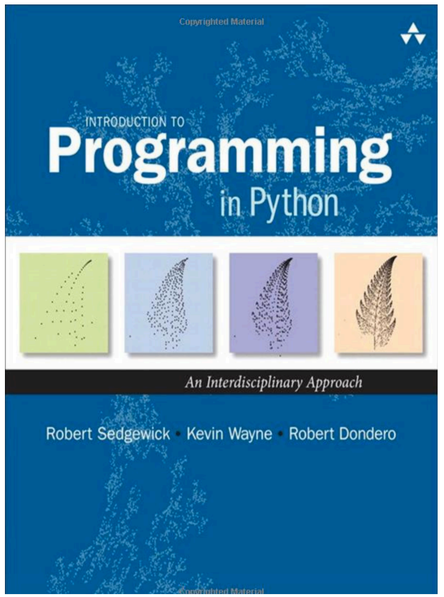
Taller de Programación

Diccionarios, APIs y Módulos

Leonardo Causa Morales
l.causa@udd.cl



Basada en presentaciones oficiales de libro Introduction to Programming in Python (Sedgewick, Wayne, Dondero).

Disponible en https://introcs.cs.princeton.edu/python

Material preparado por la Dra. Daniela Opitz

Outline

- Casos de uso para dict()
- Manejo de archivos con diccionarios
- Módulos

Ejemplo: ¿Cómo almacenar las notas de un curso?

- Podríamos usar una lista para nombre de alumno, notas y el curso:

```
nombres = ['Diego', 'Francisca', 'Loreto', 'Leo']
notas = [4.1, 5.5, 6.8, 3.9]
```

- Cada lista contiene información distinta.

- Las listas deben ser del **mismo tamaño**.

- Información entre las listas deben estar en la misma **posición**.

Extraído desde slides prof. Eric Grimson https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec6.pdf

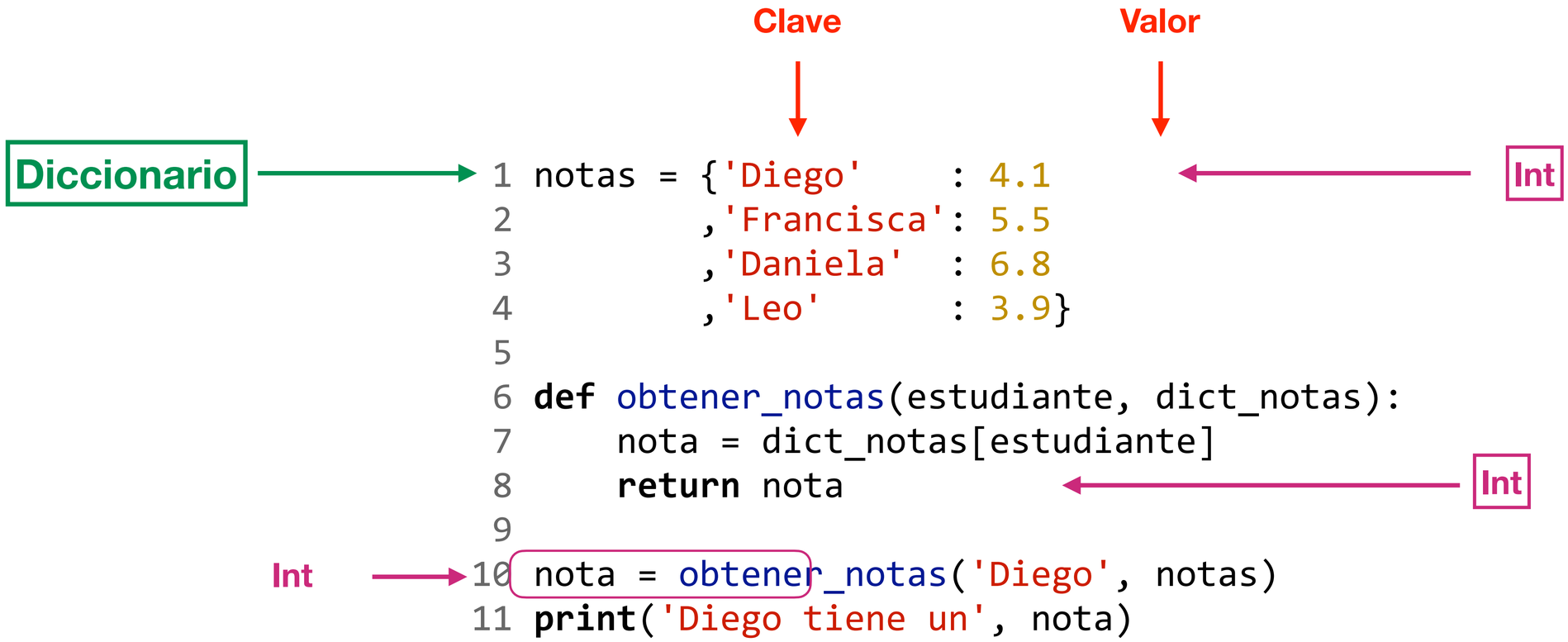
Ejemplo: ¿Cómo almacenar las notas de un curso?

Nota de Diego?

```
def obtener_notas(estudiante, lista_nombres, lista_notas):
    i = lista_nombres.index(estudiante)
    nota = lista_notas[i]

    return nota

nota = obtener_notas('Diego', nombres, notas)

print('Diego tiene un', nota)
```

- Complicado si tienes varios tipos de información que almacenar
- Debes mantener **varias listas**, y pasarlas como argumento
- Necesita de un í**ndice** (un **entero** con la posición)
- **DIFICIL DE MANTENER!**

Ejemplo: ¿Cómo almacenar las notas de un curso?



Patrón típico de uso diccionario

- Recorrer todas las llaves del diccionario:

```
for llave in notas.keys():
    print(llave)
```

- Recorrer los valores del diccionario:

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

- Recorrer todas las llaves y valores:

```
for llave, valor in notas.items():
    print(llave, valor)
```

```
notas = {'Diego'    : 4.1
        ,'Francisca': 5.5
        ,'Daniela'  : 6.8
        ,'Leo'      : 3.9}
```

Actividad

El siguiente diccionario contiene información relacionada a la pobreza regional de nuestro país. Las llaves (keys) corresponden a los nombres las regiones de Chile, y los valores (values) corresponden a la información del porcentaje de pobreza de ingreso en la respectiva región (Fuente: Casen 2015). Imprima el nombre la región con mayor población en situación de pobreza de ingresos y su respectivo porcentaje.

```
pobreza_reg={'Tarapaca' : 7.1, 'Antofagasta' : 5.4, 'Atacama' : 6.9, 'Coquimbo' :
13.8, 'Valparaíso' : 12.0, 'Libertador Bernardo OHiggins' : 13.7, 'Maule' : 18.7,
'Biobío' : 17.6, 'La Araucanía' : 23.6, 'Los Lagos' : 16.1, 'Aysen' : 6.5,
'Magallanes y La Antártica Chilena' : 4.4, 'Región Metropolitana de Santiago' : 7.1,
'Los Ríos' : 16.8, 'Arica y Parinacota' : 9.7}
```

API: Interfaz de programación de aplicaciones

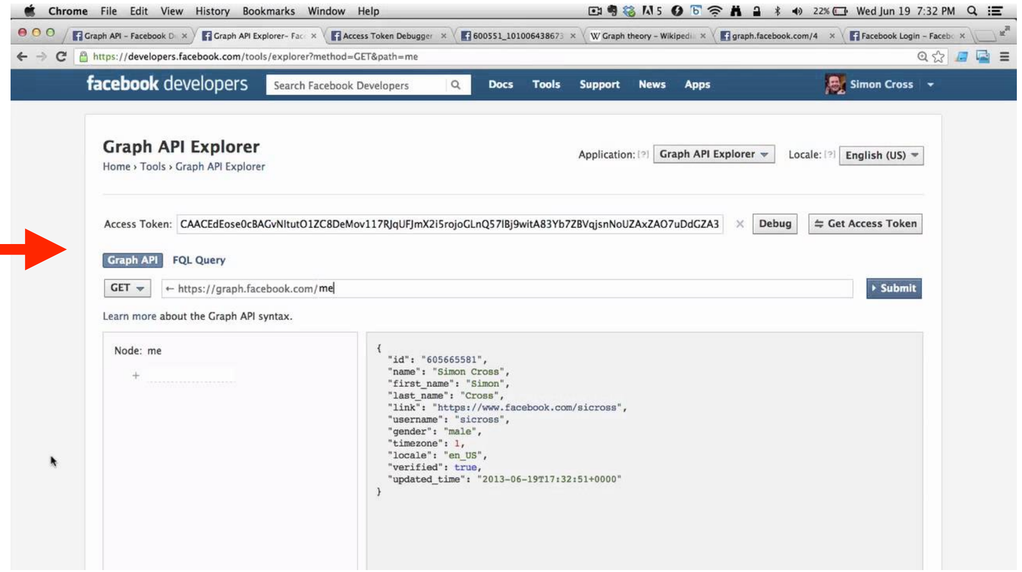
- API: conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software.

- Características:

  - Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios .
  - Uno de los principales es proporcionar un conjunto de funciones de uso general para evitar programar todo desde el principio.

Facebook Api Graph

Expone ciertas funcionalidades de Fb para que otros puedan usarla



Facebook

https://developers.facebook.com/docs/graph-api/

Api Grap

Módulos

- Hasta ahora todos nuestros programas han consistido de un simple archivo.py
- Si nuestro código es muy extenso, es complicado mantener actualizado el archivo.py
- Para solucionar lo anterior podemos definir y ejecutar funciones en otros archivos
  - Permite reutilizar código: un programa puede usar código que ya ha sido escrito, sin necesariamente copiar-pegar
  - Permite hacer programación **modular**: construir programas componiendo código de diversas fuentes (otra vez, sin copiar y pegar)

Modulo Random

```
import random

regalos = ['sartén', 'jamón', 'mp4', 'muñeca', 'tv',
           'patín', 'balón', 'reloj', 'bicicleta', 'anillo']

for sorteo in range(5):
    regalo = regalos[random.randint(0, 9)]
    print('Sorteo', sorteo + 1, ':', regalo)
```

Fuente: https://python-para-impacientes.blogspot.com/2015/09/el-modulo-random.html

Modulos

```
# Módulo de números Fibonacci

def fib(n):    # Escribe la serie Fibonacci hasta n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b



def fib2(n):   # Devuelve la serie Fibonacci hasta n
    resultado = []
    a, b = 0, 1
    while b < n:
        resultado.append(b)
        a, b = b, a+b
    return resultado
```

```
1  import fibo
```

```
1  fibo.fib(100)
```

```
1 1 2 3 5 8 13 21 34 55 89
```

```
1  fibo.fib2(100)
2
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

```
1  from fibo import fib, fib2
```

```
1  fib(100)
```

```
1 1 2 3 5 8 13 21 34 55 89
```

```
1  fib2(100)
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

```
1  from fibo import *
```

```
1  fib(100)
2
```

```
1 1 2 3 5 8 13 21 34 55 89
```

```
1  fib2(100)
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Ejecutando modelos como scripts

```
# Módulo de números Fibonacci

def fib(n):    # Escribe la serie Fibonacci hasta n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b



def fib2(n):   # Devuelve la serie Fibonacci hasta n
    resultado = []
    a, b = 0, 1
    while b < n:
        resultado.append(b)
        a, b = b, a+b
    return resultado

if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

```
[Daniela:~] daniela% cd
[Daniela:~] daniela% cd Desktop/Clase14/
[Daniela:~/Desktop/Clase14] daniela% ls
fibo.py         fibo_2.py
[Daniela:~/Desktop/Clase14] daniela% python3 fibo_2.py 100
1 1 2 3 5 8 13 21 34 55 89 [Daniela:~/Desktop/Clase14] daniela%
```

Hace que el código pueda ser ejecutado como script o como módulo importable

Modulos

matriz.py

```
def crear(m,n):
    #Retorna una matriz de ceros con m filas por n columnas
    matriz = []
    for i in range(m):
        matriz.append([0]*n)
    return matriz

def imprimir(matriz):
    #Imprime cada fila de una matriz
    for fila in matriz:
        for elem in fila:
            print(elem, end=' ')
        print()

def asignar(matriz, i,j,v):
    matriz[i][j] = v

def main():
    print(crear(10,20))

if __name__ == '__main__':
    main()
```

importa el código en matriz.py

cliente.py

```
import matriz

m = matriz.crear(3,3)
matriz.asignar(m, 0, 1, 9)
matriz.asignar(m, 2, 2, 3)
matriz.asignar(m, 1, 2, 1)
matriz.imprimir(m)
```