

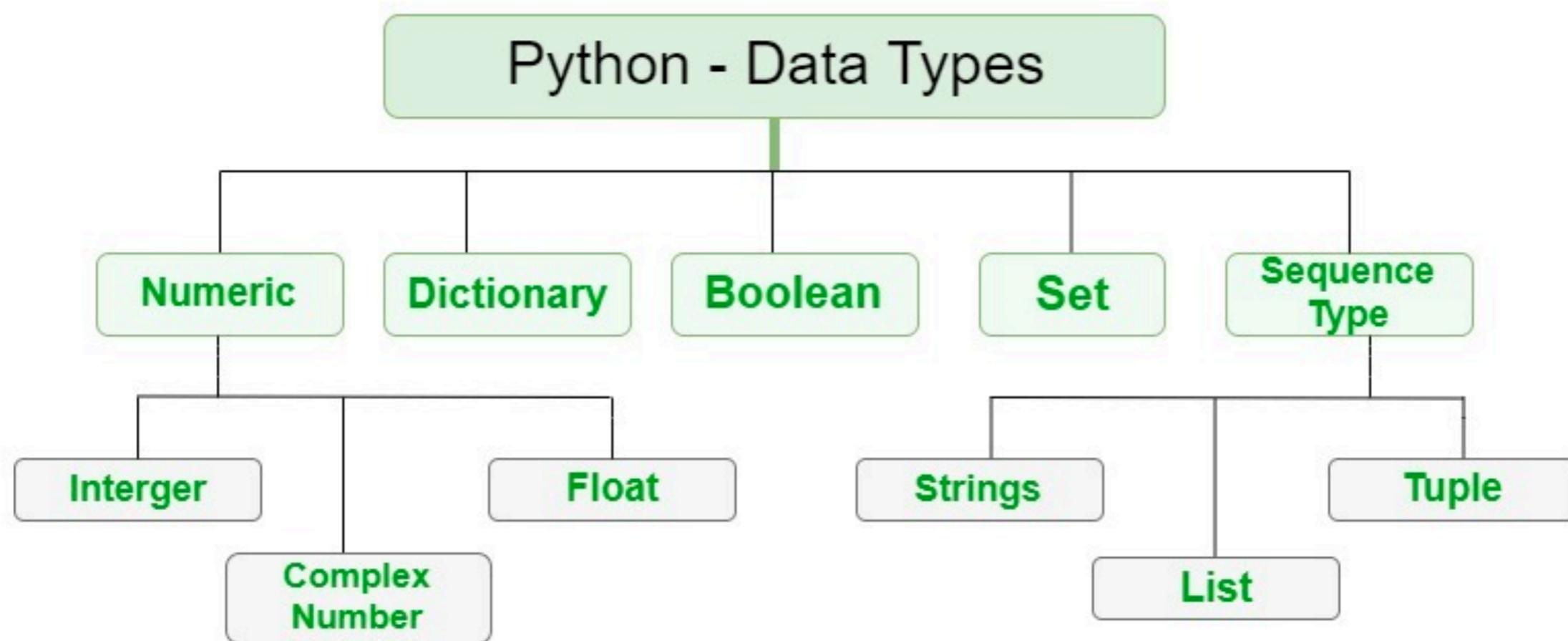
Fundamentos de programación II

Alex Di Genova

16/04/2024

¿Qué frases entienden los computadores?

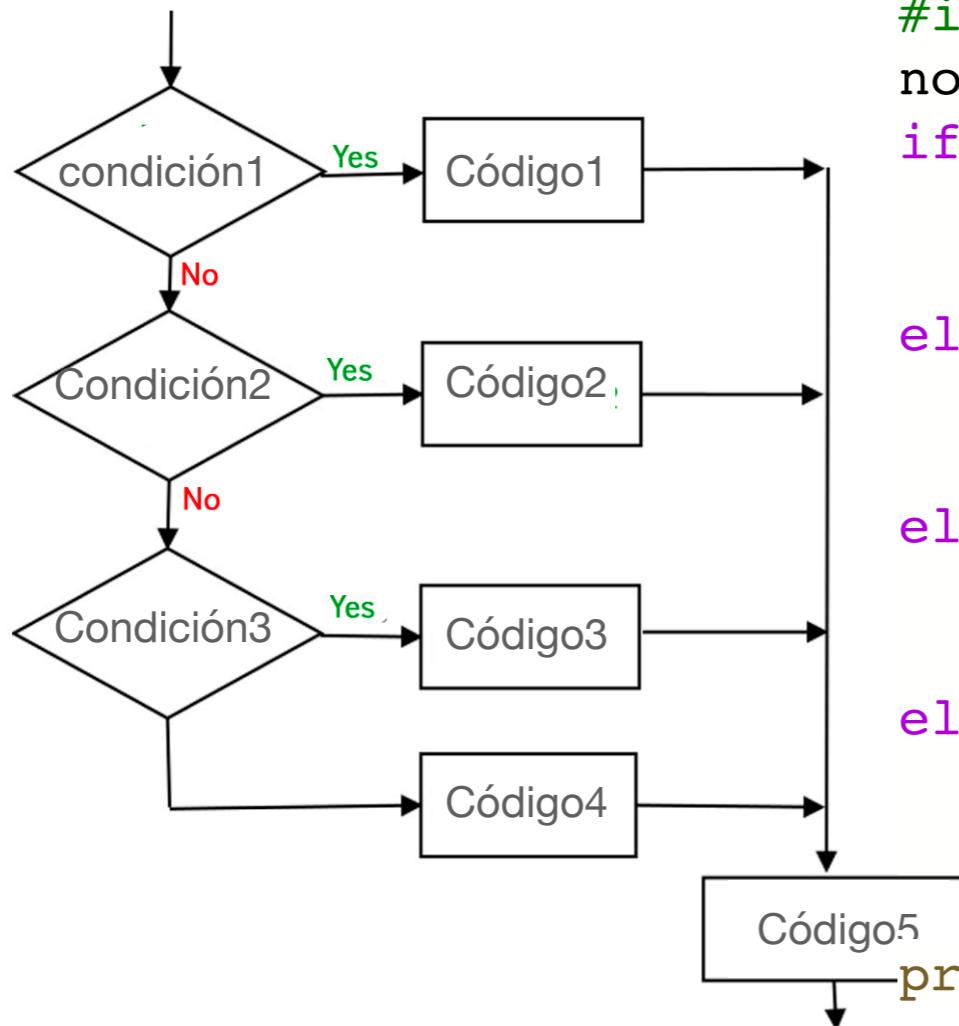
Explicación del concepto variable.



A mi computador le gusta tomar decisiones

Presentación de instrucciones de decisión (if, elif, else) en Python

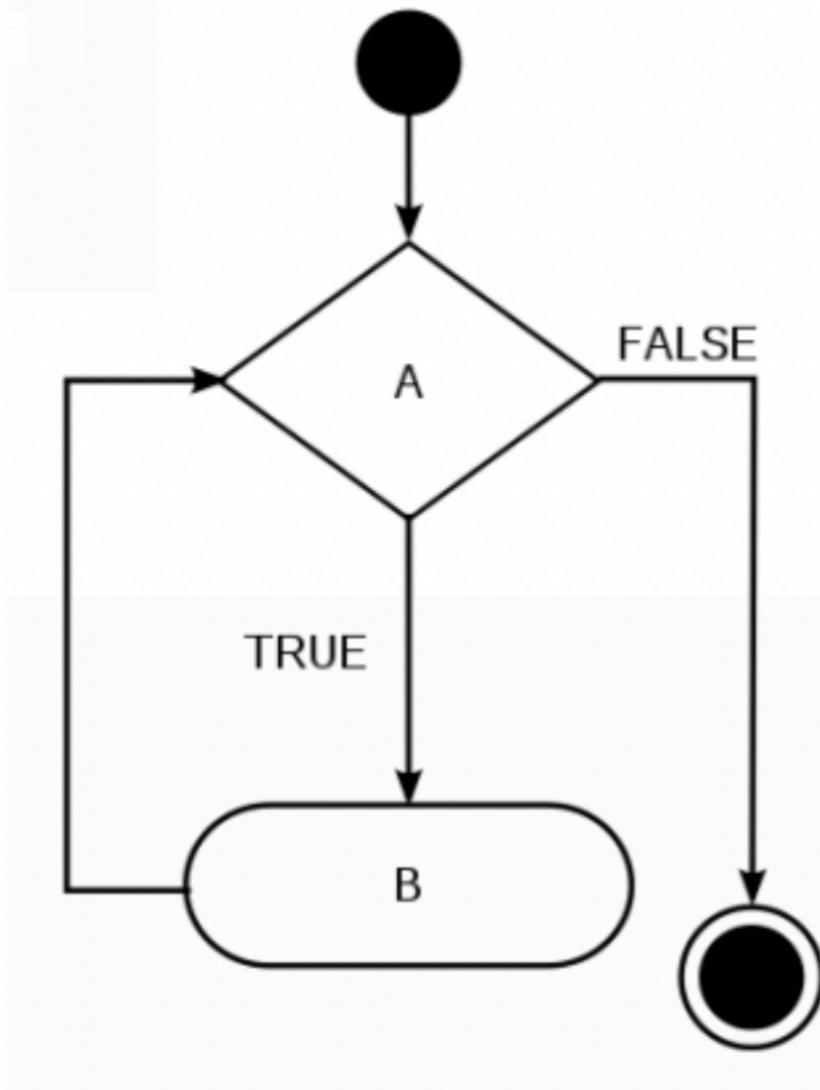
IF-elif-else



```
#if-elif-else
nombre= "Alex"
if(nombre == "Alex"):
    print("Hola Alex")
    print("Se que te gusta jugar al futbol")
elif(nombre == "Roberto"):
    print("Hola Roberto")
    print("Me contaron que te gusta leer libros")
elif(nombre == "Camila"):
    print("Hola Camila")
    print("Se que te gusta programar")
else:
    print("Hola ",nombre)
    print("No se nada de ti")
print("Bienvenido")
```

Mi computador es el mejor para repetir instrucciones.

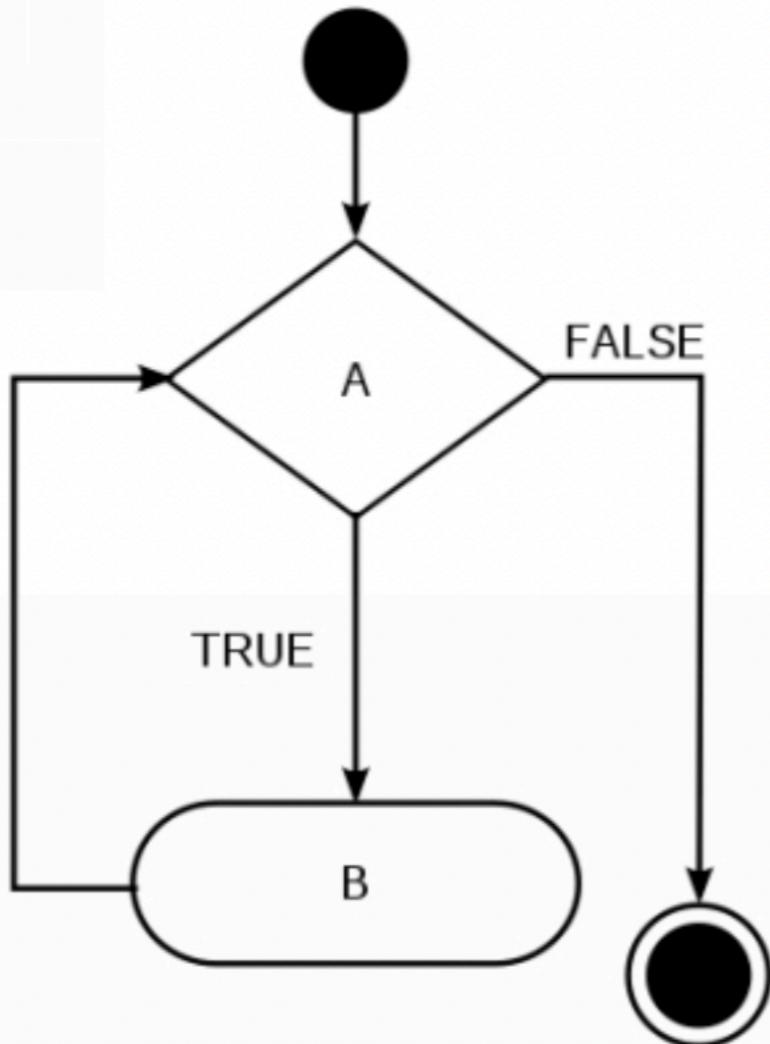
Presentación de instrucción de iteración “While” en Python



```
while 2 > 1:  
    print("Hola --- no puedo parar...")  
  
i = 1  
while i < 5:  
    print(i)  
  
i = 1  
while i < 5:  
    print(i)  
    i=i+1  
  
i = 1  
while i > 5:  
    print(i)  
    i=i+1
```

Mi computador es el mejor para repetir instrucciones.

Presentación de instrucción de iteración “For” en Python



```
for i in range(1,10):  
    print(i)
```

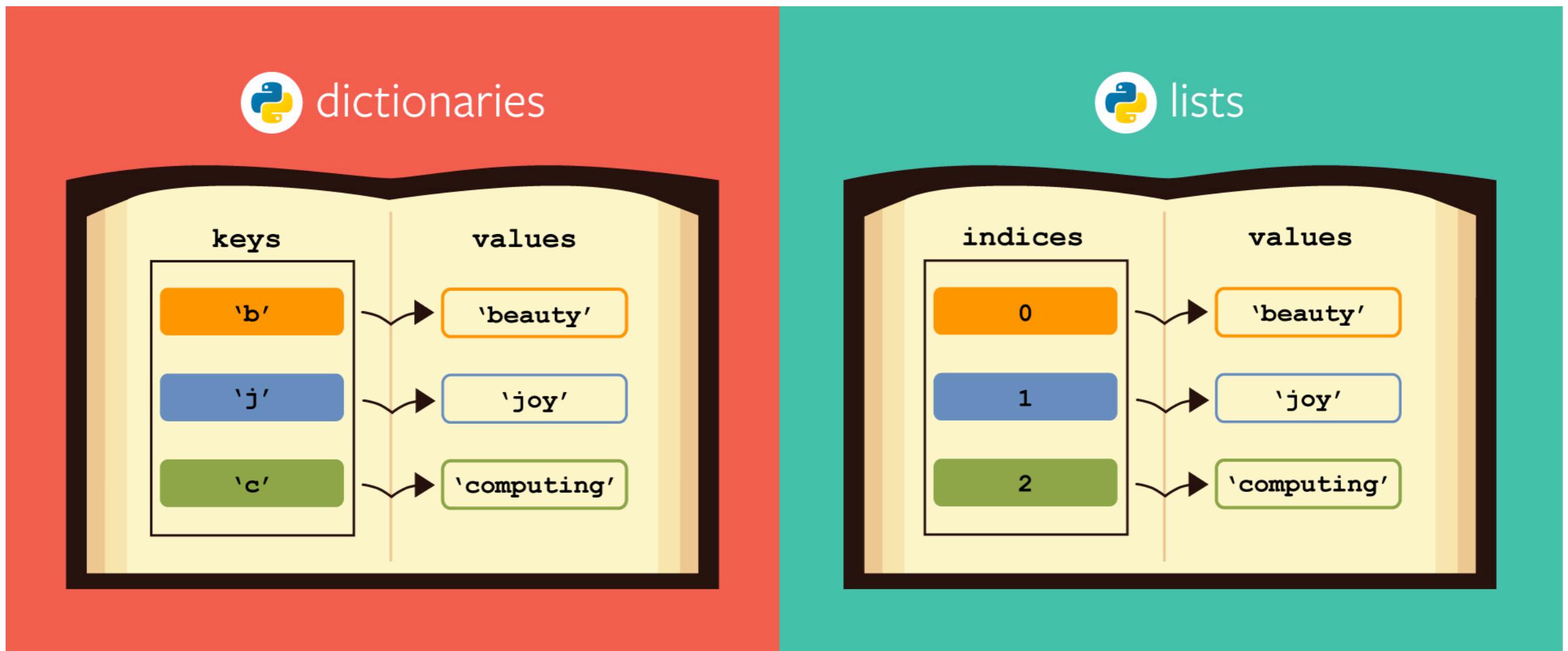
```
S="palabra"  
for i in S:  
    print(i)
```

```
for i in S:  
    for j in range(10):  
        print(i,j)
```

Mi computador sabe cómo usar un diccionario.

Presentación de diccionarios “hash” en python.

Qué es un diccionario?



Mi computador sabe cómo usar un diccionario.

Presentación de diccionarios “hash” en python.

- En Python, un diccionario es un tipo especial de lista.
- Clave: el nombre
- Valor: La cantidad o tipo de algo.

Keys
Values

```
dictionary = { "a" : 1, "b" : 2 }
```



• Diccionarios vs listas

```
[27] 1 L=list()
2 n=10000000
3 # creamos una lista con 10M numeros
4 for i in range(n):
5 | L.append(i)
6 # creamos un diccionario con 10M numeros
7 D=dict()
8 for i in range(n):
9 | D[i]=str(i)
```

```
[43] 1 import time
2 start = time.time()
3 f=0
4 for i in range(100):
5 | if random.randint(0,n) in L:
6 | | f=f+1
7 end = time.time()
8 print("Lista",end - start)
```

Lista 7.328480243682861

```
1 import time
2 start = time.time()
3 f=0
4 for i in range(100):
5 | if random.randint(0,n) in D.keys():
6 | | f=f+1
7 end = time.time()
8 print("Diccionario",end - start)
```

Diccionario 0.0005049705505371094

Mi computador sabe cómo usar un diccionario.

Presentación de diccionarios “hash” en python.

List



Guardar

Buscar



Mi computador sabe cómo usar un diccionario.

Operaciones con diccionarios (agregar, buscar, borrar) en Python.

- clear
- copy
- fromkeys
- get
- items
- keys
- pop
- popitem
- setdefault
- update
- values

```
1 #crear diccionario
2 D=dict()
3 NP={'carlos':1234,'roberto':6655,'camila':1414,'andrea':1212}
4 #acceder por clave
5 print(NP['andrea'])
6 #borra el diccionario
7 NP.clear()
8 NP={'carlos':1234,'roberto':6655,'camila':1414,'andrea':1212}
9 #copiar un diccionario
10 D=NP.copy()
11 print(D)
12 #obtener el valor por clave
13 print(NP.get("roberto"))
14 #items
15 print(NP.items())
16 #claves
17 print(NP.keys())
18 #valores
19 print(NP.values())
20 #eliminar valores por clave
21 a = NP.pop("camila")
22 print(NP, a)

→ 1212
{'carlos': 1234, 'roberto': 6655, 'camila': 1414, 'andrea': 1212}
6655
dict_items([('carlos', 1234), ('roberto', 6655), ('camila', 1414), ('andrea', 1212)])
dict_keys(['carlos', 'roberto', 'camila', 'andrea'])
dict_values([1234, 6655, 1414, 1212])
{'carlos': 1234, 'roberto': 6655, 'andrea': 1212} 1414
```

Mi computador sabe cómo usar un diccionario.

Operaciones con diccionarios (agregar, buscar, borrar) en Python.

```
# crear bd usuarios
NP={'carlos':1234,'roberto':6655,'camila':1414,'andrea':1212}
# obtener usuario y password
nombre=input("ingrese nombre:")
password=int(input("ingrese pass"))
if nombre in NP.keys():
    if NP[nombre] == password:
        print("Welcome",nombre)
    else:
        print("Password incorrecto")
else:
    print("Usuario no registrado")
```

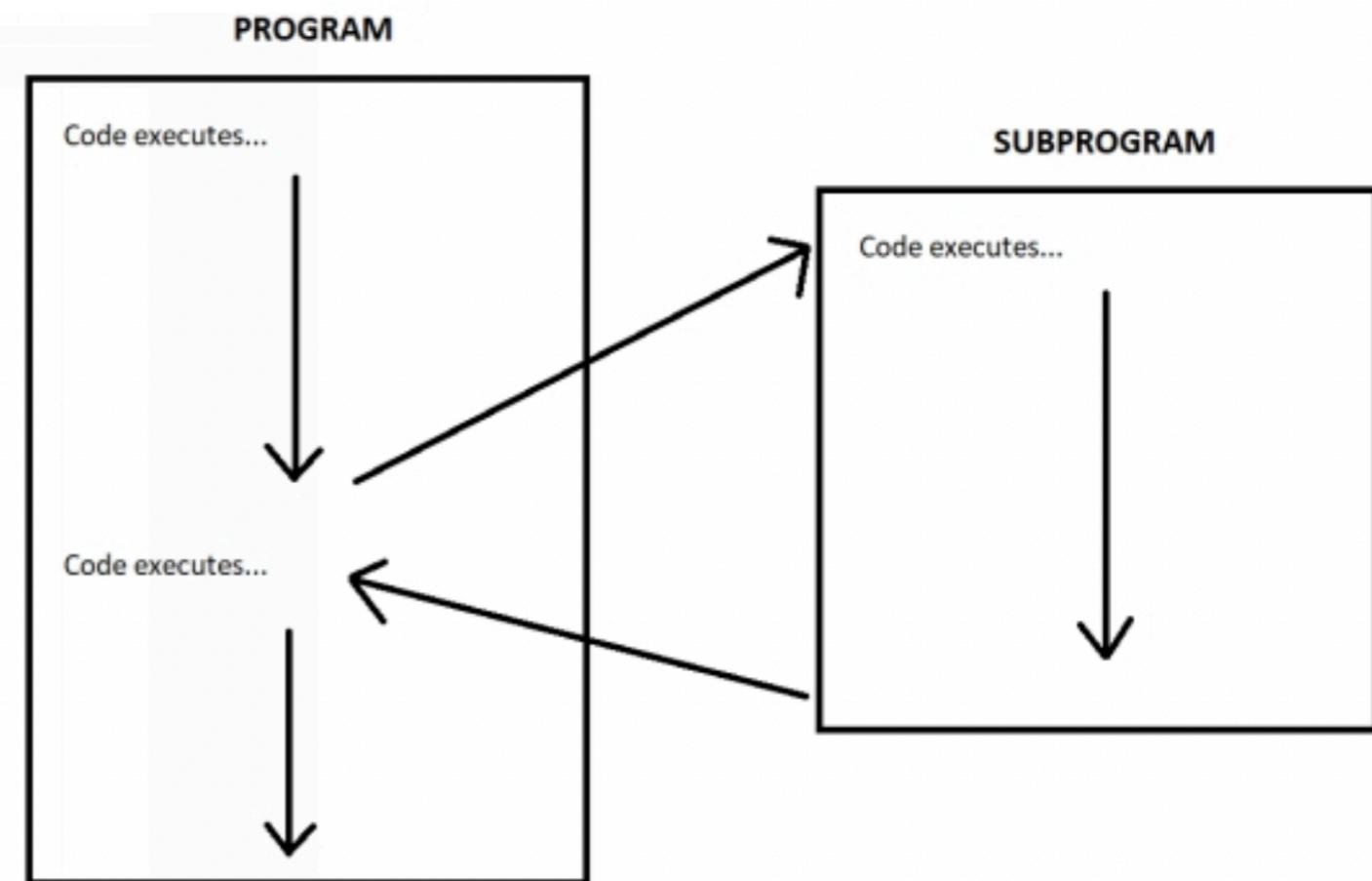
Que hace este código?

¿Puedo agregar funciones a mi computador?

No es necesario programar la rueda una y otra vez (concepto función)

- Una función es el trabajo de alguien.
 - Ejemplo : Bomberos combatir incendios.
- En programación una función es un bloque de código que hace una cosa particular.
- Lo interesante de las funciones es que se pueden reutilizar.
- Para hacer una función en Python usamos el comando **def**.
 - Nombre parametros return
- Funciones también se llaman subprogramas

```
def restar(a,b):  
    return a - b  
#llamando la nueva funcion  
print(restar(5,2))  
print(restar(9,5))
```



¿Puedo agregar funciones a mi computador?

No es necesario programar la rueda una y otra vez (concepto función)

```
1 def xxx(n):
2     i=0
3     t=0
4     while i < n:
5         t=t+i
6         i=i+1
7
8     return t
9
10 print(xxx(10))
11 print(xxx(6))
```

↳ 45
15

```
1 def zzzz(lista):
2     total=0
3     for i in lista:
4         total=total+i
5     return total/len(lista)
6
7 L=[1,2,5,6,9,10]
8
9 print(zzzz(L))
```

↳ 5.5

```
1 def yyyy(lista):
2     m = -1
3     for i in lista:
4         if(i > m):
5             m=i
6     return m
7
8 print(yyyy(L))
```

↳ 10

Que hacen las funciones?

Manipulación de archivos

```
file=open("file.txt")
for line in file:
    print(line)
file.close()
```

```
for line in open("file.txt"):
```

```
    line.rstrip()
    print(line)
```

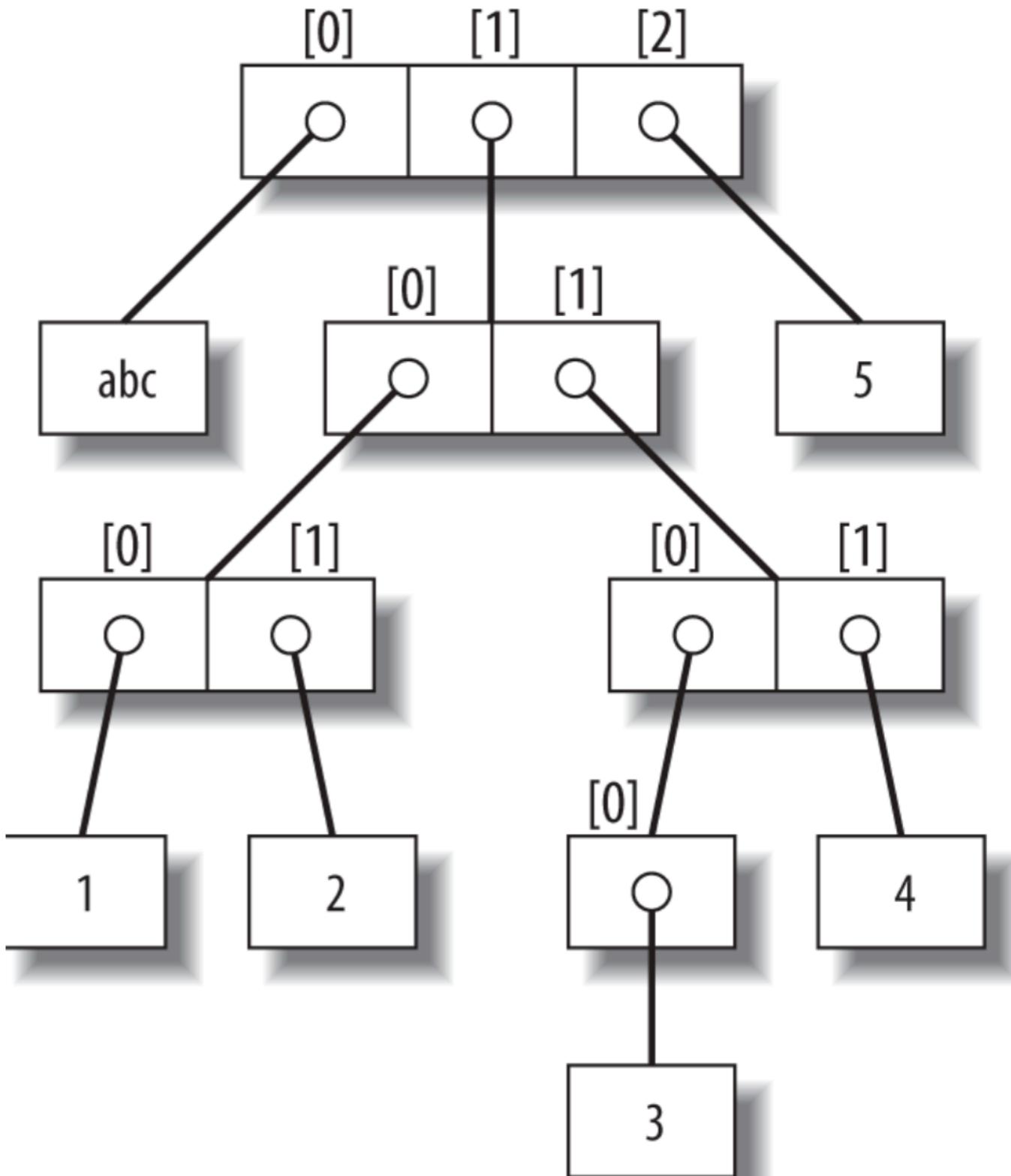
```
print(open("file.txt").read())
```

```
file=open("newfile.txt","w")
file.write("hola archivo\n")
file.close()
```

Estructuras complejas

L = ['abc', [(1, 2), ([3], 4)], 5]

- Como obtengo el valor abc?
- Como obtengo el valor 2?
- Como obtengo el valor 5?
- Como obtengo el valor 1?
- Como obtengo el valor 3?
- En python listas, diccionarios y tuplas pueden guardar cualquier objeto.

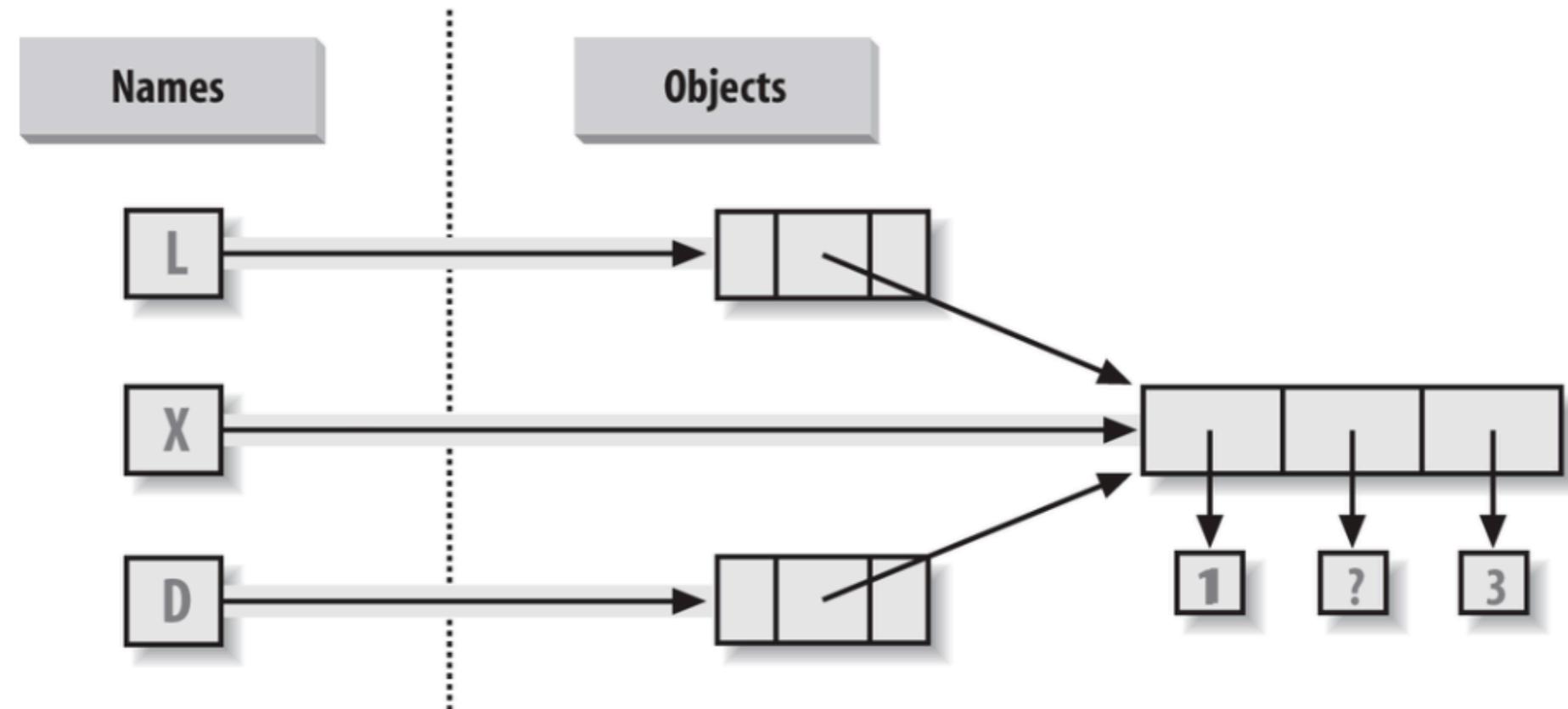


Referencia vs copias

X = [1,2,3]

L=[“a”,X,”b”]

D={‘x’:X, ‘y’:2}



X[1] = “sorpresa”

Copiar objetos

Objetos complejos

print(L) ?

Lista

#L = ['abc', [(1, 2), ([3], 4)], 5]

print(D) ?

L.copy()

Import copy

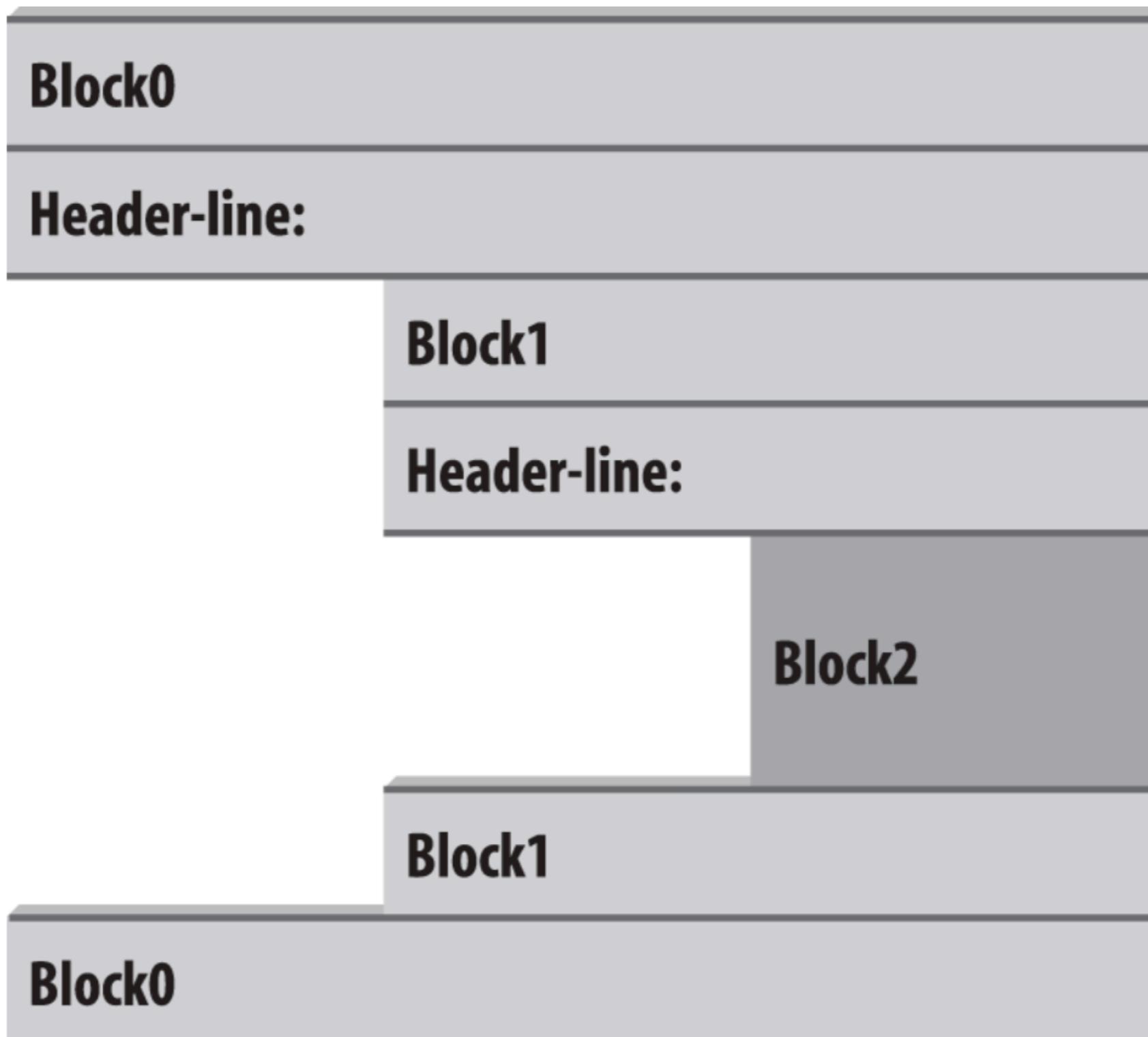
Dictionario

X=copy.deepcopy(L)

D.copy()

Reglas de identación

Bloques de código



Programación Orientada a Objetos

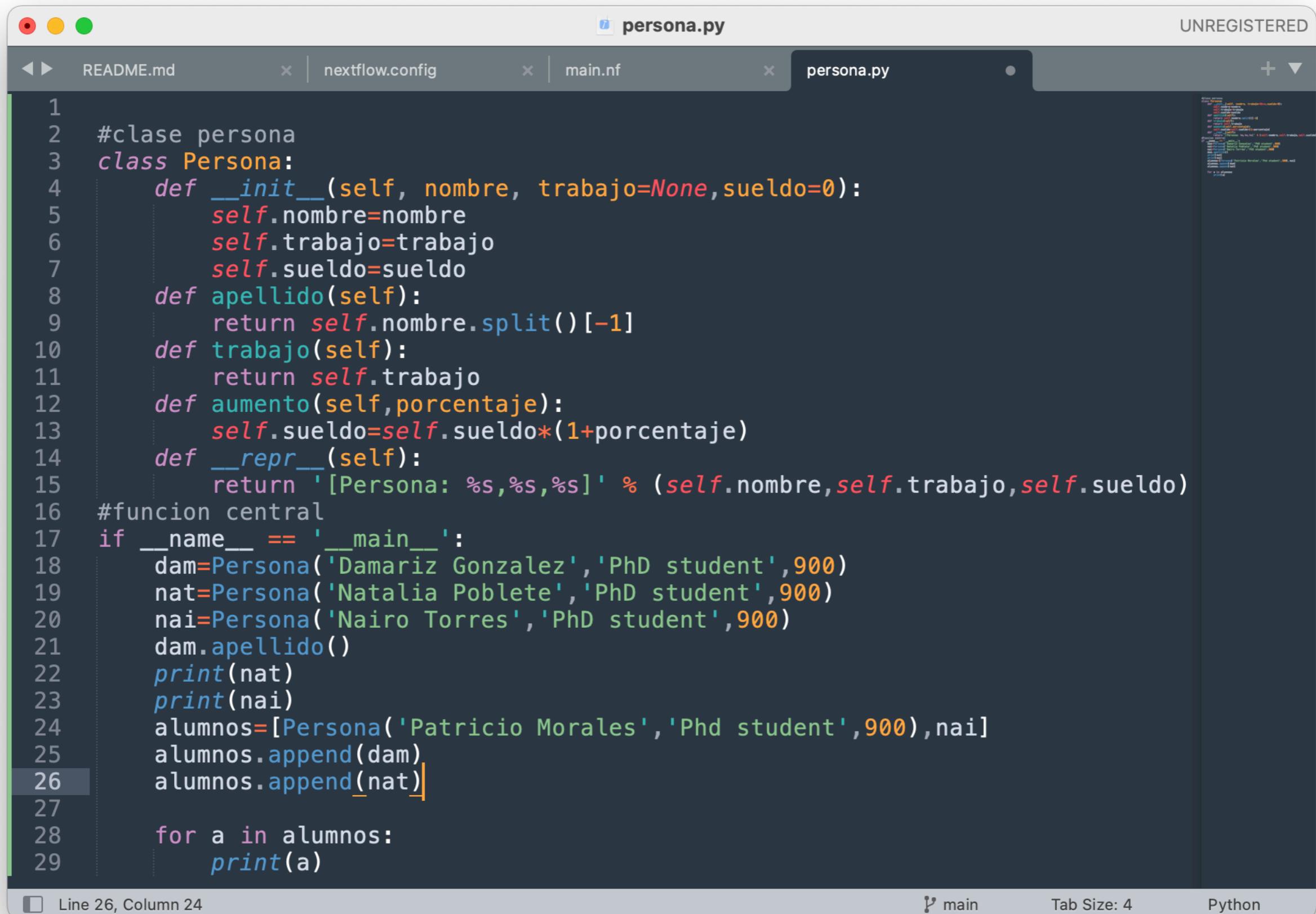
OOP

```
1 #clase persona
2 class Persona:
3     def __init__(self, nombre, trabajo=None, sueldo=0):
4         self.nombre=nombre
5         self.trabajo=trabajo
6         self.sueldo=sueldo
7     def apellido(self):
8         return self.nombre.split()[-1]
9     def trabajo(self):
10        return self.trabajo
11    def aumento(self,porcentaje):
12        self.sueldo=self.sueldo*(1+porcentaje)
13    def __repr__(self):
14        return '[Persona: %s,%s,%s]' % (self.nombre,self.trabajo,self.sueldo)
15
16 if __name__ == '__main__': ...|
```

- Mi primera clase

Programación Orientada a Objetos

OOP



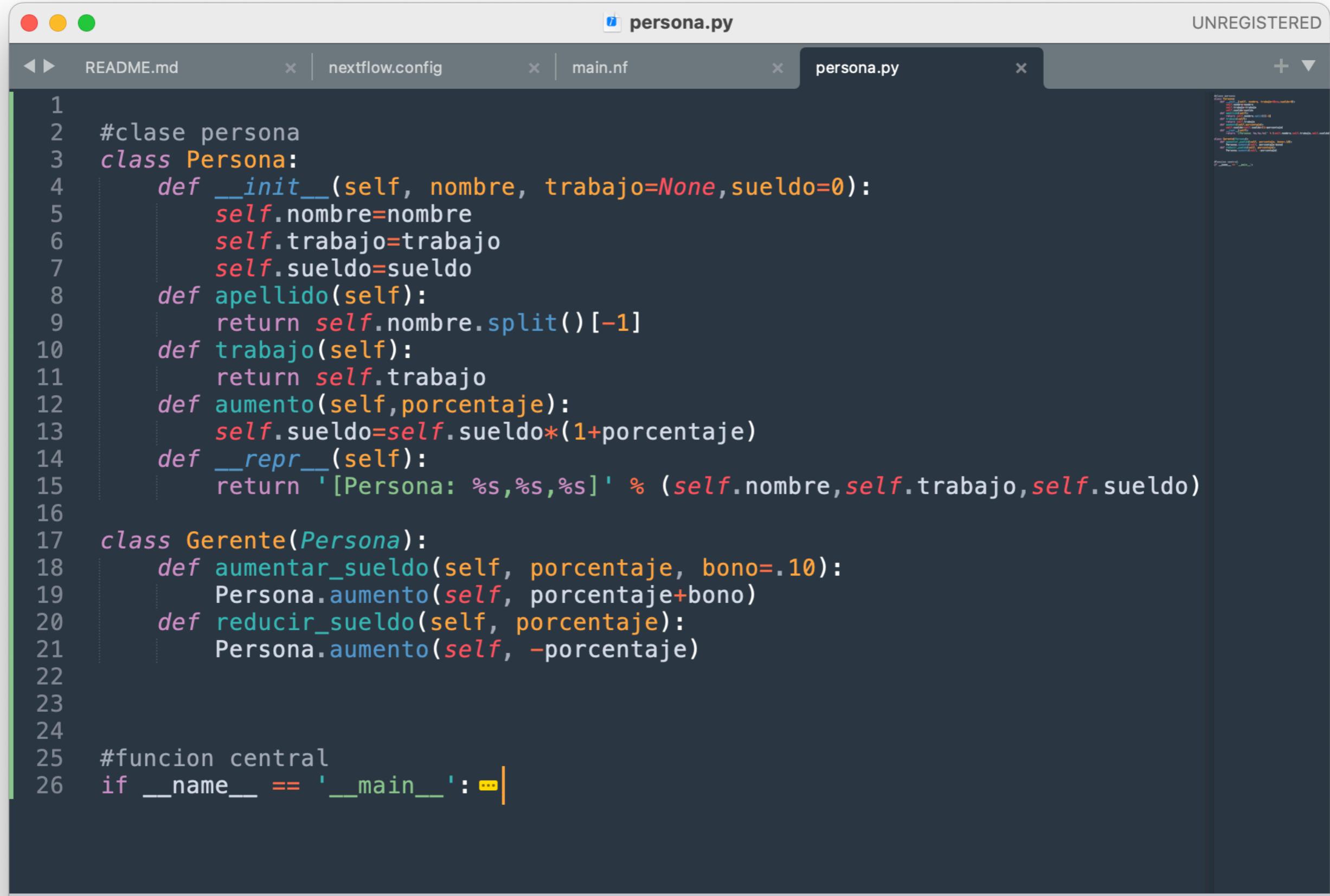
The screenshot shows a terminal window with several tabs open. The active tab is named "persona.py". The code in the terminal is as follows:

```
1 #clase persona
2 class Persona:
3     def __init__(self, nombre, trabajo=None,sueldo=0):
4         self.nombre=nombre
5         self.trabajo=trabajo
6         self.sueldo=sueldo
7     def apellido(self):
8         return self.nombre.split()[-1]
9     def trabajo(self):
10        return self.trabajo
11    def aumento(self,porcentaje):
12        self.sueldo=self.sueldo*(1+porcentaje)
13    def __repr__(self):
14        return '[Persona: %s,%s,%s]' % (self.nombre,self.trabajo,self.sueldo)
15 #funcion central
16 if __name__ == '__main__':
17     dam=Persona('Damariz Gonzalez','PhD student',900)
18     nat=Persona('Natalia Poblete','PhD student',900)
19     nai=Persona('Nairo Torres','PhD student',900)
20     dam.apellido()
21     print(nat)
22     print(nai)
23     alumnos=[Persona('Patricio Morales','Phd student',900),nai]
24     alumnos.append(dam)
25     alumnos.append(nat)
26
27     for a in alumnos:
28         print(a)
```

The code defines a `Persona` class with methods for initializing attributes, getting the last name, getting the job, applying a percentage increase to the salary, and returning a string representation of the object. It also contains a main function that creates three `Persona` objects and prints their details.

Programación Orientada a Objetos

Herencia



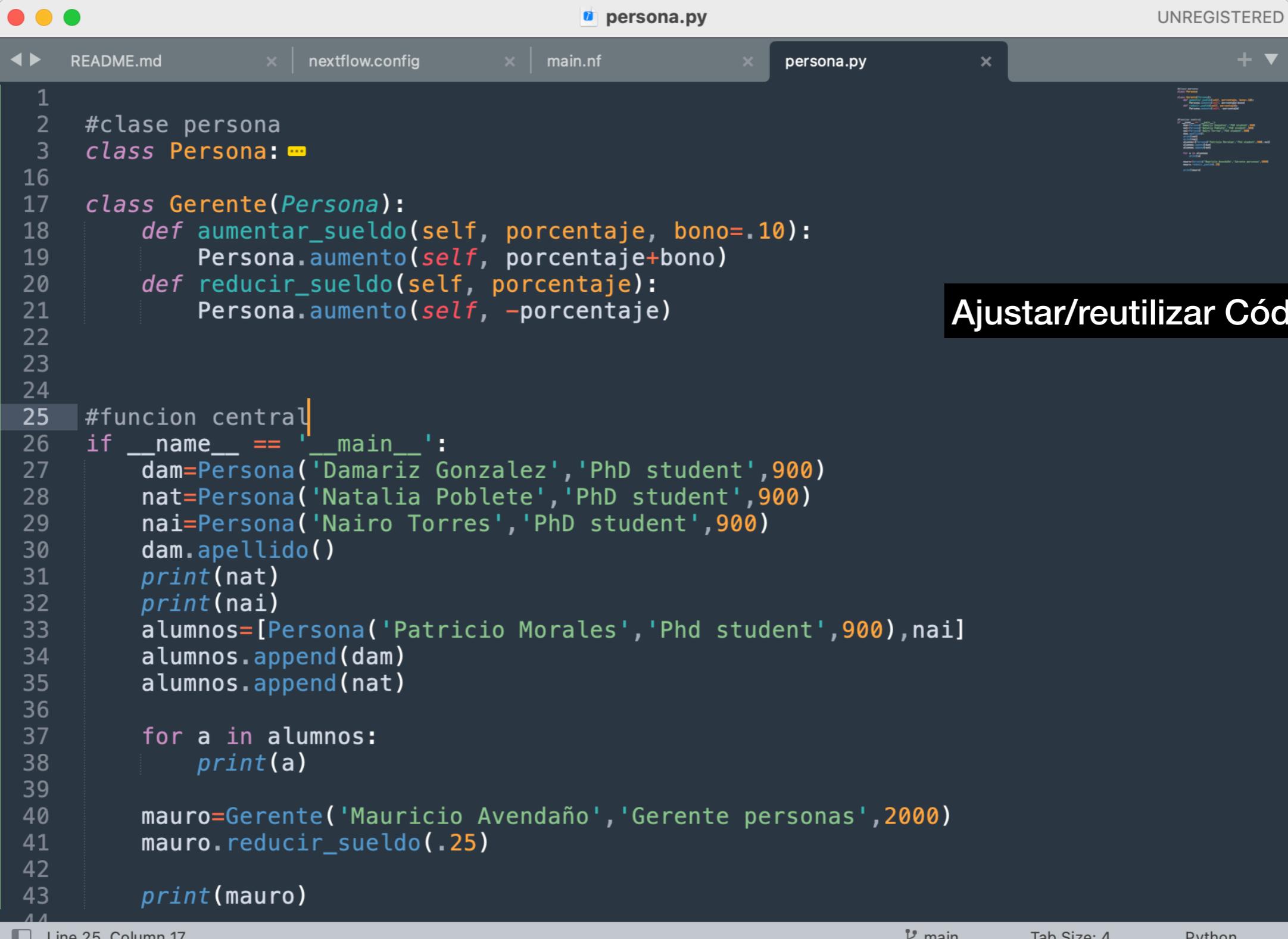
The screenshot shows a code editor window with a dark theme. The title bar says "persona.py". The tabs at the top include "README.md", "nextflow.config", "main.nf", and "persona.py". The code editor displays the following Python code:

```
1 #clase persona
2 class Persona:
3     def __init__(self, nombre, trabajo=None, sueldo=0):
4         self.nombre=nombre
5         self.trabajo=trabajo
6         self.sueldo=sueldo
7     def apellido(self):
8         return self.nombre.split()[-1]
9     def trabajo(self):
10        return self.trabajo
11    def aumento(self,porcentaje):
12        self.sueldo=self.sueldo*(1+porcentaje)
13    def __repr__(self):
14        return '[Persona: %s,%s,%s]' % (self.nombre,self.trabajo,self.sueldo)
15
16
17 class Gerente(Persona):
18     def aumentar_sueldo(self, porcentaje, bono=.10):
19         Persona.aumento(self, porcentaje+bono)
20     def reducir_sueldo(self, porcentaje):
21         Persona.aumento(self, -porcentaje)
22
23
24
25 #funcion central
26 if __name__ == '__main__': ...
```

The code defines a base class `Persona` with methods `apellido`, `trabajo`, `aumento`, and `__repr__`. It also defines a derived class `Gerente` that inherits from `Persona` and adds a method `aumentar_sueldo`. The script ends with a conditional block to run the code when executed directly.

Programación Orientada a Objetos

Herencia



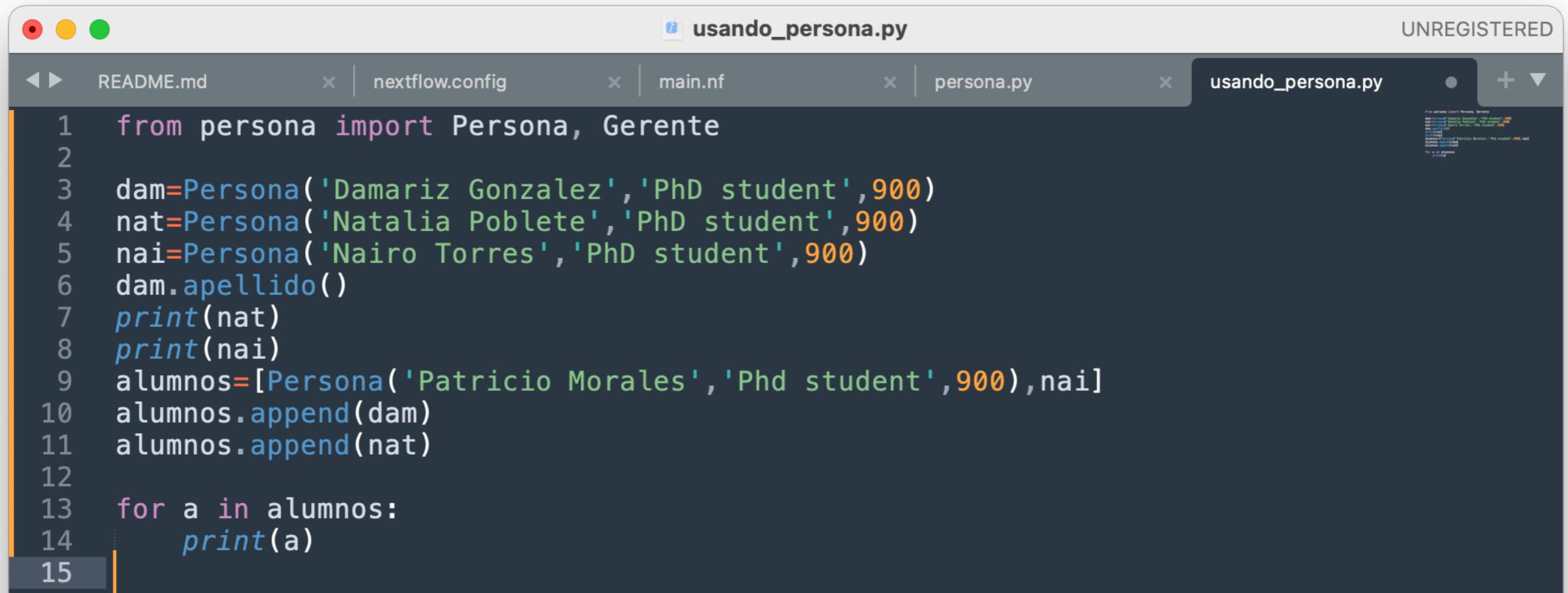
Ajustar/reutilizar Código

```
1 #clase persona
2 class Persona:
3     ...
4
5     class Gerente(Persona):
6         def aumentar_sueldo(self, porcentaje, bono=.10):
7             Persona.aumento(self, porcentaje+bono)
8         def reducir_sueldo(self, porcentaje):
9             Persona.aumento(self, -porcentaje)
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25 #funcion central
26 if __name__ == '__main__':
27     dam=Persona('Damariz Gonzalez','PhD student',900)
28     nat=Persona('Natalia Poblete','PhD student',900)
29     nai=Persona('Nairo Torres','PhD student',900)
30     dam.apellido()
31     print(nat)
32     print(nai)
33     alumnos=[Persona('Patricio Morales','Phd student',900),nai]
34     alumnos.append(dam)
35     alumnos.append(nat)
36
37     for a in alumnos:
38         print(a)
39
40     mauro=Gerente('Mauricio Avendaño','Gerente personas',2000)
41     mauro.reducir_sueldo(.25)
42
43     print(mauro)
44
```

Line 25, Column 17 main Tab Size: 4 Python

Programación Orientada a Objetos

Utilizando la clase desde otro programa



```
usando_persona.py
UNREGISTERED
README.md | nextflow.config | main.nf | persona.py | usando_persona.py
1 from persona import Persona, Gerente
2
3 dam=Persona('Damariz Gonzalez','PhD student',900)
4 nat=Persona('Natalia Poblete','PhD student',900)
5 nai=Persona('Nairo Torres','PhD student',900)
6 dam.apellido()
7 print(nat)
8 print(nai)
9 alumnos=[Persona('Patricio Morales','Phd student',900),nai]
10 alumnos.append(dam)
11 alumnos.append(nat)
12
13 for a in alumnos:
14     print(a)
15
```

Tarea1 y Control 1

Fechas

- Tarea 1:
 - Entrega enunciado 19/04
 - Fecha de entrega 03/05
- Control 1:
 - Fecha : 09/05
 - Contenidos: Unidad 1 y Unidad 2 (-nextflow)