprimir(productos):
    print("Listado completo de productos:")
    for codigo in productos:

print(codigo,productos[codigo][0],productos[codigo][1],productos[codigo][2])
```

Para la consulta por el codigo del artículo ingresamos un entero por teclado y luego verificamos si dicho número está como clave dentro del diccionario productos, en el caso afirmativo mostramos los valores de la tupla:

```
def consulta(productos):
    codigo=int(input("Ingrese el codigo de articulo a consultar:"))
    if codigo in productos:
        print(productos[codigo][0],productos[codigo][1],productos[codigo][2])
```

Finalmente la función para listar todos los artículos con stock en cero procedemos a analizar el stock de cada producto dentro de un for, en el caso que la tercer componente de la tupla almacena un cero significa que no hay productos en stock:

```
def listado_stock_cero(productos):
    print("Listado de articulos con stock en cero:")
    for codigo in productos:
        if productos[codigo][2]==0:

print(codigo,productos[codigo][0],productos[codigo][1],productos[codigo][2])
```

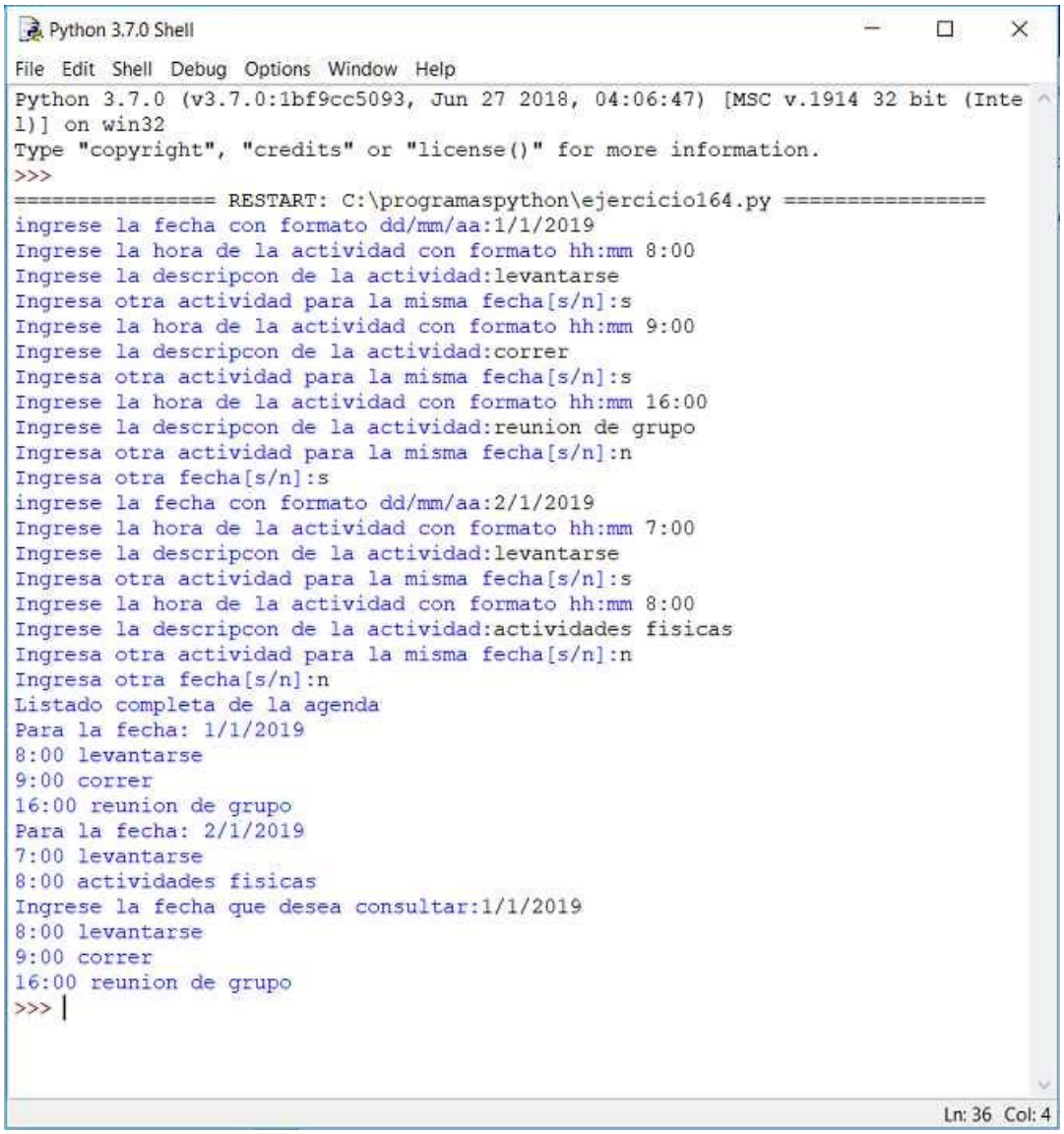
Problema 2:

Confeccionar una agenda. Utilizar un diccionario cuya clave sea la fecha. Permitir almacenar distintas actividades para la misma fecha (se ingresa la hora y la actividad)
Implementar las siguientes funciones:
1) Carga de datos en la agenda.
2) Listado completo de la agenda.
3) Consulta de una fecha.

Programa:

```
def cargar():
    agenda={}
    continua1="s"
    while continua1=="s":
        fecha=input("ingrese la fecha con formato dd/mm/aa:")
        continua2="s"
        lista=[]
        while continua2=="s":
            hora=input("Ingrese la hora de la actividad con formato hh:mm ")
            actividad=input("Ingrese la descripcon de la actividad:")
            lista.append((hora,actividad))
            continua2=input("Ingresa otra actividad para la misma
fecha[s/n]:")
        agenda[fecha]=lista
        continua1=input("Ingresa otra fecha[s/n]:")
    return agenda


def imprimir(agenda):
    print("Listado completa de la agenda")
    for fecha in agenda:
        print("Para la fecha:",fecha)
        for hora,actividad in agenda[fecha]:
            print(hora,actividad)


def consulta_fecha(agenda):
    fecha=input("Ingrese la fecha que desea consultar:")
    if fecha in agenda:
        for hora,actividad in agenda[fecha]:
            print(hora,actividad)
    else:
        print("No hay actividades agendadas para dicha fecha")


# bloque principal

agenda=cargar()
imprimir(agenda)
consulta_fecha(agenda)
```

Un ejemplo de ejecutar este programa tenemos la siguiente pantalla:



La carga de la agenda se compone de dos estructuras repetitivas anidadas.

Previo a comenzar la primer estructura repetitiva creamos el diccionario llamado agenda:

```
def cargar():
    agenda={}
```

El primer ciclo se repite mientras haya más fechas que cargar. Solicitamos que ingrese una fecha, creamos una lista para guardar todas las horas y actividades que hay par dicha fecha:

```
    continua1="s"
    while continua1=="s":
        fecha=input("ingrese la fecha con formato dd/mm/aa:")
        continua2="s"
        lista=[]
```

El segundo ciclo se repite mientras haya más actividades para el mismo día:

```
        while continua2=="s":
            hora=input("Ingrese la hora de la actividad con formato hh:mm ")
            actividad=input("Ingrese la descripcon de la actividad:")
```

```
            lista.append((hora,actividad))
            continua2=input("Ingresa otra actividad para la misma
fecha[s/n]:")
```

Cuando se terminan de cargar todas las actividades para una determinada fecha se procede a insertar la lista en el diccionario:

```
        agenda[fecha]=lista
        continua1=input("Ingresa otra fecha[s/n]:")
```

Previo a salir de la función devolvemos la agenda ya cargada:

```
    return agenda
```

Para imprimir todas las fechas y actividades por fecha también disponemos dos ciclos repetitivos anidados, en este caso for in:

```
def imprimir(agenda):
    print("Listado completa de la agenda")
    for fecha in agenda:
        print("Para la fecha:",fecha)
        for hora,actividad in agenda[fecha]:
            print(hora,actividad)
```

De forma similar para consultar las actividades de la agenda una determinada fecha procedemos a ingresar la fecha y en el caso que haya una clave en el diccionario con ese dato procedemos a recuperar la lista de actividades para dicha fecha:

```
def consulta_fecha(agenda):
    fecha=input("Ingrese la fecha que desea consultar:")
    if fecha in agenda:
        for hora,actividad in agenda[fecha]:
            print(hora,actividad)
    else:
        print("No hay actividades agendadas para dicha fecha")
```

El bloque principal no difiere de problemas anteriores donde llamamos a las tres funciones que previamente definimos:

```
# bloque principal

agenda=cargar()
imprimir(agenda)
consulta_fecha(agenda)
```

Problema propuesto

1 - Se desea almacenar los datos de 3 alumnos. Definir un diccionario cuya clave sea el número de documento del alumno. Como valor almacenar una lista con componentes de tipo tupla donde almacenamos nombre de materia y su nota. Crear las siguientes funciones:
a) Carga de los alumnos (de cada alumno solicitar su dni y los nombres de las materias y sus notas)
b) Listado de todos los alumnos con sus notas
c) Consulta de un alumno por su dni, mostrar las materias que cursa y sus notas.

Solución al problema

```
def cargar():
    alumnos={}
    for x in range(3):
        dni=int(input("Ingrese el numero de dni:"))
        listamaterias=[]
        continua="s"
        while continua=="s":
            materia=input("Ingrese el nombre de materia que cursa:")
            nota=int(input("Ingrese la nota:"))
            listamaterias.append((materia,nota))
            continua=input("Desea cargar otra materia para dicho alumno
[s/n}:")
        alumnos[dni]=listamaterias
    return alumnos


def listar(alumnos):
    for dni in alumnos:
        print("Dni del alumno",dni)
        print("Materias que cursa y notas")
        for nota,materia in alumnos[dni]:
            print(materia,nota)


def consulta_notas(alumnos):
    dni=int(input("Ingrese el dni a consultar:"))
    if dni in alumnos:
        for nota,materia in alumnos[dni]:
            print(materia,nota)


# bloque principal

alumnos=cargar()
listar(alumnos)
consulta_notas(alumnos)
```