Introducción Python 3

Autor: Esp. Ing. Ernesto Gigliotti. UTN-FRA

Instalación pip

```
curl https://bootstrap.pypa.io/get-pip.py -o
get-pip.py

sudo -H python3 get-pip.py

pip --version
```

Instalación Visual Studio Code

Bajar .deb de https://code.visualstudio.com

```
sudo dpkg -i code_1.33.1-1554971066_amd64.deb
```

Instalar extensión “Python”

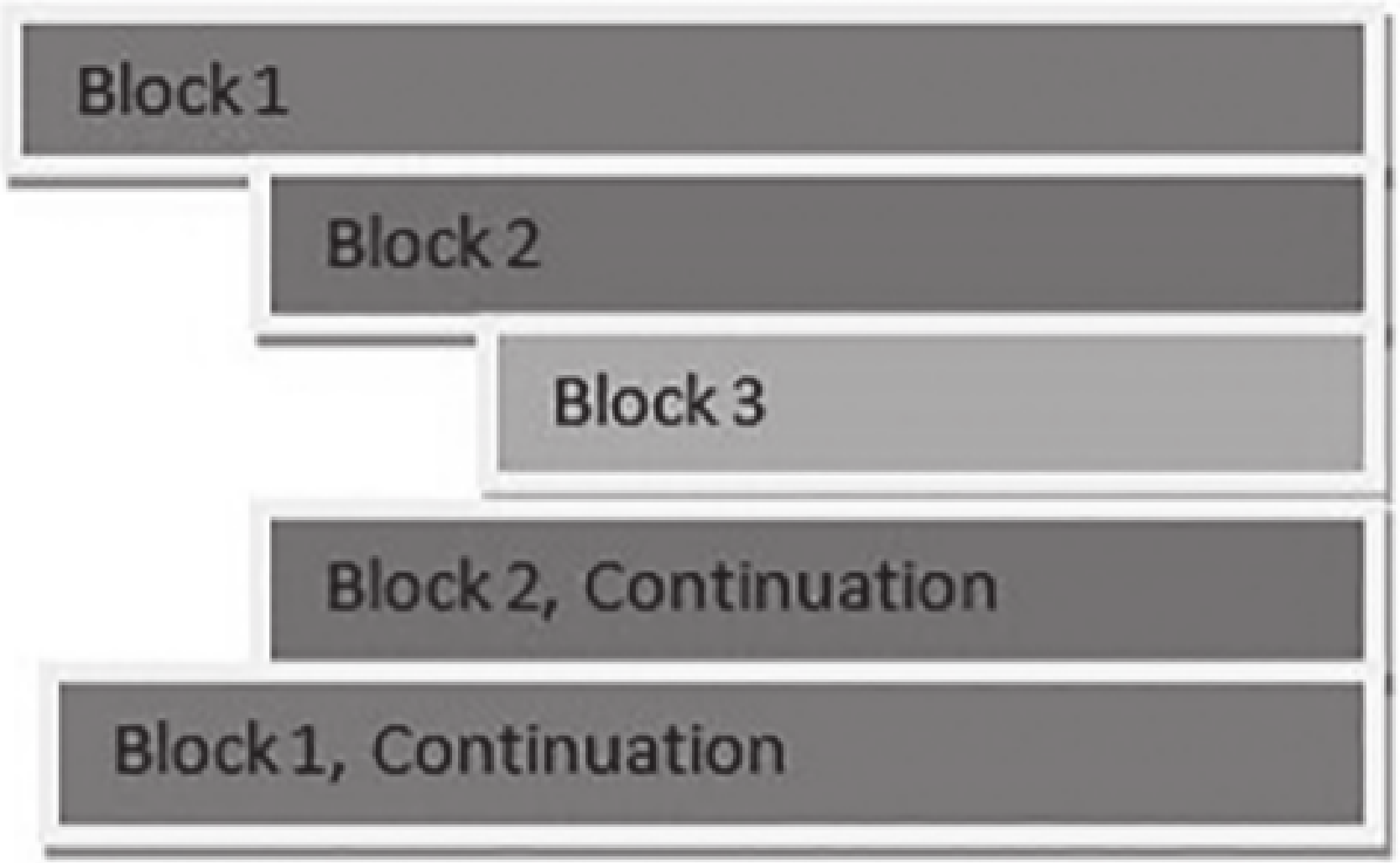
Python

- Intérprete

- Tipado dinámico
  - El tipo se define en tiempo de ejecución

- Fuertemente tipado
  - Durante las operaciones se chequea el tipo

Indentation



```
mi_variable = 27
mi_flag = True

if mi_flag:

    while mi_variable > 0:
        print(mi_variable)
        mi_variable-=1
```

Guía de estilos:
https://www.python.org/dev/peps/pep-0008/

```
mi_variable = 27
mi_flag = True

if mi_flag:

    while mi_variable > 0:
        print(mi_variable)
        mi_variable-=1
```

Números

- **Enteros: Con signo, sin límite**
- **Punto flotante**
- **Complejos**

```
nint = 27
nfloat = 3.14

print(type(nint))
print(type(nfloat))

>> <class 'int'>
>> <class 'float'>
```

Booleans

```
flag1 = True
flag2 = False

print(type(flag1))
print(flag2)

if flag1:
  print("verdadero!")

>> <class 'bool'>
>> False
>> verdadero!
```

IF/ELSE/SWITCH

```
if (flag1 and flag2 or flag3) and not flag4:
    …
elif flag5:
    ...
elif flag6:
    ...
else:
    ...
```

No usar **& | ^ ~ << >>**
se usan para operaciones binarias (como en c)

Cadenas (strings)

```
msg = "Hola mundo"
msg = 'Hola mundo'

print(msg)
print(msg[2])
print(msg[5:10]) // Slice Notation

>> Hola mundo
>> l
>> mundo
```

Cadenas: Operaciones

```
msg = "Hola"
msg2 = " mundo"
print(msg+msg2)
print(msg*2)
print("Hola" in msg)

>> Hola mundo
>> HolaHola
>> True
```

Cadenas: Formatting

```
age=32
name="Ernesto"

msg = "Hola, {1}. Tenes {0}.".format(age,
                                     name)

msg = "Hola, {1}. Tenes {0:03d}.".format(age,
                                     name)

print(msg)
>> Hola, Ernesto. Tenes 32.
>> Hola, Ernesto. Tenes 032.
```

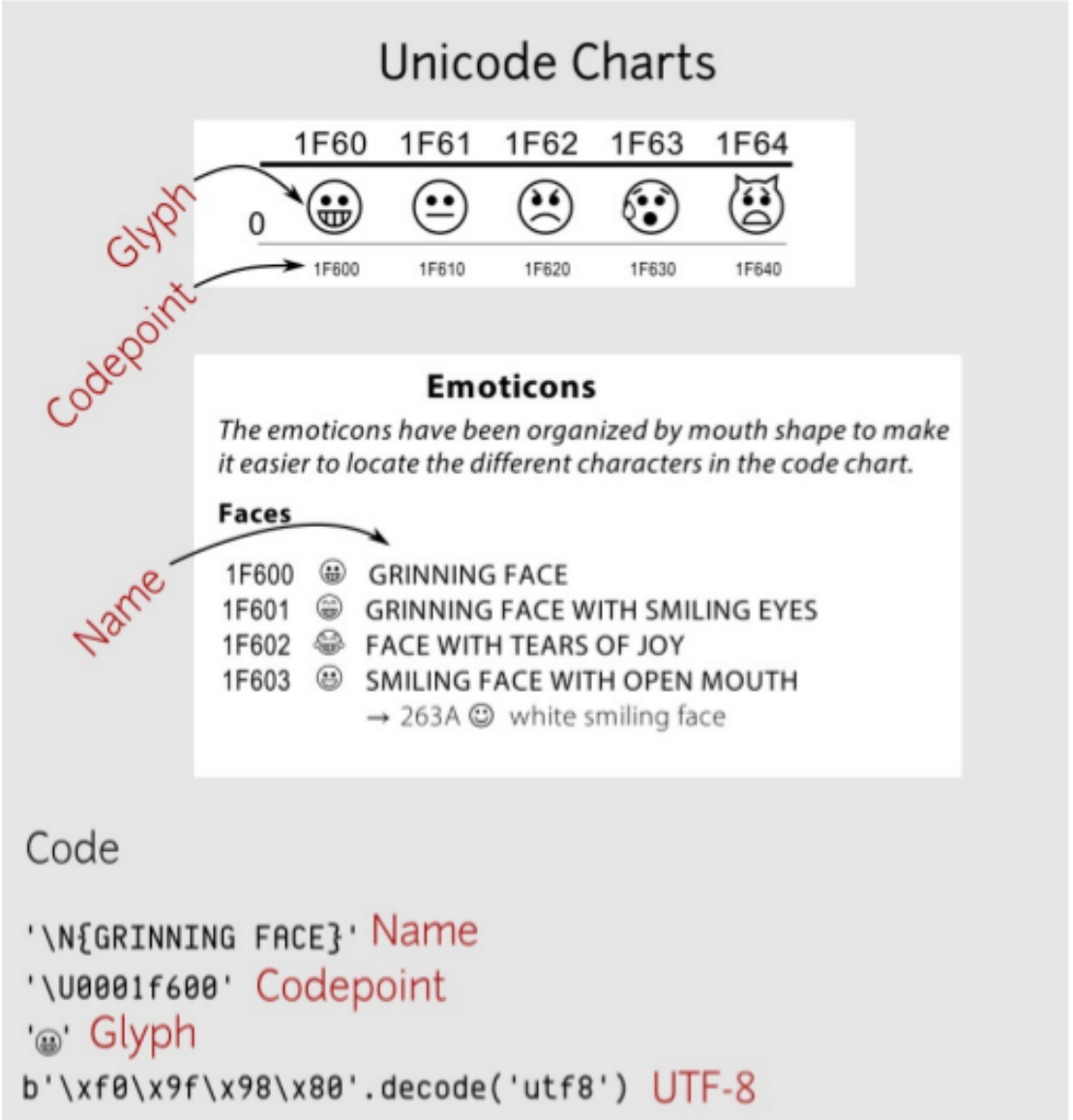
Cadenas: Formatting

```
age=32
name="Ernesto"

msg = "Hola, %s. Tenes %d." % (name,age)


print(msg)


>> Hola, Ernesto. Tenes 32.
```

bytearray

```
b = bytearray()
b.append(0x02)
b.append(0x10)
b.append(0x05)
b.append(0x10)
b.append(0x03)

print(b)
print(b[3])
print(len(b))
>> bytearray(b'\x02\x10\x05\x10\x03')
>> 16
>> 5
```

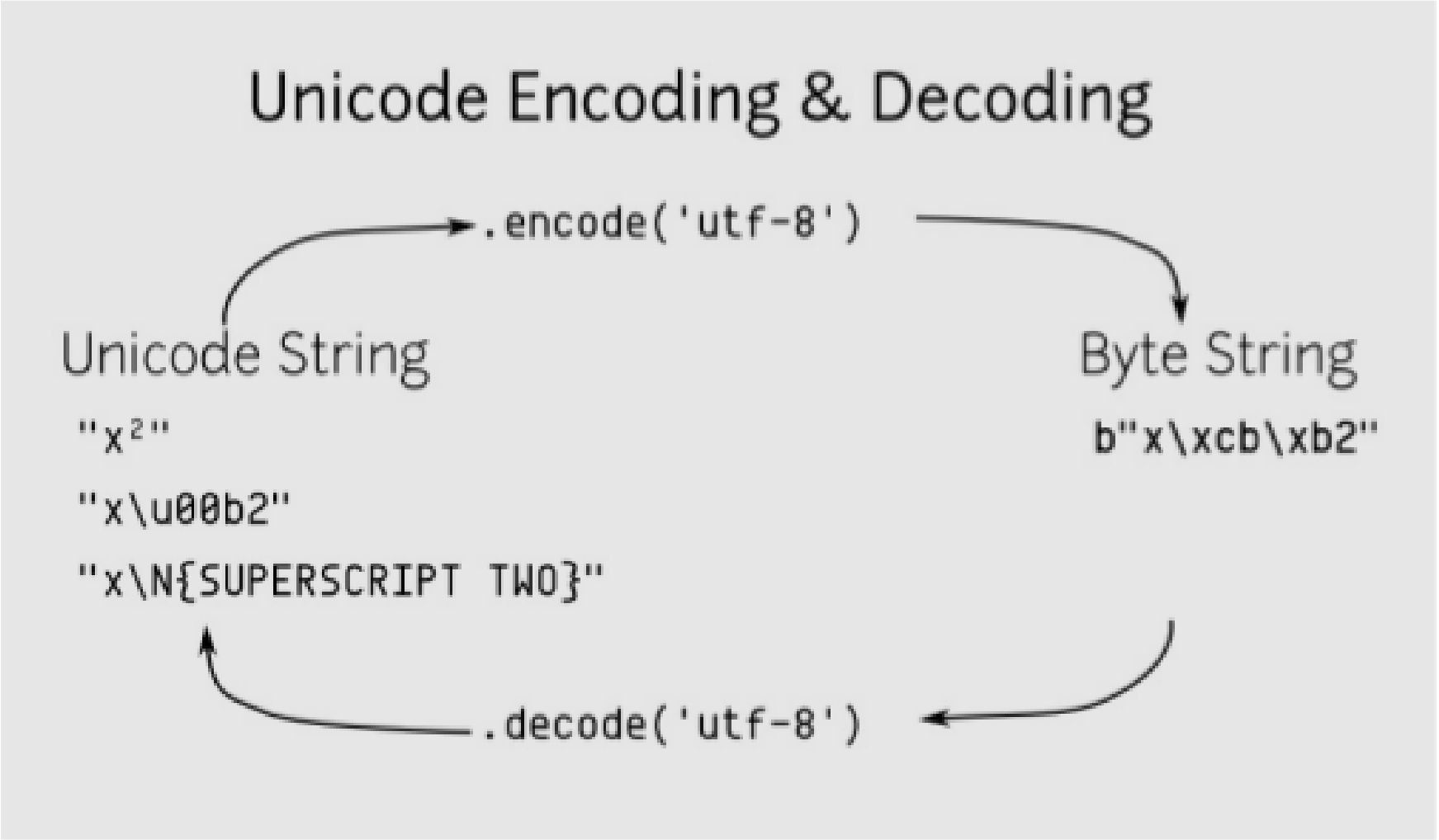
Unicode, encoding y decoding



Unicode, encoding y decoding

- En python 3 los strings se almacenan con su código UNICODE.
- El **encoding** nos permite codificar los códigos unicode para que no ocupen tanto espacio. Obtenemos un array de bytes.
  - UTF-8
  - UTF-16
  - 1252
- El **decoding** nos permite tomar un array de bytes codificados y volver a obtener los códigos unicode. Obtenemos un string de python 3.

Unicode, encoding y decoding



Conversiones

```
s="\u00D1" # unicode de la Ñ
print(s)
>>> Ñ

ba = s.encode("utf-8") #pasamos a bytes
print(ba)
>>> b'\xc3\x91' #0xC3 0x91 es la Ñ en UTF-8

s2 = ba.decode("utf-8") #pasamos a string
print(s2)
>>> Ñ
```

Conversiones

```
i=int("27")
f=float("3.14")

s = str(27)
s2 = str(3.14)


data = bytearray.fromhex("FFAA0B")
print(data[0]) # 255

data = bytearray("0123",encoding="utf-8")
print(data[0]) #48=0x30='0'
```

Leer datos de terminal:

```
nom = input("ingrese su nombre:")

print(nom)
print(type(nom)) # string
```

Listas

```
l = [1, 2, 3, 4, 5 ];
print(l)

for elemento in l:
    print(elemento)
```

```
>> [1, 2, 3, 4, 5]
>> 1
>> 2
>> 3
>> 4
>> 5
```

Listas

```
l = [1, 2, 3, 4, 5 ];
print(l[2])
print(l[-1])
print(l[1:3])
>> 3
>> 5
>> [2, 3]

l.append(6)
l.remove(2) #por valor
del l[“2”] #por posición

print(l)
>> [1, 3, 4, 5, 6]
```

Función len()

```
lista = [1, 2, 3, 4, 5 ];
cantidad_elementos = len(lista)

print(cantidad_elementos)

>> 5
```

Tuplas : Listas inmutables

```
l = (1, 2, 3, 4, 5);
print(l[2])
print(l[-1])
print(l[1:3])
>> 3
>> 5
>> (2, 3)

l.append(6) #ERROR no hay append
l.remove(2) #ERROR no hay remove
```

Variables/Objetos mutables e inmutables

- Cada variable tiene una propiedad “id”, que es la dirección de memoria, la obtenemos: **`id(var)`**

- **Mutable:** Puede ser cambiado luego de ser creado.
  - Listas
  - Diccionarios
  - Objetos
  - bytearray
- **Inmutable:** No puede ser cambiado luego de ser creado.
  - Números
  - Strings
  - Tuplas

Diccionarios

```
d = {"color":"red","state":True,"id":27}
print(d)
print(d["state"])
d["key"]="value"
print(d)
```

```
>> {'state': True, 'color': 'red', 'id': 27}

>> True

>> {'state': True, 'color': 'red', 'key':
                                'value', 'id': 27}
```

Iteraciones

```
for elem in lista:
  print(elem)

for elem in tupla:
  print(elem)

for i in range(0,10): # de 0 a 9
  print(i)

for k in dic:
  print(dic[k])
```

Funciones

```
def mi_funcion(arg1,arg2,arg3=1):
   print(arg1)
   print(arg2)
   print(arg3)
   return 5

r = mi_funcion(1,2)

mi_funcion("hola",2,3)
```

Funciones

```
def mi_funcion(arg1,arg2,arg3=1):
    """ documentacion
        de la funcion (docstring)
    """
    print(arg1)
    print(arg2)
    print(arg3)
    return 5

r = mi_funcion(1,2)

mi_funcion("hola",2,3)
```

Importando bibliotecas

Módulo: Son archivos .py. Dentro puede haber variables, funciones o clases.

```
from math import sin, pi

sin(pi/2)

>>1.0
```

Importando bibliotecas

```
import math

math.sin(math.pi/2)

>>1.0
```

Creación de módulos

```
saludos.py  #minuscula. Modulo “saludos”

def bienvenida():
        print("Bienvenido")

def adios():
        print("Chau")
```

Creación de módulos

Lo usamos:

```
import saludos

saludos.adios()

>>> Chau
```

Packages

Package: Es un directorio. Dentro debe tener un archivo llamado “__init__.py”

```
hablar/
--__init__.py
--esp/
----__init__.py
----saludos.py
```

Packages

Uso

```
from hablar.esp import saludos

saludos.adios()

>>> Chau
```

Ejecución de archivos: Con comando python3

```
>>> python3 archivo.py
```

Ejecución de archivos: Con archivo ejecutable

```
>>> chmod +x archivoExec.py
>>> ./archivoExec.py
```

En archivoExec.py:

```
#!/usr/bin/python3

print("hola mundo")
```

Bibliografía

Gowrishankar S. Veena A. (2019). Introduction to Python Programming. NW. Taylor & Francis Group, LLC.

Matt Harrison. Illustrated guide to python 3. (2017). Treading on Python Series

https://pip.pypa.io/en/stable/installing/

https://www.learnpython.org/es/

https://docs.python.org/3/library/stdtypes.html