

Laboratorio de Datos

Primer cuatrimestre de 2023

Introducción a Python - parte 2

Contenido

- + Repaso ejercicios
- + Python desde la terminal
- + Correr archivos .py
- + IDE - Spyder
- + Funciones
- + Diccionarios
- + Módulos
- + Numpy
- + Manejo de archivos

Repaso

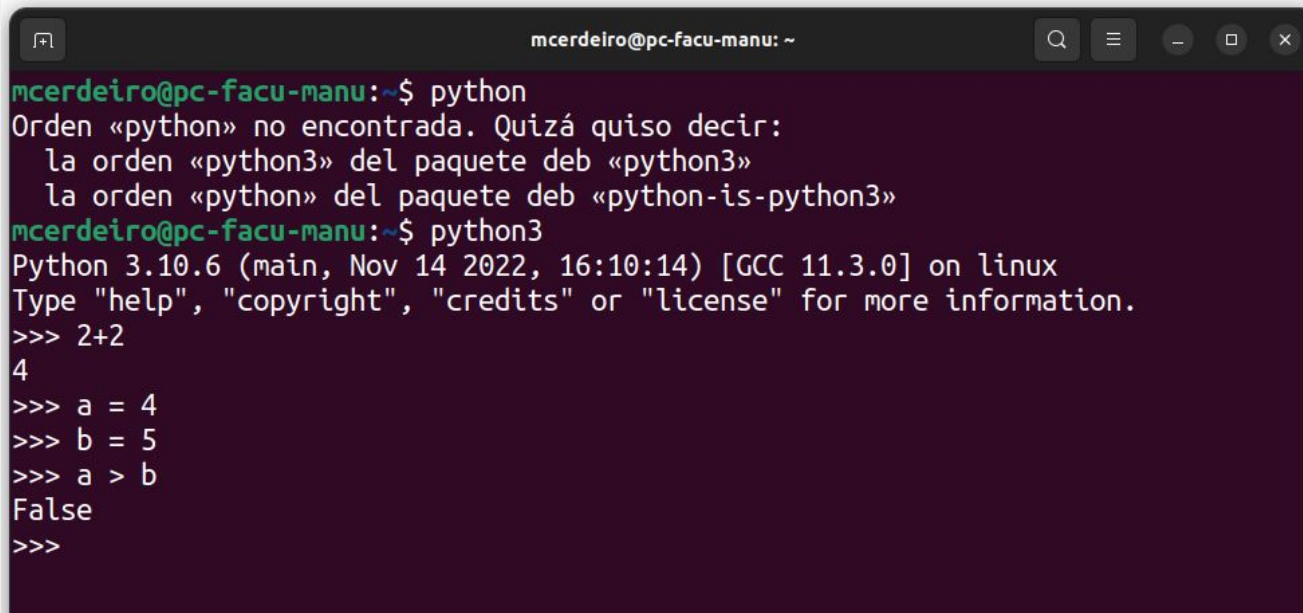
Ejercicios

- + Geringoso
- + Rebotes
- + Hipoteca
- + Inclusive

Python desde la terminal

Python desde la terminal

En una terminal, usamos "python" o "python3" para abrir un intérprete de python.

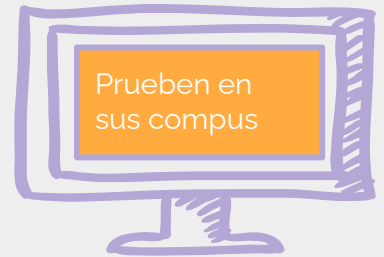
A terminal window with a dark background and light text. The window title is 'mcerdeiro@pc-facu-manu: ~'. The prompt is 'mcerdeiro@pc-facu-manu:~\$'. The first command is 'python', which results in an error message: 'Orden «python» no encontrada. Quizá quiso decir: la orden «python3» del paquete deb «python3» la orden «python» del paquete deb «python-is-python3»'. The second command is 'python3', which successfully starts the Python 3.10.6 interpreter. The interpreter shows the version and date, and prompts for help. The user enters '2+2', and the interpreter returns '4'. The user enters 'a = 4', 'b = 5', and 'a > b', and the interpreter returns 'False'. The prompt 'mcerdeiro@pc-facu-manu:~\$' is visible at the top of the terminal window.

```
mcerdeiro@pc-facu-manu:~$ python
Orden «python» no encontrada. Quizá quiso decir:
  la orden «python3» del paquete deb «python3»
  la orden «python» del paquete deb «python-is-python3»
mcerdeiro@pc-facu-manu:~$ python3
Python 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+2
4
>>> a = 4
>>> b = 5
>>> a > b
False
>>>
```

Python desde la terminal

Al terminar, salimos con `exit()`

```
mcerdeiro@pc-facu-manu: ~  
mcerdeiro@pc-facu-manu:~$ python  
Orden «python» no encontrada. Quizá quiso decir:  
  la orden «python3» del paquete deb «python3»  
  la orden «python» del paquete deb «python-is-python3»  
mcerdeiro@pc-facu-manu:~$ python3  
Python 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 2+2  
4  
>>> a = 4  
>>> b = 5  
>>> a > b  
False  
>>> exit()  
mcerdeiro@pc-facu-manu:~$
```



Archivos .py

Uso de archivos .py

Podemos guardar el código como archivo de texto con extensión .py y luego ejecutarlo desde la terminal con python. Por ejemplo:
Creamos un archivo de texto, con este contenido, y lo guardamos como hipoteca.py

```
# hipoteca.py

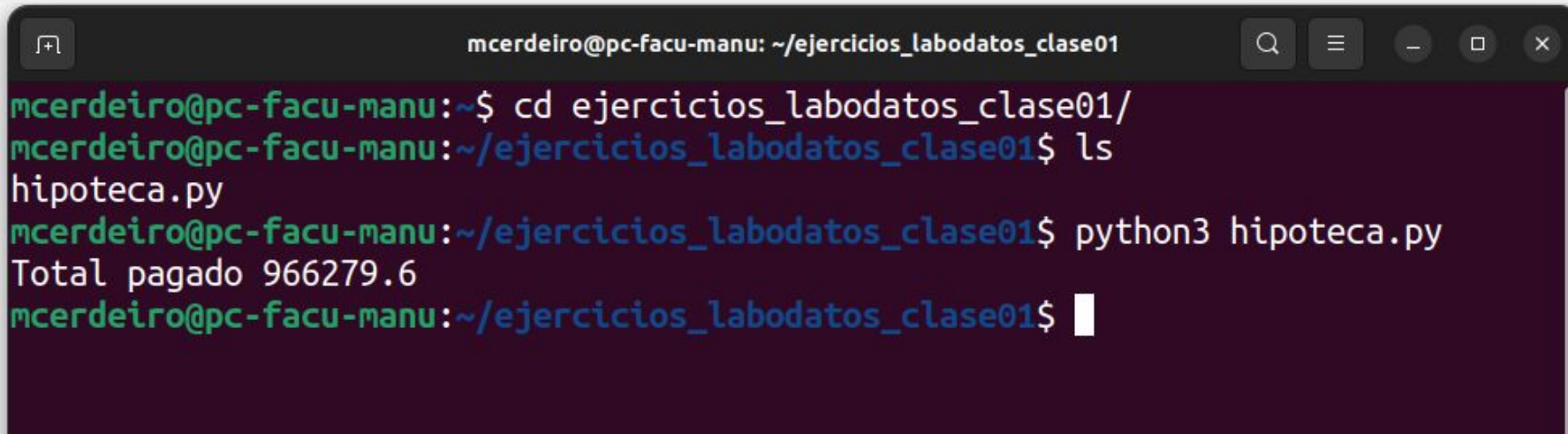
saldo = 500000.0
tasa = 0.05
pago_mensual = 2684.11
total_pagado = 0.0

while saldo > 0:
    saldo = saldo * (1+tasa/12) - pago_mensual
    total_pagado = total_pagado + pago_mensual

print('Total pagado', round(total_pagado, 2))
```

Uso de archivos .py

Ahora abrimos una terminal, nos situamos en el mismo directorio en el que está el archivo hipoteca.py y ejecutamos "python3 hipoteca.py"

A terminal window with a dark background and light-colored text. The window title bar shows the user 'mcerdeiro' on a machine 'pc-facu-manu' in the directory '~/ejercicios_labodatos_clase01'. The terminal content shows the user navigating to the directory, listing files to confirm 'hipoteca.py' is present, and then running 'python3 hipoteca.py'. The script outputs 'Total pagado 966279.6' before returning to the prompt.

```
mcerdeiro@pc-facu-manu: ~/ejercicios_labodatos_clase01
mcerdeiro@pc-facu-manu:~$ cd ejercicios_labodatos_clase01/
mcerdeiro@pc-facu-manu:~/ejercicios_labodatos_clase01$ ls
hipoteca.py
mcerdeiro@pc-facu-manu:~/ejercicios_labodatos_clase01$ python3 hipoteca.py
Total pagado 966279.6
mcerdeiro@pc-facu-manu:~/ejercicios_labodatos_clase01$
```

Uso de archivos .py

También podemos ejecutar con `-i` y quedarnos en el intérprete de python luego de la ejecución del programa. Las variables quedan en el estado final del programa.

```
mcerdeiro@pc-facu-manu: ~/ejercicios_labodatos_clase01
mcerdeiro@pc-facu-manu:~$ cd ejercicios_labodatos_clase01/
mcerdeiro@pc-facu-manu:~/ejercicios_labodatos_clase01$ ls
hipoteca.py
mcerdeiro@pc-facu-manu:~/ejercicios_labodatos_clase01$ python3 hipoteca.py
Total pagado 966279.6
mcerdeiro@pc-facu-manu:~/ejercicios_labodatos_clase01$ python3 -i hipoteca.py
Total pagado 966279.6
>>> saldo
-1.568757007813474
>>> total_pagado
966279.59999999957
>>> print(tasa)
0.05
>>> 
```

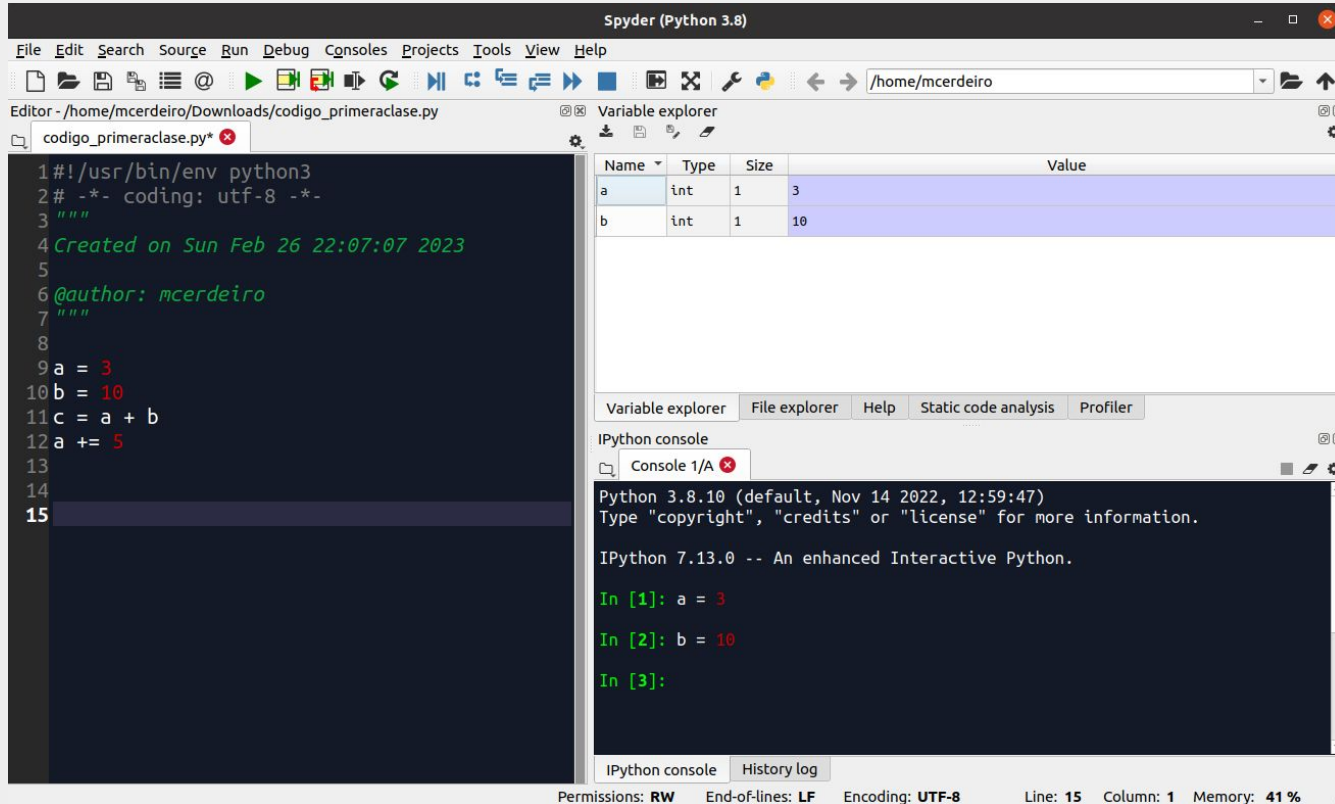
Spyder



Entorno de desarrollo integrado (*IDE*) - Spyder

- + código abierto (open source)
- + multiplataforma (sirve en linux, ios, windows...)
- + es cómodo para escribir código, ejecutarlo, corregirlo, probarlo y utilizarlo, en el mismo entorno
- + el editor de texto remarca palabras clave del lenguaje
- + tiene atajos (shortcut) útiles (y modificables a gusto de cada unx)

Entorno de desarrollo integrado (IDE) - Spyder



Entorno de desarrollo integrado (IDE) - Spyder

The image shows the Spyder Python IDE interface with several components and annotations:

- Editor:** The main window for editing code. It contains a file named `codigo_primeraclase.py`. The code is a Python script with a shebang, encoding declaration, docstring, and variable assignments.
- Variable explorer:** A panel on the right that displays the current state of variables in the program. It shows a table with columns for Name, Type, Size, and Value.
- IPython console:** A panel at the bottom right for running code and seeing the output. It shows the Python 3.8.10 prompt and the IPython 7.13.0 prompt.
- Annotations:** Four callout boxes with arrows pointing to specific parts of the interface:
 - nombre del programa:** Points to the file name `codigo_primeraclase.py` in the editor's title bar.
 - directorio actual de trabajo:** Points to the current working directory `/home/mcerdeiro` in the top right of the interface.
 - explorador de variables (muestra el estado del programa):** Points to the Variable explorer panel.
 - intérprete de python:** Points to the IPython console panel.

Code in the editor:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Sun Feb 26 22:07:07 2023
5
6@author: mcerdeiro
7"""
8
9a = 3
10b = 10
11c = a + b
12a += 5
13
14
15
```

Variable explorer table:

Name	Type	Size	Value
a	int	1	3
b	int	1	10

IPython console output:

```
Python 3.8.10 (default, Nov 14 2022, 12:59:47)
Type "copyright", "credits" or "license" for more information.

IPython 7.13.0 -- An enhanced Interactive Python.

In [1]: a = 3
In [2]: b = 10
In [3]:
```

Status bar: Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 15 Column: 1 Memory: 41 %

Funciones

Funciones

Las funciones son una herramienta para encapsular pedazos de código, facilitando su reutilización.

```
def sumar_le_uno(x):  
    res = x + 1  
    return res
```

```
def sumar(x,y):  
    res = x + y  
    return res
```

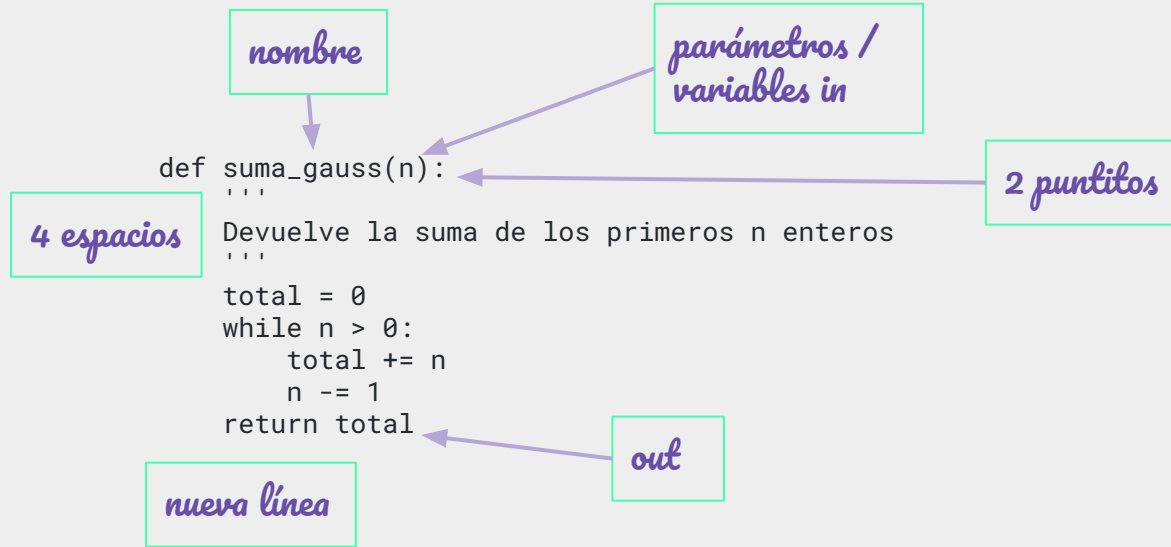
Funciones

Las funciones son una herramienta para encapsular pedazos de código, facilitando su reutilización.

```
def suma_gauss(n):  
    '''  
    Devuelve la suma de los primeros n enteros  
    '''  
    total = 0  
    while n > 0:  
        total += n  
        n -= 1  
    return total
```

Funciones

Definición de una función.



Funciones

Definición de una función.

```
def suma_gauss(n):  
    '''  
    Devuelve la suma de los primeros n enteros  
    '''  
    total = 0  
    while n > 0:  
        total += n  
        n -= 1  
    return total
```

Llamada (uso) de una función.

```
suma_gauss(10)  
  
x = 25  
suma_gaus(x)
```

Ejercicios

- ★ Definir una función **maximo(a, b)** que tome dos parámetros numéricos y devuelva el máximo.
- ★ Definir una función **dos_pertenece(lista)** que tome una lista y devuelva True si la lista tiene al 2 y False en caso contrario.
- ★ Definir una función **pertenece(lista, elem)** que tome una lista y un elemento y devuelva True si la lista tiene al elemento dado y False en caso contrario.
- ★ Definir una función **es_par(n)** que devuelva True si el número es par y False en caso contrario.
- ★ Definir una función **mas_larga(lista1, lista2)** que tome dos listas y devuelva la más larga.

Ejercicios

- ★ Definir una función **tachar_pares(lista)** que tome una lista de números y devuelva una similar pero donde los números pares estén reemplazados por 'x'.
- ★ Definir una función **cant_e** que tome una lista y devuelva la cantidad de letras 'e' que tiene.
- ★ Definir una función **sumar_unos** que tome una lista, les sume 1 a todos sus elementos, y devuelva la lista modificada.
- ★ Definir la función **mezclar(cadena1, cadena2)** que tome dos strings y devuelva el resultado de intercalar elemento a elemento. Por ejemplo: si intercalamos Pepe con Jose daría PJeopsee.

Funciones

A tener en cuenta:

Cuando una función involucra tipos de datos mutables, puede modificarlas (estén o no en el return).

Repasar en las funciones de los ejercicios previos,
¿cuáles modifican variables y cuáles no?

A veces esto es deseable y a veces NO.
¿Cómo lo podemos evitar?

Copias

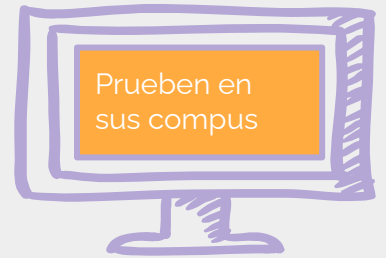
Las listas tienen un método para hacer copias.

```
a = [2,3,[100,101],4]
b = a.copy()
b == a # True
b is a # False
```


Probar en **pythontutor**

```
a = [2,3,[100,101],4]
b = a.copy()
b == a # True
b is a # False
```

```
a.append(5)
print(b)
```

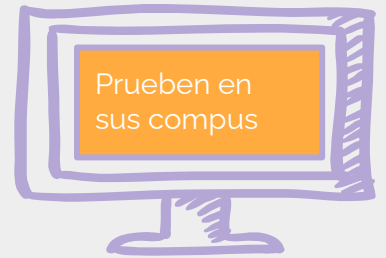


Probar en pythontutor

```
a = [2,3,[100,101],4]
b = a.copy()
b == a # True
b is a # False
```

```
a.append(5)
print(b)
```

```
a[2].append(102)
print(b)
```



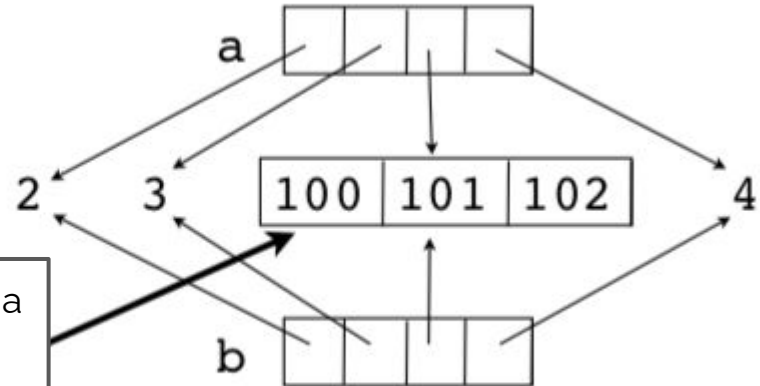
Probar en pythontutor

```
a = [2,3,[100,101],4]
b = a.copy()
b == a # True
b is a # False
```

```
a.append(5)
print(b)
```

```
a[2].append(102)
print(b)
```

Esta lista interna es la misma para a y b.



Deepcopy

```
import copy
```

```
a = [2,3,[100,101],4]
```

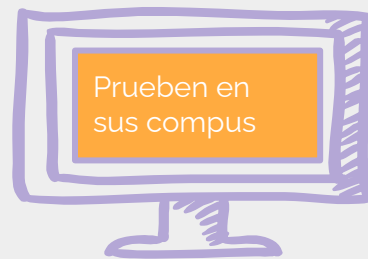
```
b = copy.deepcopy(a)
```

```
a.append(5)
```

```
print(b)
```

```
a[2].append(102)
```

```
print(b)
```



Ejercicio

Probar las funciones anteriores utilizando y no utilizando copias.

Diccionarios

Diccionarios

Los diccionarios son útiles si vamos a querer buscar rápidamente (por claves).

- Se construyen con llaves
- Cada entrada tiene una clave y un valor, separados por dos puntitos :
- Las entradas se separan con comas

```
{clave1: valor1, clave2: valor2, ... }
```

- Se acceden con corchetes indicando una clave
- Tanto las claves como los valores pueden ser de distintos tipos de objetos
- Las claves deben ser de tipo inmutable

Ejemplo

```
dias_engl = {'lunes': 'monday', 'martes': 'tuesday', 'miércoles': 'wednesday', 'jueves':  
'thursday'}
```

```
>>> dias_engl['lunes']  
'monday'
```

```
>>> dias_engl['viernes']  
Traceback (most recent call last):
```

```
  File "<ipython-input-33-ee0fa133453b>", line 1, in <module>  
    dias_engl['viernes']
```

```
KeyError: 'viernes'
```

```
>>> dias_engl['viernes'] = 'friday'      # agrego la entrada  
>>> dias_engl['viernes']  
'friday'
```


Ejemplo

También se pueden armar y modificar agregando entradas:

```
feriados = {} # Empezamos con un diccionario vacío
```

```
# Agregamos elementos
```

```
feriados[(1, 1)] = 'Año nuevo'
```

```
feriados[(1, 5)] = 'Día del trabajador'
```

```
feriados[(13, 9)] = 'Día del programador'
```

Diccionarios

También se pueden armar a partir de una lista de tuplas (clave, valor).

```
>>> cuadrados = dict([(1,1), (2,4), (3,9), (4,16)])  
>>> cuadrados[2]  
4
```

La función **zip** genera tuplas a partir de dos listas:

```
>>> for t in zip([1,2,3,4],[1,4,9,16]):  
        print(t)  
(1, 1)  
(2, 4)  
(3, 9)  
(4, 15)
```

Es decir que podemos construir el diccionario a partir de dos listas (claves y valores);

```
>>> cuadrados = dict(zip([1,2,3,4],[1,4,9,16]))
```

Ejercicio

Construí una función `traductor_geringoso(lista)` que, a partir de una lista de palabras, devuelva un diccionario geringoso.

Las claves del diccionario deben ser las palabras de la lista y los valores deben ser sus traducciones al geringoso.

Por ejemplo:

```
>>> lista = ['banana', 'manzana', 'mandarina']  
>>> traductor_geringoso(lista)
```

```
{'banana': 'bapanapanapa', 'manzana': 'mapanzapanapa', 'mandarina': 'mapandaparipinapa'}
```

Módulos

Módulos

Si bien Python tiene muchas funciones que se pueden usar directamente, hay muchas otras que están disponibles dentro de módulos.

Un **módulo** es una **colección de funciones** que alguien (o una comunidad) desarrollaron y empaquetaron para que estén disponibles para todo el mundo.

Para que las funciones estén disponibles para ser utilizadas en mi programa, tengo que usar la instrucción **import**.

Módulos

Si quiero generar números aleatorios, que están en el módulo random, tengo que escribir:

```
import random
random.seed(COMPLETAR)
prueba = random.random()
print(prueba)
prueba = random.random()
print(prueba)
prueba = random.random()
print(prueba)
```

Módulos

Módulo math tiene funciones matemáticas.

```
import math
```

```
math.sqrt(2)
```

```
math.exp(x)
```

```
math.cos(x)
```

```
math.gcd(15, 12)
```

Ejercicio

Escribir una función `general_tirar()` que simule una tirada de dados para el juego de la generala. Es decir, debe devolver una lista aleatoria de 5 valores de dados. Por ejemplo `[2,3,2,1,6]`.

Archivos

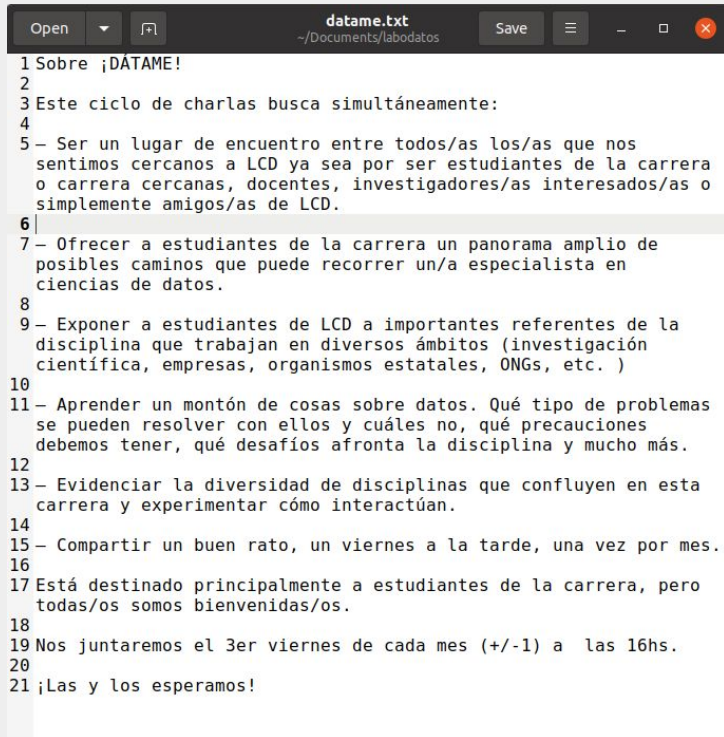
Archivos

Frecuentemente vamos a utilizar una fuente de datos, que en muchos casos va a estar en un archivo.

Tenemos que poder manejar archivos: leer, crear, modificar, guardar archivos de distintos tipos.

```
f = open(nombre_archivo, 'rt' )      # abrir para lectura ('r' de read, 't' de text)
data = f.read()
f.close()
data
print(data)
```

Ejemplo



```
1 Sobre ¡DATAME!  
2  
3 Este ciclo de charlas busca simultáneamente:  
4  
5 - Ser un lugar de encuentro entre todos/as los/as que nos  
sentimos cercanos a LCD ya sea por ser estudiantes de la carrera  
o carrera cercanas, docentes, investigadores/as interesados/as o  
simplemente amigos/as de LCD.  
6 |  
7 - Ofrecer a estudiantes de la carrera un panorama amplio de  
posibles caminos que puede recorrer un/a especialista en  
ciencias de datos.  
8  
9 - Exponer a estudiantes de LCD a importantes referentes de la  
disciplina que trabajan en diversos ámbitos (investigación  
científica, empresas, organismos estatales, ONGs, etc. )  
10  
11 - Aprender un montón de cosas sobre datos. Qué tipo de problemas  
se pueden resolver con ellos y cuáles no, qué precauciones  
debemos tener, qué desafíos afronta la disciplina y mucho más.  
12  
13 - Evidenciar la diversidad de disciplinas que confluyen en esta  
carrera y experimentar cómo interactúan.  
14  
15 - Compartir un buen rato, un viernes a la tarde, una vez por mes.  
16  
17 Está destinado principalmente a estudiantes de la carrera, pero  
todas/os somos bienvenidas/os.  
18  
19 Nos juntaremos el 3er viernes de cada mes (+/-1) a las 16hs.  
20  
21 ¡Las y los esperamos!
```

Ejemplo:
nombre_archivo = 'datame.txt'

Archivos

```
with open(nombre_archivo, 'rt') as file:    # otra forma de abrir archivos
    data = file.read()
    # 'data' es una cadena con todo el texto en el archivo

data
print(data)
```

Para leer una archivo línea por línea, usá un ciclo for como éste:

```
with open(nombre_archivo, 'rt') as file:
    for line in file:
        # Procesar la línea
```

Python tiene dos modos de salida. En el primero, escribimos data en el intérprete y Python muestra la representación **cruda** de la cadena, incluyendo comillas y códigos de escape (\n). Cuando escribimos print(data), en cambio, se imprime la salida **formateada** de la cadena.

Ejercicio

Escribir un programa que recorra las líneas del archivo 'datame.txt' e imprima solamente las líneas que contienen la palabra 'estudiante'.

Archivos .csv

csv = comma separated values

- son “planillas” guardadas como texto
- cada línea de texto es una fila de la planilla
- las comas separan las columnas

a,b,c
d,e,f
x,y,z
u,v,w

a	b	c
d	e	f
x	y	z
u	v	w

Archivos .csv

csv = comma separated values

Ejemplo:
nombre_archivo =
'cronograma_sugerido.csv'

Cuatrimestre	Asignatura	Correlatividad de Asignaturas
3	Álgebra I	CBC
3	Algoritmos y Estructuras de Datos I	CBC
4	Análisis I	CBC
4	Electiva de Introducción a las Ciencias Naturales	CBC
5	Análisis II	Análisis I
5	Álgebra Lineal Computacional	Álgebra I – Algoritmos y Estructuras de Datos I
5	Laboratorio de Datos	Algoritmos y Estructuras de Datos I
6	Análisis Avanzado	Análisis II, Álgebra I
6	Algoritmos y Estructuras de Datos II	Algoritmos y Estructuras de Datos I
7	Probabilidad	Análisis Avanzado
7	Algoritmos y Estructura de Datos III	Algoritmos y Estructuras de Datos II
8	Intr. a la Estadística y Ciencia de Datos	Lab de Datos, Probabilidad, Álgebra Lineal Computacional
8	Intr. a la Investigación Operativa y Optimización	Alg y Estruct de Datos III, Análisis II, Álgebra Lineal Computacional
8	Intr. al Modelado Continuo.	Análisis Avanzado, Álgebra Lineal Computacional, Alg y Estructura de Datos II

Archivos .csv

```
nombre_archivo = 'cronograma_sugerido.csv'  
with open(nombre_archivo, 'rt') as file:  
    for line in file:  
        datos_linea = line.split(',')  
        print(datos_linea[1])
```

¿Cómo podemos armar una lista con todas las materias del cronograma?

Ejercicio

- ★ Utilizando el archivo `cronograma_sugerido`, armar una lista de las materias del cronograma, llamada `lista_materias`.
- ★ Luego, definir una función `cuantas_materias(n)` que, dado un número de cuatrimestre (n entre 3 y 8), devuelva la cantidad de materias a cursar en ese cuatrimestre.
Por ejemplo:

```
>>> cuantas_materias(5)
```

```
3
```

```
>>> cuantas_materias(7)
```

```
10
```

Módulo csv

Es útil para trabajar con archivos .csv

```
f = open(nombre_archivo)
filas = csv.reader(f)
for fila in filas:
    instrucciones
```

Acá **filas** es un iterador.

```
f.close()
```

Si la primera fila son encabezados, podemos guardarlos así:

```
f = open(nombre_archivo)
filas = csv.reader(f)
encabezado = next(filas) # un paso del iterador
for fila in filas:        # ahora el iterador sigue desde la segunda fila
    instrucciones

f.close()
```

Ejemplo

Definimos `registros(nombre_archivo)` que recorre el archivo indicado, conteniendo información de un cronograma sugerido de cursada, y devuelve la información como una lista de diccionarios. Las claves de los diccionarios son las columnas del archivo, y los valores son las entradas de cada fila para esa columna.

```
def registros(nombre_archivo):  
    lista = []  
    with open(nombre_archivo, 'rt') as f:  
        filas = csv.reader(f)  
        encabezado = next(filas)  
        for fila in filas:  
            registro = dict(zip(encabezado, fila)) # armo el diccionario de cada fila  
            lista.append(registro)                # lo agrego a la lista  
    return lista
```

Ejercicio

Definir una función `materias_cuatrimestre(nombre_archivo, n)` que recorra el archivo indicado, conteniendo información de un cronograma sugerido de cursada, y devuelva una lista de diccionarios con la información de las materias sugeridas para cursar el n-ésimo cuatrimestre.

Debe funcionar así:

```
materias_cuatrimestre('cronograma_sugerido.csv', 3):
```

```
[{'Cuatrimestre': '3',  
  'Asignatura': 'Álgebra I',  
  'Correlatividad de Asignaturas': 'CBC'},  
 {'Cuatrimestre': '3',  
  'Asignatura': 'Algoritmos y Estructuras de Datos I',  
  'Correlatividad de Asignaturas': 'CBC'}]
```

1 ÁRBOL

CADA

7 PERSONAS



Datos de árboles en CABA

HAY 372.687
ÁRBOLES
EN LAS CALLES DE
BUENOS AIRES



**MAPA
DE
DENSIDAD**

*PARQUE CHAS EL BARRIO
CON MAYOR DENSIDAD



ESPECIES MÁS FRECUENTES

Fresno americano	39%	141820	
Plátano	9%	34781	
Paraíso	7%	24558	
Ficus	7%	24076	
Tilo	5%	17477	
Jacarandá	3%	11044	

Información basada en el censo de árboles de alineación 2011
realizado por el Gobierno de la Ciudad de Buenos Aires.
La información pudo sufrir variaciones en periodos posteriores.



Buenos Aires Ciudad

EN TODO ESTÁS VOS

Datos de arbolado porteño

Título de la columna	Tipo de dato	Descripción
long	Número flotante (float)	Coordenadas para geolocalización
lat	Número flotante (float)	Coordenadas para geolocalización
id_arbol	Número entero (integer)	Identificador único del árbol
altura_tot	Número entero (integer)	Altura del árbol (m)
diametro	Número entero (integer)	Diámetro del árbol (cm)
inclinacio	Número entero (integer)	Inclinación del árbol (grados)
id_especie	Número entero (integer)	Identificador de la especie
nombre_com	Texto (string)	Nombre común del árbol
nombre_cie	Texto (string)	Nombre científico del árbol
tipo_folla	Texto (string)	Tipo de follaje del árbol
espacio_ve	Texto (string)	Nombre del espacio verde
ubicacion	Texto (string)	Dirección del espacio verde
nombre_fam	Texto (string)	Nombre de la familia del árbol
nombre_gen	Texto (string)	Nombre del género del árbol
origen	Texto (string)	Origen del árbol
coord_x	Número flotante (float)	Coordenadas para localización
coord_y	Número flotante (float)	Coordenadas para localización

Ejercicios para entregar

Guía de arbolado porteño en pdf

- Entrega en grupos de 3
- Fecha límite jueves 30/3

