Algoritmos y Programación I
FIUBA

Clase Diccionarios
Python

Pablo Guarna
1er. cuatrimestre 2021

1

Diccionarios - que son

- **DICCIONARIOS,** permiten almacenar otros tipos de datos

Están compuestos de pares de **claves** y **valores**

torneos= {'Racing ': 18,'VELEZ':11, 'ROSARIO':4}

Ideales para guardar datos de personas con teléfonos, stock de productos, cursos con alumnos, usuarios con Mails,etc.

Las **claves son únicas** dentro del diccionario.

```
>>> print(torneos)
{'Racing ': 18, 'VELEZ': 11, 'ROSARIO': 4}

>>> print(torneos['ROSARIO'])
4

>>> type (torneos)
<class 'dict'>
```

2

Diccionarios

Un diccionario es similar a una lista, pero se accede a los valores buscando una ***clave*** *en* lugar de un índice numérico.

Una **clave** puede ser cualquier **cadena** o **número o** cualquiera de tipo **inmutable**.

Los **Valores** pueden ser de **cualquier tipo**

Diccionario_vacio = {}

La longitud len() de un diccionario es el número de pares clave-valor que tiene. Cada par cuenta sólo una vez, incluso si el valor es una lista.

Las **listas** que están indexadas con valores **numéricos.**
Los **diccionarios** se indexan con **Claves.**

El diccionario es un conjunto **NO ordenable** de elementos

3

Operaciones con Diccionarios

- **Guardar un valor con la clave**:

```
>>> torneos ['FERRO']=2
{'Racing': 18, 'VELEZ': 11, 'ROSARIO': 4, 'FERRO': 2}
```

Extraer un valor con la clave:

```
>>> torneos ['VELEZ']
11
```

Borrar un par Clave Valor

```
>>> del torneos ['VELEZ']
{'Racing': 18, 'ROSARIO': 4, 'FERRO': 2}
```

4

Tipos de valores

- **No todos los valores del diccionario tiene que ser del mismo tipo:**

```
>>> torneos['ALMAGRO']='NINGUNO'
{'Racing': 18, 'ROSARIO': 4, 'FERRO': 2, 'ALMAGRO': 'NINGUNO'}

>>> torneos['BOCA']= '?'
 {'Racing': 18, 'ROSARIO': 4, 'FERRO': 2, 'ALMAGRO': 'NINGUNO', 'Boca': '?'}


 torneos[1.5]='ns/nc'
{'Racing': 18, 'ROSARIO': 4, 'FERRO': 2, 'ALMAGRO': 'NINGUNO', 'Boca': '?', 1.5: 'ns/nc'}
```

- En resumen: las **claves** son inmutables números, strings, booleanas
- los valores pueden ser mutables o inmutables

5

Borrado de elementos

- **Elimino todo lo que tenga valores extraños a una tabla de torneos:**

```
>>> del torneos['Boca']
>>> del torneos[1.5]
>>> del torneos['ALMAGRO']
```

Me queda:

```
{'Racing': 18, 'ROSARIO': 4, 'FERRO': 2}

>>> print(len(torneos))
3
```

6

FOR EN DICCIONARIOS

For en diccionarios
Se puede usar el For en un Diccionario para recorrer sus *claves* como en el siguiente caso:

```
torneos = {'Racing : [18], 'ROSARIO': 5, 'FERRO': 2}
for clave in torneos:
      print (clave , ' tiene', torneos[clave] , 'torneos')
```

```
Racing  tiene 18 torneos
ROSARIO  tiene 5 torneos
FERRO  tiene 2 torneos
```

7

Ejercicio literario Simple

Ejercicio
Utilice un for para ir a través del diccionario libros e imprimir todas las autores.

```
libros = {
  " El Aleph.": "Borges" ,
  "Sobre Heroes y Tumbas.":"E.Sabato",
  "La Gesta del Marrano":" M. AGUINIS",
 "Misteriosa Buenos Aires.":  "Mujica Laines",
"Fundacion":"I.Asimov"
}
```

8

Resolución ejerció literario Simple Python

Ejercicio
Utilice un for para ir a través del diccionario libros e imprimir todas las autores.

```
libros = {
  " El Aleph.": "Borges" ,
  "Sobre Heroes y Tumbas.":"E.Sabato",
  "La Gesta del Marrano":" M. AGUINIS",
 "Misteriosa Buenos Aires.":  "Mujica Laines",
"Fundacion":"I.Asimov"
}
for titulo in libros:
        print (libros[titulo])
```

```
I.Asimov
 M. AGUINIS
E.Sabato
Borges
Mujica Laines
```

9

Cambios en los diccionarios

También podría sumar un torneo a ROSARIO
de esta forma estaría reemplazando el valor 4 por el 5

```
>>> torneos['ROSARIO']=5
{'Racing': [18], 'ROSARIO': 5, 'FERRO': 2}
```

mejor seria que le sume uno al valor original que era 4

```
>>> torneos['ROSARIO']=4
>>> torneos['ROSARIO']+=1
>>> torneos
{'Racing': [18], 'ROSARIO': 5, 'FERRO': 2}
```

10

Cambios en los diccionarios

Si quiero incrementar el valor de un diccionario de una clave inexistente:

```
>>> torneos['INDEPENDIENTE']+=1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'INDEPENDIENTE'
```

Por lo cual antes de cambiar un valor de un diccionario, tengo verificar la existencia de la clave

11

Verificar si existe la clave

- **Cambios en los diccionarios:**

```
{'Racing': 18, 'ROSARIO': 4, 'FERRO': 2}

equipo='INDEPENDIENTE'
if equipo  in torneos:
      print('esta')
else:
   print('no esta')
```

```
No esta
```

12

Verificar clave

- **Si tengo un diccionario cuyos valores son listas**

```
copas={'Racing': [18, 3], 'ROSARIO': [4,3], 'FERRO': [2,0]}
```

```
>>> type(copas['ROSARIO'])
<class 'list'>
```

Puedo trabajar con los métodos que ofrecen las listas, por ejemplo:

```
>>> copas['ROSARIO'].remove(3)
>>> print(copas)
{'Racing': [18, 3], 'ROSARIO': [4], 'FERRO': [2, 0]}
>>> copas['ROSARIO']
[4]
```

13

Ejercicio a desarrollar

Programar una función que dada una palabra, calcule la cantidad de veces que esta cada una de las letras que contiene

14

Desarrollo del Ejercicio

Programar una función que dada una palabra, calcule la cantidad de veces que esta cada una de las letras que contiene

```
def cantidaddeletras (palabra):
    cantletras ={}
    for letra in palabra:
        if letra not in cantletras:
            cantletras[letra]=1
        else:
            cantletras[letra]+=1

    for letra in cantletras:
        print ("La letra ", letra , " se encuentra", cantletras[letra] , ' veces')

palabra=input('cargar palabra:')
cantidaddeletras(palabra)
```

15

Salida

cargar palabre:algoritmo
La letra a se encuentra 1 veces
La letra l se encuentra 1 veces
La letra g se encuentra 1 veces
La letra o se encuentra 2 veces
La letra r se encuentra 1 veces
La letra i se encuentra 1 veces
La letra t se encuentra 1 veces
La letra m se encuentra 1 veces

16

Cuestionario

Se tiene un diccionario de alumnos con clave Padrón y en los valores una lista que incluye: Apellido , los códigos de carreras que cursa, y los mails.

alumnos={171717:('Lopez',[9,10],('llopez@fi.uaba.ar','lopecito77@gmail.com'))}

Analizar

1- que se obtiene con alumnos [1], los datos del primer par? ,o que?

2-alumnos['171717'] que información da?

3-como obtengo los datos del primer mail

4- de que tipo es el conjunto de los mails

5- alumnos de [171717][1][0] que muestra

6- como puedo insertar un nuevo código de carrera

17

Cuestionario - Respuestas

Se tiene un diccionario de alumnos con clave Padrón y en los valores Apellido , códigos de carreras que cursa, y una lista de mails.

alumnos={171717:('Lopez',[9,10],('llopez@fi.uaba.ar','lopecito77@gmail.com'))}

1- que se obtiene con alumnos [1], los datos del primer par? o que?

```
>>> alumnos [1]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 1
```

2-alumnos['171717'] que información da?

```
>>> alumnos['171717']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: '171717'
```

18

Cuestionario - Respuestas 3 a 5

Se tiene un diccionario de alumnos con clave Padrón y en los valores Apellido , códigos de carreras que cursa, y una lista de mails.

alumnos={171717:('Lopez',[9,10],('llopez@fi.uaba.ar','lopecito77@gmail.com'))}

3-como obtengo los datos del primer mail

```
>>> alumnos[171717][2][1]
'lopecito77@gmail.com'
```

4- de que tipo es el conjunto de los mails

```
>>> type(alumnos[171717][2])
<class 'tuple'>
```

5- alumnos de [171717][1][0] que muestra

```
>>> alumnos[171717][1][0]
9
```

19

Cuestionario

Se tiene un diccionario de alumnos con clave Padrón y en los valores Apellido , códigos de carreras que cursa, y una lista de mails.

alumnos={171717:('Lopez',[9,10],('llopez@fi.uaba.ar','lopecito77@gmail.com'))}

6- como puedo insertar un nuevo código de carrera,

¿podría hacer lo mismo con los mails¡

```
>>> alumnos[171717][1].append(1)
>>> print (alumnos)
{171717: ('Lopez', [9, 10, 1], ('llopez@fi.uaba.ar', 'lopecito77@gmail.com'))}
```

20

Ejercicio de Stock Inventario Python

- Se tiene cargado en memoria un diccionario llamado Stock con clave Producto y valores cantidad y precio. Se pide calcular el valor total del inventario

stock={1:[2,300],2:[5000,3],5:[60,400]}

A trabajar!!!

21

Resolución Ejercicio de Stock Inventario

```
stock={1:[2,300],2:[5000,3],5:[60,400]}
valor_inventario=0
for codigo in stock:
    valor_inventario+=stock[codigo][0]*stock[codigo][1]
print('el valor total del inventario es ', valor_inventario)
```

el valor total del inventario es 39600

22

Diccionarios Métodos keys()

alumnos={37954018:'Lopez',6338:'Juarez', 98231496:'Uriel'}

Aplicando keys() a un diccionario, devuelve un iterable con todas las claves

```
for clave in alumnos.keys():
    print (clave)
```

```
37954018
6338
98231496
```

```
>>> print(sorted(alumnos.keys()))
[6338, 37954018, 98231496]
```

23

Diccionarios Métodos values()

alumnos={37954018:'Lopez',6338:'Juarez', 98231496:'Uriel'}

Aplicando values() a un diccionario, devuelve una iterable con todos los valores

```
for valor in alumnos.values():
    print (valor)
```

```
Lopez
Juarez
Uriel
```

```
lista=sorted(alumnos.values())
print(lista)
```

```
['Juarez', 'Lopez', 'Uriel']
```

24

Diccionarios Métodos items()

alumnos={37954018:'Lopez',6338:'Juarez', 98231496:'Uriel'}

Aplicando **items()** a un diccionario, devuelve una iterable con tuplas que contiene los pares clave valor

```
for pares in alumnos.items():
    print (pares)
(37954018, 'Lopez')
(6338, 'Juarez')
(98231496, 'Uriel')

lista=sorted(alumnos.items())
print(lista)

[(6338, 'Juarez'), (37954018, 'Lopez'), (98231496, 'Uriel')]
```

25

Diccionarios Otros Métodos

alumnos={37954018:'Lopez',6338:'Juarez', 98231496:'Uriel'}

- **Si quiero extraer y eliminar un elemento del diccionario uso pop**

```
>>> elemento=alumnos.pop(6338)
>>> print(elemento)
Juarez
>>> print(alumnos)
{37954018: 'Lopez', 98231496: 'Uriel'}
```

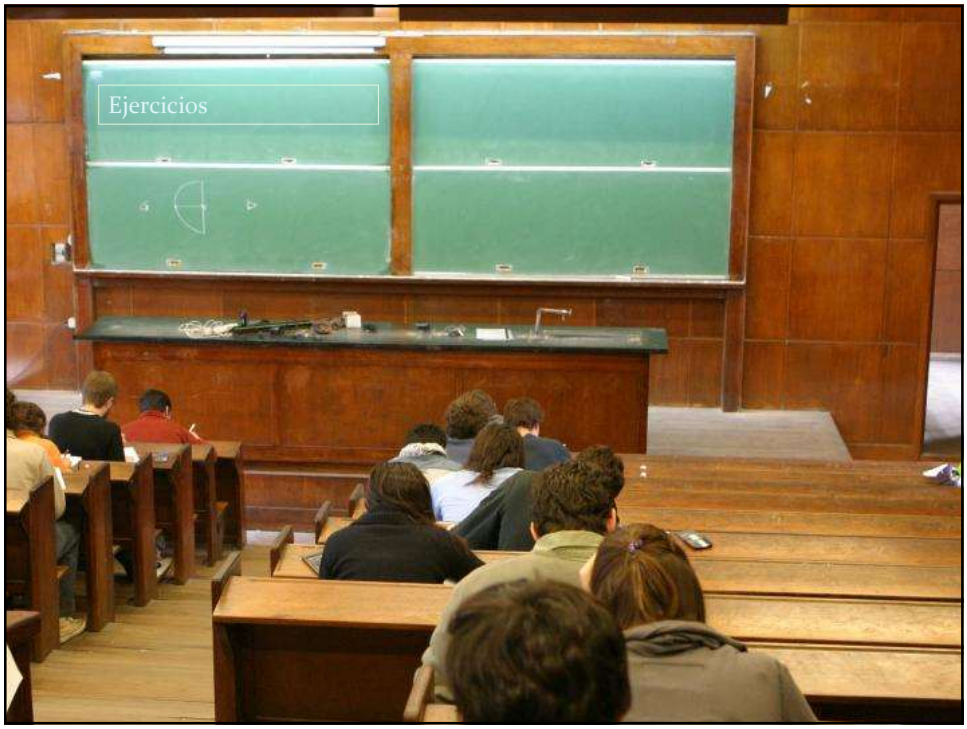
Para buscar un elemento por su clave y si no lo encuentra devuelve un valor por defecto:

```
>>> alumnos.get(98231496,'no existe')
'Uriel'
>>> alumnos.get(6,'no existe')
'no existe'
```

Para limpiar un diccionario uso clear

```
>>> alumnos.clear()
>>> print (alumnos)
{}
```

26



27

Algoritmos y Programación I — Python

Zip toma como argumento dos o más objetos iterables (cada uno de ellos con la misma longitud) y devuelve un nuevo iterable cuyos elementos son tuplas que contienen un elemento de cada uno de los iteradores originales.

Ejercicio
En el siguiente ejemplo el **diccionario precios** y el **diccionario stock** tienen las mismas claves, puede acceder al diccionario stock mientras realiza un for con el diccionario precios.
Cuando está imprimiendo, puede utilizar la sintaxis del ejemplo anterior.

Ejemplo

```
precios = {"banana": 50,"manzana": 28,"nananja": 15,"pera": 30}

stock = {"banana": 60, "manzana ": 0, "nananja ": 32, "pera": 15}
```

28

```
 precios = {"banana": 50,"manzana": 28,"nananja": 15,"pera": 30}
stock = {"banana": 60, "manzana": 0, "nananja": 32, "pera": 15}

for frutap, frutas in zip( precios, stock):
        print( 'La ',frutas, 'vale:',precios[frutap], ' y hay ',stock[frutas])
```

```
La  banana vale: 50  y hay  60
manzana
La  manzana vale: 28  y hay  0
nananja
La  nananja vale: 15  y hay  32
pera
La  pera vale: 30  y hay  15
```

29

Generar un Diccionario de 2 listas

```
>>> producto=['ctacte', 'cajaahorro','prestamo','plazofiijo']
>>> codigo=[1,2,3,4]


>>> dicproductos=dict(zip(producto, codigo))

>>> dicproductos
{'ctacte': 1, 'cajaahorro': 2, 'prestamo': 3, 'plazofiijo': 4}



>>> type(dicpoductos)
<class 'dict'>
```

30

ZIP

Zip toma como argumento dos o más objetos iterables (cada uno de ellos con la misma longitud) y devuelve un nuevo iterable cuyos elementos son tuplas que contienen un elemento de cada uno de los iteradores originales.

Ejemplo Para los siguientes diccionarios calcular para cada fruta el Total de los precios por la cantidad en stock de todas las frutas y el total parcial por fruta

```
precios = {"banana": 50,"manzana": 28,"naranja": 15, "pera": 30}
stock = {  "banana": 60,  "manzana": 00, "naranja": 32, "pera": 15}
total = 0
for frutas in zip (precios, stock):
      total = total + precios[frutas[0]] * stock[frutas[0]]
      print(frutas[0],'valor total', precios[frutas[0]] * stock[frutas[0]])
print ("Monto Final:",total)
```

31

ZIP

Ejemplo Para los siguientes diccionarios calcular para cada fruta el Total de los precios por la cantidad en stock y el total final de cada fruta.

```
precios = {"banana": 50,"manzana": 28,"naranja": 15, "pera": 30}
stock = {  "banana": 60,  "manzana": 00, "naranja": 32, "pera": 15}
total = 0
for frutas in zip (precios, stock):
      total = total + precios[frutas[0]] * stock[frutas[0]]
      print(frutas[0],'valor total', precios[frutas[0]] * stock[frutas[0]])
print ("Monto Final:",total)
```

32

Slida ZIP

banana valor total 3000
manzana valor total 0
naranja valor total 480
pera valor total 450
Monto Final: 3930

33

Ejercicio de Stock Descripciones

- Se tiene cargado en memoria un **diccionario llamado Stock** con clave Producto y valores cantidad y precio. Se pide calcular el valor total por producto con su nombre y el valor total del inventario

Y una lista con las descripciones

descripciones=[(1,'Martillo'),(2,'tornillo'),(5,'mechas')]

stock={1:[2,300],2:[5000,3],5:[60,400]}

34

Ejercicio de Stock Descripciones Resolución

```
dicdesc={}
descripciones=[(1,'Martillo'),(2,'tornillo'),(5,'mechas')]
stock={1:[2,300],2:[5000,3],5:[60,400]}
for cod, desc  in descripciones:
   dicdesc[cod]= desc
 valor_inventario=0
for codigo in  stock:
      valor_prod= stock[codigo][0]*stock[codigo][1]
      valor_inventario+=valor_prod
      print('el valor total del producto ' ,dicdesc[codigo] , ' es ',
valor_prod)
print('el valor total del inventario es ', valor_inventario)
```

35

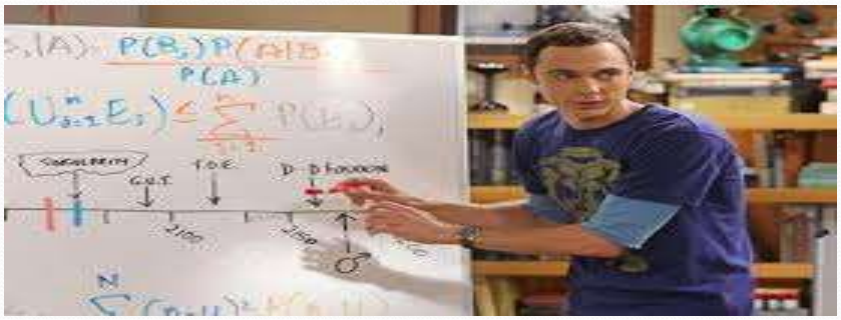
Ejercicio de Stock Descripciones Salida

El valor total del producto Martillo es 600
el valor total del producto tornillo es 15000
el valor total del producto mechas es 24000
el valor total del inventario es 39600

36

Problema

Se pide que ingresen por teclado pares de Equipo-Puntos ganados, el mismo par se puede ingresar varias veces. Se pide generar una Tabla de Puntos Acumulados para cada equipo, ordenando la tabla por puntos en forma decreciente



37

```
def solicitar_valor(mensaje):
    valor = input(mensaje)
    return valor

def CargarTabla():
    nombre = solicitar_valor("cagar equipo:")
    if nombre != 'NO':
        puntos = int(solicitar_valor("cargar puntos:"))
    tabla={}
    while nombre != 'NO':
        if nombre not in tabla:
            tabla[nombre] = puntos
        else:
            tabla[nombre] += puntos
        nombre = solicitar_valor("cagar equipo")
        if nombre != 'NO':
            puntos = int(solicitar_valor("cargar puntos"))
    return tabla

def OrdenarTabla(t):
    listatabla=t.items()
    print(sorted(listatabla, key=lambda l: l[1],reverse=True))

def OrdenarTabla2(t):
    import operator
    tord = sorted(t.items(), key=operator.itemgetter(1), reverse=True)
    print(tord)

t=CargarTabla()
OrdenarTabla(t)
OrdenarTabla2(t)
```

38

Python permite definir **funciones** mínimas, de una solo línea con una sintaxis abreviada

F=lambda argumento: resultado
Ejemplo:

```
b=lambda x, y :  x+y
```

Es igual a

```
def  a(x, y):
        return x + y
```

39

Resumen

Diccionarios {}	**con clave y valor,** **clave única** **clave inmutable** **valor cualquier tipo** **es iterable** **mutable**
Listas []	**secuencia iterable** **indexa por posición** **mutable**
Listas (,)	**secuencia iterable** **indexa por posición** **inmutable**

40