__init__: Introducción a la programación con Python!

Berenice Larsen y **Andrés Delfino**
berelarsenp@gmail.com y adelfino@gmail.com

Virtudes

- Libre
- Maduro
- Expresivo
- Legible
- Fácil de aprender
- Cuenta con una amplia biblioteca estándar

Guido van Rossum

- Creador de Python
- Dictador benevolente de por vida
- Participó en la creación de ABC
- Participa del desarrollo de mypy
- Sitio personal
- Neopythonic
- All Things Pythonic
- The History of Python


Python
is for girls!

Python is for girls!

- Organizaciones internacionales:
  - DjangoGirls
  - PyLadies
- Organizaciones nacionales
  - LinuxChix
  - Chicas en tecnología
  - DjangoGirls Argentina
  - [LAS]DeSistemas

Python is for girls!

- Preguntas solo de mujeres
- Preguntas alternadas entre mujeres y varones
- Remera “Python is for girls”

Compatibilidad de versiones

Python 3 y Legacy Python

- Python 3
  - Salió en 2008
  - Introdujo cambios incompatibles con versiones anteriores
  - Proyectos nuevos se basan en esta versión
- Legacy Python (Python 2)
  - Última major version (2.7) liberada en 2010
  - Solo recibe correcciones
  - Exclusivamente para soportar proyectos existentes
  - Fecha de vencimiento: 1° de enero de 2020

Poniendo en contexto

Definición de bloques

- Se sigue la off-side rule: los bloques son definidos por indentación
- Facilita mantener el código legible

PEP8

Ejemplo

Ejemplos

```
if True:
    if True:
            pass
    else:
     pass

if True:
    pass
 pass
#IndentationError
```

Tipado

- Dinámico
  - Los chequeos de tipado se producen en tiempo de ejecución
- Fuerte
  - No hay conversiones implícitas salvo contadas excepciones (como la conversión entre valores numéricos).
- Sigue la filosofía duck typing
  - El énfasis está puesto en las características de un objeto y no en su clase
- Incluye herramientas opcionales para posibilitar el análisis de tipado estático

Identificadores

- Las minúsculas y mayúsculas son significativas (case sensitive)
- Varios identificadores pueden hacer referencia a un mismo objeto

Ejemplo

Ejemplos

```
a = 1
print(A)                    #NameError

a = 1
b = a
print(a, b, a is b) #1 1 True
a = 2
print(a, b, a is b) #2 1 False
```

Mutabilidad

- Objeto inmutable
  - No puede ser modificado tras su creación
  - Puede hacer referencia a objetos mutables
- Objeto mutable
  - Puede ser modificado tras su creación
  - Puede ser modificado a través de cualquier identificador que lo referencie
  - Para copiar objetos mutables se hacen shallow copies

Ejemplo

Ejemplo

```
tupla = 1, "A"

lista = []
tupla = lista,

print(tupla)     #([],)

lista.append(1)
print(tupla)     #([1],)

lista2 = lista
lista2.append(2)
print(tupla)     #([1, 2],)
```

Tipos de datos

Booleano y Numéricos

- bool
  - True (1) o False (0)
- int
  - Precisión ilimitada
- float
- complex

Ejemplo

Ejemplos

```
i = 18_446_744_073_709_551_615
i = 0xffffffffffffffff
i = 0o1777777777777777777777
i = 0b1111

f = 2.7
f = 1e-003

c = 5+2j
```

Secuencias

Objetos que pueden contener varios ítems y que permiten acceder a cualquiera de ellos a través de un índice entero

- str (inmutable)
- Binarias
  - bytes (inmutable)
  - bytesarray (mutable)
- Estructuras
  - tuple (inmutable)
  - list (mutable)

Secuencias - Cadenas de texto: str

- Son delimitadas por comillas simples/dobles, o triple comilla simple/doble
- Las comillas de un tipo pueden ser incluidas en cadenas delimitadas por comillas de otro tipo
- Se pueden escapar las comillas con barra inversa
- Se pueden representar caracteres de control con barra inversa
- Codificadas por defecto en Unicode

Ejemplo

Ejemplos

```
print('Python', "es", '''para''', """mujeres""")

print('Bonito "sombrero"')

print('Escapando una comilla simple: \'')

print('Línea 1', '\n', 'Línea 2', sep='')

print('\U0001F40D', '\N{snake}')
```

Formateo de texto

Crear cadenas de texto que incluyan variables formateadas de una determinada manera.

- Literal f (formatted string literal)
- Método format
- Operador módulo (interpolación o printf style)

Ejemplo

Ejemplos

```
nombre = 'Laura'

print(f'{nombre} no está')

print('{} se fue'.format(nombre))

print('%s se escapa de mi vida' % nombre)
```

Secuencias - Estructuras secuenciales

Objetos que pueden contener ítems de diferentes tipos.

- tuple
  - Creación por constructor o al separar expresiones por comas
  - Inmutable
- list
  - Creación por constructor o por display
  - Mutable

Ejemplos

Ejemplos

```
tupla = ()
tupla = 'mate',
tupla = 'mate', 'café', 'harina'

lista = []
lista = ['mate']
lista = ['mate', 'café', 'harina']
lista.append('palmitos')
print(lista)
```

Subscripting

- Referencia a un ítem de una secuencia
- La numeración inicia en cero
- Se accede de forma relativa al final de la secuencia con valores negativos

Ejemplo

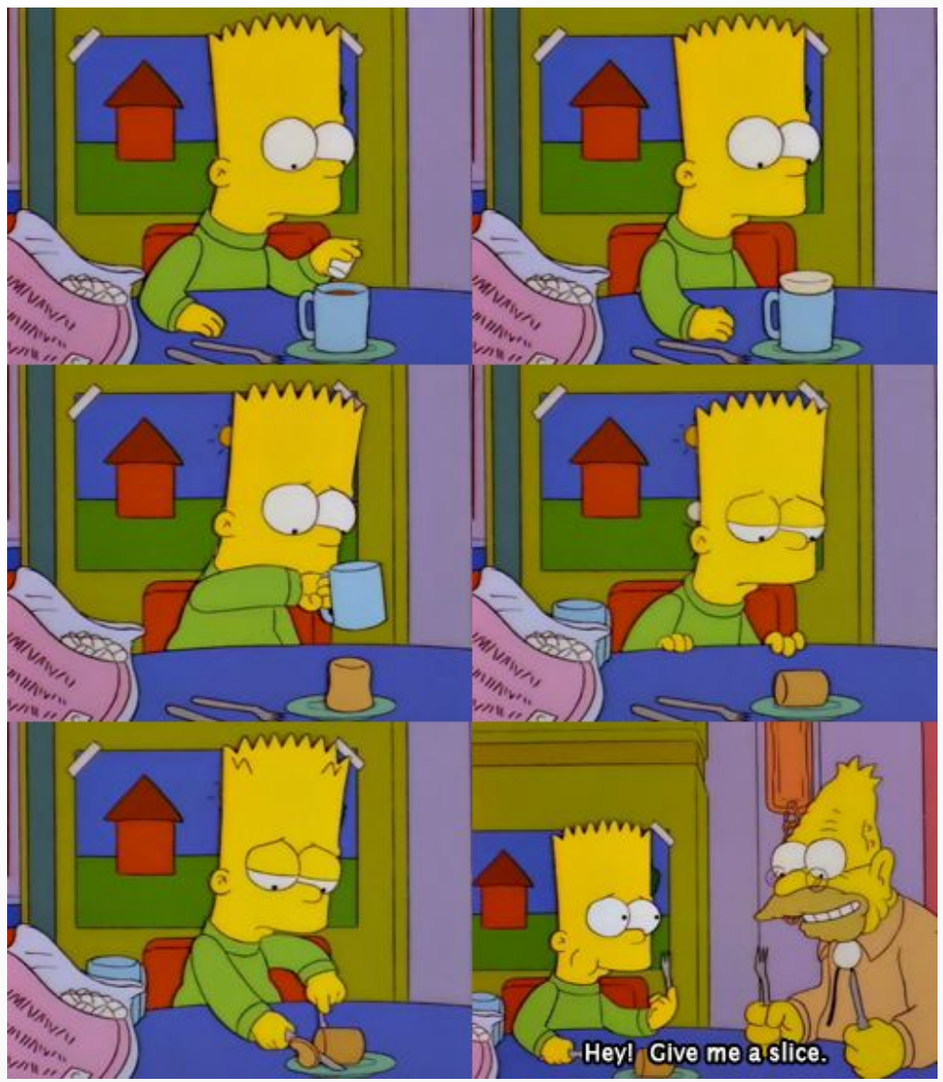
Ejemplos

```
lista = ['yerba', 'mermeladas',
         'cacoa', 'picadillo']

print(lista[0])  # yerba
print(lista[-1]) # picadillo

lista[2] = 'cacao'
print(lista)
```

Slicing

- Referencia a más de un ítem de una secuencia
- Se puede especificar inicio, final y paso
- El inicio es inclusivo
- El final es exclusivo
- Se pueden seleccionar ítems no consecutivos especificando la cantidad de pasos

[Ejemplo]



Ejemplos

```
lista = ['x', 'x', 'x', 'paté', 'caballa',
         'arroz', 'arvejas', 'x', 'x', 'x']
print(lista[3:-3])

lista = ['sardinas', 'x', 'atún', 'x',
         'choclo', 'x', 'lentejas']
print(lista[::2])

lista2 = lista[:] #crear shallow copy
```

Hasheabilidad

Tienen un hash que no cambia durante todo su ciclo de vida
Intervienen los métodos especiales __hash__ y __eq__

- Objetos hasheables
  - Los objetos inmutables built-in cuando solo contienen objetos hasheables, entre otros
  - Las instancias de clases creadas por el usuario (son iguales únicamente a si mismas)
- Objetos no hasheables
  - Los objetos mutables built-in, entre otros

Conjuntos

- set
  - Estructura de datos que contiene objetos hasheables diferentes
  - No preserva el orden en que se añaden los ítems
  - Creación por constructor o por display
  - Mutable
- frozenset
  - Creación por constructor
  - Inmutable

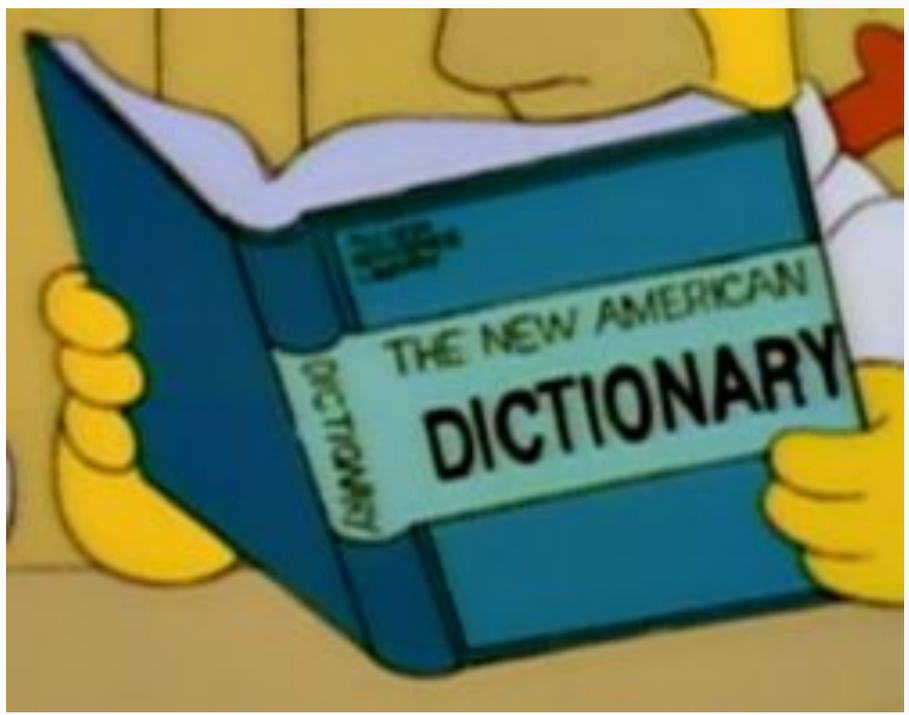
Ejemplo

Ejemplos

```
conjunto = set()

conjunto = {'c', 'a', 'b', 'c'}
conjunto.add('a')
print(conjunto)                    #{'a', 'b', 'c'}

conjunto1 = {'a', 'b', 1, 3}
conjunto2 = {'b', 2, 4, 'r', 'a'}
print(conjunto1 & conjunto2)       #{'a', 'b'}
```

dict

- Estructura de datos que asocia un objeto hasheable (clave) a otro objeto (valor)
- Preserva el orden en que se añaden los ítems a partir de la versión 3.7
- Creación por constructor o por display
- Mutable



Ejemplo

Ejemplos

```
diccionario = {}

diccionario = {'lenny': (0xff, 0xff, 0xff),
               'carl':  (0x00, 0x00, 0x00)}
diccionario[1, 2, 3] = 1
diccionario[print] = 1

diccionario.keys()
diccionario.values()
diccionario.items()
```

Valores de verdad (truthy and falsy)

- Valores considerados falsos
  - None
  - Números iguales a 0
  - Secuencias vacías
  - Conjuntos vacíos
  - Diccionarios vacíos
  - Los objetos cuyo método especial __bool__ retorna False
  - Los objetos cuyo método especial __len__ retorna 0
- Se consideran verdaderos el resto de los valores

Comprobación de tipo y conversión

- type
  - Retorna el tipo de un objeto
- isinstance
  - Comprueba si un objeto es instancia de una determinada clase
- issubclass
  - Comprueba si una clase es una subclase de otra

Ejemplos

Ejemplos

```
type(123)                          #<class 'int'>
isinstance(123, int)               #True
issubclass(dict, Exception) #False

num_int = 123
a_str = str(num_int)
print(type(a_str))

to_list = list(a_str)
print(to_list)
```

Operadores

Operadores

- Aritméticos
  `+, -, *, /, %, **, //`
- Comparación
  `<, <=, >, >=,  ==, !=`
- Booleanos
  `and, or, not`
- Identitarios y de contención
  `is, is not, in, not in`

Ejemplo

- Bitwise
  `&, |, ^, ~`
- Shifting
  `<<, >>`

Ejemplos

```
1 in [1, 2, 3]

'hams' in 'steamed hams'

respuesta.upper() in ('S', 'N')

diccionario = {1: 'uno', 2: 'dos'}
'dos' in diccionario
'dos' in diccionario.values()
```

Particularidades en expresiones

- El funcionamiento de los operadores está determinado por las clases de los objetos en que se aplican
- Una misma expresión puede ser operando de dos operadores diferentes
- Las asignaciones son sentencias y por lo tanto no pueden ser incluidas en expresiones

Ejemplo

Ejemplos

```
'Felicitaciones, %s' % 'Shinji'
5 % 2

1 <= 54 <= 100

a, b = 1, 2
a, b = b, a
print(a, b)
```

Retornos de and y or

- and
  - Retorna el valor de la primera expresión si es falsa; si no, retorna el valor de la segunda expresión
- or
  - Retorna el valor de la primera expresión si es verdadera; si no, retorna el valor de la segunda expresión

Ejemplo

Ejemplos

```
print('Mariana' and '')              #''
print('Mariana' and 'López')         #López

print('Mariana' or 'm4r14n4')        #Mariana
print('' or 'm4r14n4')               #m4r14n4

profesion = input('Profesión: ')
profesion = profesion or 'Empleadx'
print(profesion)
```

Estructuras de control

Condicionales

- if
- Expresiones condicionales

Ejemplo

Ejemplos

```
x = 1
if x == 1:
    print('Uno')
elif x == 2:
    print('Dos')
else:
    print('Ni uno ni dos')

'Uno' if x == 1 else 'Diferente a uno'
```

Iterables

- Un iterable es un objeto que puede devolver ítems de uno en uno
- Todas las secuencias son iterables
- Algunos iterables built-in
  - Cadenas de texto (por cada caracter)
  - bytes, bytearray (por cada byte)
  - Tuplas, listas, conjuntos (por cada ítem)
  - Diccionarios (por cada clave, valor, o par clave/valor)

Bucles

- for
  - Ejecuta un bloque por cada ítem de un iterable
  - La clase range puede usarse para producir un bucle por sucesión numérica
  - El bloque else se ejecuta si no se llama a break
- while
  - El bloque else se ejecuta si no se llama a break

Ejemplos

Ejemplos

```
for letra in 'prueba':
    print('Letra:', letra)

for n in range(10):
    if n == 5:
        break
else:
    print('No se ejecutó break')
```

Ejemplos

```
flag = True
i = 0
while flag:
    i += 1
    if i == 5:
        flag = False
else:
    print('No se ejecutó break')
```

Comprehensions

- Crear una lista, diccionario o set a partir de un iterable expresado en una única línea
- Se puede incluir una condición
- Se pueden procesar los ítems producidos por el for antes de agregarlos al nuevo iterable

Ejemplo

Ejemplos

```
palabras = ('HAS', 'eN', 'hOTel', 'Buscado',
            'aLGuNa', 'vez', 'Un', 'inTERnET')

palabras = [p.lower() for p in palabras]

{p: set(p) for p in palabras if 'n' in p}

{len(p) for p in palabras}
```

Módulos y paquetes

Módulos y paquetes

- Permiten la reutilización de código importando archivos
- Todo script es un módulo que puede ser importado
- Es importante tenerlo presente a la hora de escribir nuestros programas para que sean fácilmente reusables
- Se consulta el nombre del módulo para saber si está siendo importado
- Los conjunto de módulos relacionados son agrupados en paquetes

Manejo de excepciones

Manejo de excepciones

- Las excepciones son instancias de la clase Exception, creadas por el intérprete o por el usuario
- Se producen con la sentencia raise
- Se manejan con la sentencia try

Ejemplo

Ejemplos

```
n = 0
try:
    1 / n
except (ZeroDivisionError, OSError) as e:
    print('Excepción', e)
    1 / n
else:
    print('else')
finally:
    print('finally')
```

traceback

- Cuando se produce una excepción, Python crea un objeto traceback
- Los traceback contienen información detallada sobre las condiciones en que se generó la excepción

Ejemplo

Ejemplo

```
def dividir(a, b):
    return a / b

def operar(operacion, a, b):
    if operacion == 'dividir':
        return dividir(a, b)

resultado = operar('dividir', 54, 0)
```

Ejemplo

```
Traceback (most recent call last):
  File "...", line 7, in <module>
  operar(0, 1, 0)
  File "...", line 5, in operar
  dividir(a, b)
  File "...", line 2, in dividir
  a / b
ZeroDivisionError: division by zero
```

Funciones

Funciones

- Pueden recibir parámetros por posición o por keyword
- Los argumentos pueden tener valores por defecto
- Pueden recibir una cantidad variable de argumentos
- Pueden contener otras funciones
- Por defecto retornan None

Ejemplo

Ejemplos

```
def restar(a, b=2):
    return a - b

print(restar())           #TypeError
print(restar(3))          #1
print(restar(9, 4))       #5
print(restar(b=3, a=10)) #7
```

Ejemplos

```
def sumar(*args):
    suma = 0
    for n in args:
        suma += n
    return suma

def preparar(diccio, **kwargs):
    for k, v in kwargs.items():
        diccio[k] = v
```

Ámbitos de variables (scopes)

- Variables locales
  - Las creadas en el bloque en ejecución
- Variables globales
  - Las creadas a nivel módulo
- Variables no locales
  - Las que no son locales o globales
  - Se pueden encontrar en funciones anidadas

Acceso a variables globales y no locales

- Lectura
  - Acceso de lectura si no se las modifica
  - Si se las intenta modificar, se crean variables locales
- Escritura
  - global
    - Permite modificar variables globales
  - nonlocal
    - Permite modificar variables no locales

Ejemplo

Ejemplos

```
variable = 1

def f():
    print(variable)
f()

def f2():
    variable = 2
f2()
print(variable)
```

```
def f3():
    global variable
    variable = 2
f3()
print(variable)
```

Ejemplos

```
x = 0
def outer():
    x = 1
    def inner():
        x = 2
        print("inner:", x)

    inner()
    print("outer:", x)

outer()
print("global:", x)
```

```
x = 0
def outer():
    x = 1
    def inner():
        nonlocal x
        x = 2
        print("inner:", x)

    inner()
    print("outer:", x)

outer()
print("global:", x)
```

Objetos (OOP)

Clases

- Todas son instancias de object
- Pueden heredar de múltiples superclases
- Sus atributos siempre son públicos
- Pueden ser modificadas en tiempo de ejecución
- Una variable puede ser reemplazada por una propiedad (con getter, setter y deletter) sin necesidad de cambiar la API

Ejemplo

Métodos

- Funciones que reciben como primer argumento la instancia por la cual son invocados
- Métodos de clase
  - Creados con el decorador classmethod
  - Recibe como primer parámetro la clase de la instancia
- Métodos estáticos
  - Creados con el decorador staticmethod
  - No recibe como primer parámetro la instancia de la clase

Ejemplos

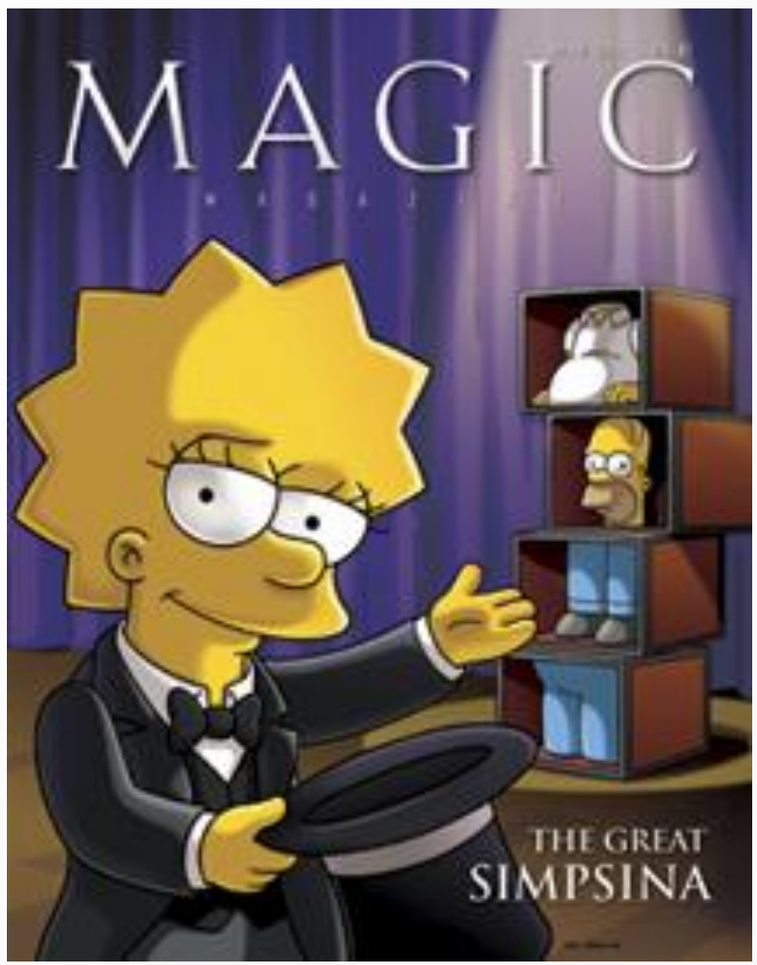
```
class Empleadx:
    def __init__(self, nombre):
        self.nombre = nombre

    def saludar(self):
        print(f'Soy {self.nombre}')

e = Empleadx('Paz')
e.saludar()
```

Ejemplos

```
class Empleadx:
    def __init__(self, nombre):
        self.nombre = nombre

    @property
    def nombre(self):
        return self._nombre

    @nombre.setter
    def nombre(self, valor):
        self._nombre = valor

    def saludar(self):
        print(f'Soy {self.nombre}')

e = Empleadx('Paz')
e.saludar()
```

Métodos especiales (magic/dunder methods)

- Son invocados por Python en determinadas circunstancias
- __str__ provee una representación textual de la instancia
- __repr__ provee una representación que *debería* indicar cómo reproducir una instancia
- Protocolo de context manager
- Protocolo de iterador



Context managers

- Clases que implementan los métodos especiales __enter__ y __exit__
- __enter__ monta un contexto (ej, abre un archivo)
- __exit__ desmonta el contexto (ej, cierra el archivo abierto)
- with
  - Ejecuta __enter__, luego un bloque y finalmente __exit__, aún si durante la ejecución del bloque se produjo una excepción

Ejemplo

Ejemplos

```
try:
    archivo = open('archivo')
    for linea in archivo:
        print(linea)
finally:
    archivo.close()
```

Ejemplos

```
suma = 0

with open(numeros.txt') as numeros:
    for linea in numeros:
        suma += int(linea)

print(suma)
```

Endut!
Hoch Hech!