

# Introducción a la programación

## Usando Python

G. Sebastián Pedersen

Instituto de Industria  
Universidad Nacional de General Sarmiento

Matemática para Economistas III, 1er. cuat. 2019

# Introducción

## Info general:

G. Sebastián Pedersen – [sebasped@gmail.com](mailto:sebasped@gmail.com)

Página del curso: <https://sebasped.github.io/pythonungs/>

# Introducción

## Info general:

G. Sebastián Pedersen – sebasped@gmail.com

Página del curso: <https://sebasped.github.io/pythonungs/>

- ▶ Objetivo: programar un sistema de EDOs y graficarlo.

# Introducción

## Info general:

G. Sebastián Pedersen – sebasped@gmail.com

Página del curso: <https://sebasped.github.io/pythonungs/>

- ▶ Objetivo: programar un sistema de EDOs y graficarlo.
- ▶ ¿Qué es programar?

# Introducción

## Info general:

G. Sebastián Pedersen – sebasped@gmail.com

Página del curso: <https://sebasped.github.io/pythonungs/>

- ▶ Objetivo: programar un sistema de EDOs y graficarlo.
- ▶ ¿Qué es programar?
  - ▶ Programar  $\neq$  saber un lenguaje de programación.

# Introducción

## Info general:

G. Sebastián Pedersen – sebasped@gmail.com

Página del curso: <https://sebasped.github.io/pythonungs/>

- ▶ Objetivo: programar un sistema de EDOs y graficarlo.
- ▶ ¿Qué es programar?
  - ▶ Programar  $\neq$  saber un lenguaje de programación.
  - ▶ Programar  $\neq$  saber usar una computadora.

# Introducción

## Info general:

G. Sebastián Pedersen – sebasped@gmail.com

Página del curso: <https://sebasped.github.io/pythonungs/>

- ▶ Objetivo: programar un sistema de EDOs y graficarlo.
- ▶ ¿Qué es programar?
  - ▶ Programar  $\neq$  saber un lenguaje de programación.
  - ▶ Programar  $\neq$  saber usar una computadora.
  - ▶ Frase de Edgar Dijkstra: “La Ciencia de la Computación no tiene que ver con las computadoras más que la Astronomía con los telescopios”.

# Introducción

## Info general:

G. Sebastián Pedersen – sebasped@gmail.com

Página del curso: <https://sebasped.github.io/pythonungs/>

- ▶ Objetivo: programar un sistema de EDOs y graficarlo.
- ▶ ¿Qué es programar?
  - ▶ Programar  $\neq$  saber un lenguaje de programación.
  - ▶ Programar  $\neq$  saber usar una computadora.
  - ▶ Frase de Edgar Dijkstra: “La Ciencia de la Computación no tiene que ver con las computadoras más que la Astronomía con los telescopios”.
- ▶ ¿Qué lenguajes de programación conocen?



# Introducción

## Info general:

G. Sebastián Pedersen – [sebasped@gmail.com](mailto:sebasped@gmail.com)

Página del curso: <https://sebasped.github.io/pythonungs/>

- ▶ Objetivo: programar un sistema de EDOs y graficarlo.
- ▶ ¿Qué es programar?
  - ▶ Programar  $\neq$  saber un lenguaje de programación.
  - ▶ Programar  $\neq$  saber usar una computadora.
  - ▶ Frase de Edgar Dijkstra: “La Ciencia de la Computación no tiene que ver con las computadoras más que la Astronomía con los telescopios”.
- ▶ ¿Qué lenguajes de programación conocen?
- ▶ ¿Cuál lenguaje vamos a usar nosotros?

# Introducción

## Info general:

G. Sebastián Pedersen – sebasped@gmail.com

Página del curso: <https://sebasped.github.io/pythonungs/>

- ▶ Objetivo: programar un sistema de EDOs y graficarlo.
- ▶ ¿Qué es programar?
  - ▶ Programar  $\neq$  saber un lenguaje de programación.
  - ▶ Programar  $\neq$  saber usar una computadora.
  - ▶ Frase de Edgar Dijkstra: “La Ciencia de la Computación no tiene que ver con las computadoras más que la Astronomía con los telescopios”.
- ▶ ¿Qué lenguajes de programación conocen?
- ▶ ¿Cuál lenguaje vamos a usar nosotros?
  - ▶ Respuesta corta: Python.

# Introducción

## Info general:

G. Sebastián Pedersen – sebasped@gmail.com

Página del curso: <https://sebasped.github.io/pythonungs/>

- ▶ Objetivo: programar un sistema de EDOs y graficarlo.
- ▶ ¿Qué es programar?
  - ▶ Programar  $\neq$  saber un lenguaje de programación.
  - ▶ Programar  $\neq$  saber usar una computadora.
  - ▶ Frase de Edgar Dijkstra: “La Ciencia de la Computación no tiene que ver con las computadoras más que la Astronomía con los telescopios”.
- ▶ ¿Qué lenguajes de programación conocen?
- ▶ ¿Cuál lenguaje vamos a usar nosotros?
  - ▶ Respuesta corta: Python.
  - ▶ Respuesta larga: no importa demasiado. Lo importante son los conocimientos básicos de programación, que son comunes a la mayoría de los lenguajes.

# Recursos Python

(Están también en la página <https://sebasped.github.io/pythonungs/>)

- ▶ Comunidades:

- ▶ <http://www.python.org.ar/>
- ▶ <https://argentinaenpython.com/>
- ▶ <https://twitter.com/pibesdesistemas>
- ▶ <https://www.chicasentecnologia.org/>
- ▶ <https://twitter.com/lasdesistemas>
- ▶ <https://www.meetup.com/Buenos-Aires-Python-Meetup/>
- ▶ <https://twitter.com/linuxchixar>

# Recursos Python

(Están también en la página <https://sebasped.github.io/pythonungs/>)

- ▶ Comunidades:

- ▶ <http://www.python.org.ar/>
- ▶ <https://argentinaenpython.com/>
- ▶ <https://twitter.com/pibesdesistemas>
- ▶ <https://www.chicasentecnologia.org/>
- ▶ <https://twitter.com/lasdesistemas>
- ▶ <https://www.meetup.com/Buenos-Aires-Python-Meetup/>
- ▶ <https://twitter.com/linuxchixar>

- ▶ Material, cursos, tutoriales, bibliografía:

- ▶ Tutorial de Python para no programadores: [http://jjc.freeshell.org/easytut/easytut\\_es/easytut.html](http://jjc.freeshell.org/easytut/easytut_es/easytut.html)
- ▶ <http://www.python.org.ar/wiki/AprendiendoPython>
- ▶ <https://argentinaenpython.com/quiero-aprender-python/aprenda-a-pensar-como-un-programador-con-python.pdf>
- ▶ <https://launchpadlibrarian.net/18980633/Python%20para%20todos.pdf>
- ▶ Cursos online (en inglés): coursera, datacamp, udemy, Stanford online, edx, codeacademy, Harvard online, etc.

# Recursos Python

(Están también en la página <https://sebasped.github.io/pythonungs/>)

- ▶ Comunidades:

- ▶ <http://www.python.org.ar/>
- ▶ <https://argentinaenpython.com/>
- ▶ <https://twitter.com/pibesdesistemas>
- ▶ <https://www.chicasentecnologia.org/>
- ▶ <https://twitter.com/lasdesistemas>
- ▶ <https://www.meetup.com/Buenos-Aires-Python-Meetup/>
- ▶ <https://twitter.com/linuxchixar>

- ▶ Material, cursos, tutoriales, bibliografía:

- ▶ Tutorial de Python para no programadores: [http://jjc.freeshell.org/easytut/easytut\\_es/easytut.html](http://jjc.freeshell.org/easytut/easytut_es/easytut.html)
- ▶ <http://www.python.org.ar/wiki/AprendiendoPython>
- ▶ <https://argentinaenpython.com/quiero-aprender-python/aprenda-a-pensar-como-un-programador-con-python.pdf>
- ▶ <https://launchpadlibrarian.net/18980633/Python%20para%20todos.pdf>
- ▶ Cursos online (en inglés): coursera, datacamp, udemy, Stanford online, edx, codeacademy, Harvard online, etc.

- ▶ Buscar en internet: hay mucho mucho hecho ya.

# Python actualmente es muy popular

## Índice Tiobe:

<https://www.tiobe.com/tiobe-index/>

### TIOBE Index for March 2019

#### March Headline: Powershell enters the TIOBE index top 50

There are hardly any interesting changes in the TIOBE index this month. We had to use our magnifying glass to spot some news, i.e. that Powershell entered the TIOBE index top 50. The Powershell scripting language is more than 12 years old and it has been in the top 50 before. Powershell is typically used for basic scripting. Until recently it was only available for Windows but Microsoft used its .NET Core platform to create Powershell Core. This version is open source and runs on all major platforms. This might be the reason why Powershell is getting more popular again.

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found [here](#).

Mar 2019	Mar 2018	Change	Programming Language	Ratings	Change
1	1		Java	14.880%	-0.06%
2	2		C	13.305%	+0.55%
3	4	▲	Python	8.262%	+2.39%
4	3	▼	C++	8.126%	+1.67%
5	6	▲	Visual Basic .NET	6.429%	+2.34%
6	5	▼	C#	3.267%	-1.80%
7	8	▲	JavaScript	2.426%	-1.49%
8	7	▼	PHP	2.420%	-1.59%
9	10	▲	SQL	1.926%	-0.76%
10	14	▲	Objective-C	1.681%	-0.09%

# Elementos básicos programación

- ▶ Diferencia entre algoritmo y programa.



# Elementos básicos programación

- ▶ Diferencia entre algoritmo y programa.
- ▶ Herramientas esenciales:
  - ▶ Tipos de datos: enteros, reales, strings, etc.
  - ▶ Variables y expresiones.
  - ▶ Instrucciones: asignación, condicional, ciclo.
  - ▶ Funciones, pasajes de parámetros.

# Elementos básicos programación

- ▶ Diferencia entre algoritmo y programa.
- ▶ Herramientas esenciales:
  - ▶ Tipos de datos: enteros, reales, strings, etc.
  - ▶ Variables y expresiones.
  - ▶ Instrucciones: asignación, condicional, ciclo.
  - ▶ Funciones, pasajes de parámetros.
- ▶ Estructuras de datos:
  - ▶ Listas, arreglos.
  - ▶ Conjuntos, diccionarios.
  - ▶ Pilas, colas.

# Elementos básicos programación

- ▶ Diferencia entre algoritmo y programa.
- ▶ Herramientas esenciales:
  - ▶ Tipos de datos: enteros, reales, strings, etc.
  - ▶ Variables y expresiones.
  - ▶ Instrucciones: asignación, condicional, ciclo.
  - ▶ Funciones, pasajes de parámetros.
- ▶ Estructuras de datos:
  - ▶ Listas, arreglos.
  - ▶ Conjuntos, diccionarios.
  - ▶ Pilas, colas.
- ▶ ¿Cómo se aprende a programar?

# Elementos básicos programación

- ▶ Diferencia entre algoritmo y programa.
- ▶ Herramientas esenciales:
  - ▶ Tipos de datos: enteros, reales, strings, etc.
  - ▶ Variables y expresiones.
  - ▶ Instrucciones: asignación, condicional, ciclo.
  - ▶ Funciones, pasajes de parámetros.
- ▶ Estructuras de datos:
  - ▶ Listas, arreglos.
  - ▶ Conjuntos, diccionarios.
  - ▶ Pilas, colas.
- ▶ ¿Cómo se aprende a programar? Programando... no hay manera de aprender algo sin hacerlo.

## Algoritmo y programa

- ▶ Un *algoritmo* es una secuencia de instrucciones. Por ejemplo:

# Algoritmo y programa

- ▶ Un *algoritmo* es una secuencia de instrucciones. Por ejemplo:
  1. Moje el cabello.
  2. Coloque champú.
  3. Masajee suavemente y deje actuar por 2 min.
  4. Enjuague.
  5. Repita el procedimiento desde 1.

# Algoritmo y programa

- ▶ Un *algoritmo* es una secuencia de instrucciones. Por ejemplo:
  1. Moje el cabello.
  2. Coloque champú.
  3. Masajee suavemente y deje actuar por 2 min.
  4. Enjuague.
  5. Repita el procedimiento desde 1.
- ▶ Un *programa* es una implementación de un algoritmo en un lenguaje de programación.

# Algoritmo y programa

- ▶ Un *algoritmo* es una secuencia de instrucciones. Por ejemplo:
  1. Moje el cabello.
  2. Coloque champú.
  3. Masajee suavemente y deje actuar por 2 min.
  4. Enjuague.
  5. Repita el procedimiento desde 1.
- ▶ Un *programa* es una implementación de un algoritmo en un lenguaje de programación.
  - ▶ El programa representa al algoritmo en el lenguaje.



# Algoritmo y programa

- ▶ Un *algoritmo* es una secuencia de instrucciones. Por ejemplo:
  1. Moje el cabello.
  2. Coloque champú.
  3. Masajee suavemente y deje actuar por 2 min.
  4. Enjuague.
  5. Repita el procedimiento desde 1.
- ▶ Un *programa* es una implementación de un algoritmo en un lenguaje de programación.
  - ▶ El programa representa al algoritmo en el lenguaje.
  - ▶ Las instrucciones son propias del lenguaje.

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.scre
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ltdate = response.data[0]['created_at']
        ltdate2 = datetime.strptime(ltdate, '%a %b %d %H:%M:%S +0000 %Y')
        today = datetime.now()
        howlong = (today-ltdate2).days
        if howlong < daywindow:
            print i.screen_name, 'has tweeted in the past', daywindow,
            totaltweets += len(response.data)
            for j in response.data:
                if j.entities.urls:
                    for k in j.entities.urls:
                        newurl = k['expanded_url']
                        urlset.add((newurl, j.user.screen_name))
        else:
            print i.screen_name, 'has not tweeted in the past', daywindi
```

# Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo entero (int).

# Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo entero (int).
- ▶ 2.5 es un valor de tipo número “real” (float).

# Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo entero (int).
- ▶ 2.5 es un valor de tipo número “real” (float).
- ▶ “hola” es un valor de tipo caracter (string).

# Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo entero (int).
- ▶ 2.5 es un valor de tipo número “real” (float).
- ▶ “hola” es un valor de tipo caracter (string).
- ▶ False es un valor de tipo booleano (bool).

# Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo entero (int).
- ▶ 2.5 es un valor de tipo número “real” (float).
- ▶ “hola” es un valor de tipo caracter (string).
- ▶ False es un valor de tipo booleano (bool).

Los operaciones son, por ejemplo:

- ▶ Suma/Resta:  $3+4 \rightarrow 7$ .

# Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo entero (int).
- ▶ 2.5 es un valor de tipo número “real” (float).
- ▶ “hola” es un valor de tipo caracter (string).
- ▶ False es un valor de tipo booleano (bool).

Los operaciones son, por ejemplo:

- ▶ Suma/Resta:  $3+4 \rightarrow 7$ .
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.

# Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo entero (int).
- ▶ 2.5 es un valor de tipo número “real” (float).
- ▶ “hola” es un valor de tipo caracter (string).
- ▶ False es un valor de tipo booleano (bool).

Los operaciones son, por ejemplo:

- ▶ Suma/Resta:  $3+4 \rightarrow 7$ .
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.
- ▶ Producto:  $2*8 \rightarrow 16$ .



# Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo entero (int).
- ▶ 2.5 es un valor de tipo número “real” (float).
- ▶ “hola” es un valor de tipo caracter (string).
- ▶ False es un valor de tipo booleano (bool).

Los operaciones son, por ejemplo:

- ▶ Suma/Resta:  $3+4 \rightarrow 7$ .
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.
- ▶ Producto:  $2*8 \rightarrow 16$ .
- ▶ División:  $5/2 \rightarrow$

# Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo entero (int).
- ▶ 2.5 es un valor de tipo número “real” (float).
- ▶ “hola” es un valor de tipo caracter (string).
- ▶ False es un valor de tipo booleano (bool).

Los operaciones son, por ejemplo:

- ▶ Suma/Resta:  $3+4 \rightarrow 7$ .
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.
- ▶ Producto:  $2*8 \rightarrow 16$ .
- ▶ División:  $5/2 \rightarrow 2$ . **Ojo, por defecto es división entera.**

# Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo entero (int).
- ▶ 2.5 es un valor de tipo número “real” (float).
- ▶ “hola” es un valor de tipo caracter (string).
- ▶ False es un valor de tipo booleano (bool).

Los operaciones son, por ejemplo:

- ▶ Suma/Resta:  $3+4 \rightarrow 7$ .
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.
- ▶ Producto:  $2*8 \rightarrow 16$ .
- ▶ División:  $5/2 \rightarrow 2$ . **Ojo, por defecto es división entera.**
- ▶ División “común”:  $5/2.0$  o  $5.0/2$  o  $5.0/2.0$  (i.e. que alguno sea un float).

# Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo entero (int).
- ▶ 2.5 es un valor de tipo número “real” (float).
- ▶ “hola” es un valor de tipo caracter (string).
- ▶ False es un valor de tipo booleano (bool).

Las operaciones son, por ejemplo:

- ▶ Suma/Resta:  $3+4 \rightarrow 7$ .
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.
- ▶ Producto:  $2*8 \rightarrow 16$ .
- ▶ División:  $5/2 \rightarrow 2$ . **Ojo, por defecto es división entera.**
- ▶ División “común”:  $5/2.0$  o  $5.0/2$  o  $5.0/2.0$  (i.e. que alguno sea un float).
- ▶ Resto:  $5 \% 2 \rightarrow 1$ .

# Tipos de Datos y Comparaciones

- ▶ Igualdad:  $i == k$ .

# Tipos de Datos y Comparaciones

- ▶ Igualdad:  $i == k$ .
  - ▶ Probar  $2 == 3$ ,  $4 == 4$ ,  $'a' == 'a'$ .

# Tipos de Datos y Comparaciones

- ▶ Igualdad:  $i == k$ .
  - ▶ Probar  $2 == 3$ ,  $4 == 4$ ,  $'a' == 'a'$ .
- ▶ Distinto:  $i != k$ .

# Tipos de Datos y Comparaciones

- ▶ Igualdad:  $i == k$ .
  - ▶ Probar  $2 == 3$ ,  $4 == 4$ ,  $'a' == 'a'$ .
- ▶ Distinto:  $i != k$ .
  - ▶ Probar  $2 != 3$ .



# Tipos de Datos y Comparaciones

- ▶ Igualdad:  $i == k$ .
  - ▶ Probar  $2 == 3$ ,  $4 == 4$ ,  $'a' == 'a'$ .
- ▶ Distinto:  $i != k$ .
  - ▶ Probar  $2 != 3$ .
- ▶ Menor:  $i < k$ .

# Tipos de Datos y Comparaciones

- ▶ Igualdad:  $i == k$ .
  - ▶ Probar  $2 == 3$ ,  $4 == 4$ ,  $'a' == 'a'$ .
- ▶ Distinto:  $i != k$ .
  - ▶ Probar  $2 != 3$ .
- ▶ Menor:  $i < k$ .
- ▶ Mayor:  $i > k$ .

# Tipos de Datos y Comparaciones

- ▶ Igualdad:  $i == k$ .
  - ▶ Probar  $2 == 3$ ,  $4 == 4$ ,  $'a' == 'a'$ .
- ▶ Distinto:  $i != k$ .
  - ▶ Probar  $2 != 3$ .
- ▶ Menor:  $i < k$ .
- ▶ Mayor:  $i > k$ .
- ▶ Menor o igual:  $i <= k$ .

# Tipos de Datos y Comparaciones

- ▶ Igualdad:  $i == k$ .
  - ▶ Probar  $2 == 3$ ,  $4 == 4$ ,  $'a' == 'a'$ .
- ▶ Distinto:  $i != k$ .
  - ▶ Probar  $2 != 3$ .
- ▶ Menor:  $i < k$ .
- ▶ Mayor:  $i > k$ .
- ▶ Menor o igual:  $i \leq k$ .
- ▶ Mayor o igual:  $i \geq k$ .

## Tipos de Datos: Listas

Una *lista* es una colección de valores que se acceden mediante un índice:

45	657	6756	4	23	5	324	2344
0	1	2	3	4	5	6	7

Ojo, el primer elemento tiene índice 0.

## Tipos de Datos: Listas

Una *lista* es una colección de valores que se acceden mediante un índice:

45	657	6756	4	23	5	324	2344
0	1	2	3	4	5	6	7

Ojo, el primer elemento tiene índice 0.

Algunos ejemplos de listas y operaciones:

► [2, 3, 5]

## Tipos de Datos: Listas

Una *lista* es una colección de valores que se acceden mediante un índice:

45	657	6756	4	23	5	324	2344
0	1	2	3	4	5	6	7

Ojo, el primer elemento tiene índice 0.

Algunos ejemplos de listas y operaciones:

- $[2, 3, 5]$  ¿Cuál es el elemento con índice 1?

## Tipos de Datos: Listas

Una *lista* es una colección de valores que se acceden mediante un índice:

45	657	6756	4	23	5	324	2344
0	1	2	3	4	5	6	7

Ojo, el primer elemento tiene índice 0.

Algunos ejemplos de listas y operaciones:

- ▶ `[2, 3, 5]` ¿Cuál es el elemento con índice 1?
- ▶ `[2, 3.5, 'patito', 8]`.



## Tipos de Datos: Listas

Una *lista* es una colección de valores que se acceden mediante un índice:

45	657	6756	4	23	5	324	2344
0	1	2	3	4	5	6	7

Ojo, el primer elemento tiene índice 0.

Algunos ejemplos de listas y operaciones:

- ▶ [2, 3, 5] ¿Cuál es el elemento con índice 1?
- ▶ [2, 3.5, 'patito', 8].
- ▶ [ ] lista vacía.

## Tipos de Datos: Listas

Una *lista* es una colección de valores que se acceden mediante un índice:

45	657	6756	4	23	5	324	2344
0	1	2	3	4	5	6	7

Ojo, el primer elemento tiene índice 0.

Algunos ejemplos de listas y operaciones:

- ▶ `[2, 3, 5]` ¿Cuál es el elemento con índice 1?
- ▶ `[2, 3.5, 'patito', 8]`.
- ▶ `[]` lista vacía.
- ▶ `c = [2, 3, 5]` define una lista de nombre `c`.
- ▶ `c[i]` accede o devuelve el elemento con índice `i` de la lista.

## Tipos de Datos: Listas

Una *lista* es una colección de valores que se acceden mediante un índice:

45	657	6756	4	23	5	324	2344
0	1	2	3	4	5	6	7

Ojo, el primer elemento tiene índice 0.

Algunos ejemplos de listas y operaciones:

- ▶ `[2, 3, 5]` ¿Cuál es el elemento con índice 1?
- ▶ `[2, 3.5, 'patito', 8]`.
- ▶ `[]` lista vacía.
- ▶ `c = [2, 3, 5]` define una lista de nombre `c`.
- ▶ `c[i]` accede o devuelve el elemento con índice `i` de la lista.
- ▶ `len(c)` devuelve la longitud de la lista.

## Tipos de Datos: Listas

Una *lista* es una colección de valores que se acceden mediante un índice:

45	657	6756	4	23	5	324	2344
0	1	2	3	4	5	6	7

Ojo, el primer elemento tiene índice 0.

Algunos ejemplos de listas y operaciones:

- ▶ `[2, 3, 5]` ¿Cuál es el elemento con índice 1?
- ▶ `[2, 3.5, 'patito', 8]`.
- ▶ `[]` lista vacía.
- ▶ `c = [2, 3, 5]` define una lista de nombre `c`.
- ▶ `c[i]` accede o devuelve el elemento con índice `i` de la lista.
- ▶ `len(c)` devuelve la longitud de la lista.
- ▶ `c.append(x)` agrega el elemento `x` al final de la lista `c`.
- ▶ Y muchas operaciones más que iremos viendo.

# Variables, expresiones y asignaciones

Una *variable* es una dirección de memoria que almacena un valor.

# Variables, expresiones y asignaciones

Una *variable* es una dirección de memoria que almacena un valor.

Una *expresión* es una combinación de variables, valores y operadores.

# Variables, expresiones y asignaciones

Una *variable* es una dirección de memoria que almacena un valor.

Una *expresión* es una combinación de variables, valores y operadores.

Una *asignación* es una instrucción que guarda en una variable una expresión.

# Variables, expresiones y asignaciones

Una *variable* es una dirección de memoria que almacena un valor.

Una *expresión* es una combinación de variables, valores y operadores.

Una *asignación* es una instrucción que guarda en una variable una expresión.

►  $b = 3$  asigna a la variable  $b$  el valor 3.



# Variables, expresiones y asignaciones

Una *variable* es una dirección de memoria que almacena un valor.

Una *expresión* es una combinación de variables, valores y operadores.

Una *asignación* es una instrucción que guarda en una variable una expresión.

- ▶  $b = 3$  asigna a la variable  $b$  el valor 3.
- ▶  $1+1$  es una expresión que da como resultado 2.

# Variables, expresiones y asignaciones

Una *variable* es una dirección de memoria que almacena un valor.

Una *expresión* es una combinación de variables, valores y operadores.

Una *asignación* es una instrucción que guarda en una variable una expresión.

- ▶ `b = 3` asigna a la variable `b` el valor 3.
- ▶ `1+1` es una expresión que da como resultado 2.
- ▶ `long = len([1,3,'a'])` asigna a la variable `long` la longitud de la lista `[1,3,'a']`

Asignación: `variable = expresión`.

Anaconda, Spyder, Python, IDEs...

¿Y qué demonios son todas esas cosas...?

# Anaconda, Spyder, Python, IDEs...

¿Y qué demonios son todas esas cosas...?

- ▶ Respuesta corta:

# Anaconda, Spyder, Python, IDEs...

¿Y qué demonios son todas esas cosas...?

- ▶ Respuesta corta:

- ▶ Anaconda: nuclea un montón de paquetes y “librerías” (bibliotecas) para usar y no tener que andar reinventando la rueda cada vez.

# Anaconda, Spyder, Python, IDEs...

¿Y qué demonios son todas esas cosas...?

- ▶ Respuesta corta:
  - ▶ Anaconda: nuclea un montón de paquetes y “librerías” (bibliotecas) para usar y no tener que andar reinventando la rueda cada vez.
  - ▶ Spyder: es un entorno que facilita programar en Python. Es una IDE.

# Anaconda, Spyder, Python, IDEs...

¿Y qué demonios son todas esas cosas...?

► Respuesta corta:

- Anaconda: nuclea un montón de paquetes y “librerías” (bibliotecas) para usar y no tener que andar reinventando la rueda cada vez.
- Spyder: es un entorno que facilita programar en Python. Es una IDE.
- IDE = integrated development environment  $\equiv$  entorno de desarrollo integrado.

# Anaconda, Spyder, Python, IDEs...

¿Y qué demonios son todas esas cosas...?

- ▶ Respuesta corta:
  - ▶ Anaconda: nuclea un montón de paquetes y “librerías” (bibliotecas) para usar y no tener que andar reinventando la rueda cada vez.
  - ▶ Spyder: es un entorno que facilita programar en Python. Es una IDE.
  - ▶ IDE = integrated development environment  $\equiv$  entorno de desarrollo integrado.
- ▶ Respuesta larga: nada de esto es necesario ni importante para aprender a programar o para codear en Python. Pasa más por gustos personales.



# Anaconda, Spyder, Python, IDEs...

¿Y qué demonios son todas esas cosas...?

- ▶ Respuesta corta:
  - ▶ Anaconda: nuclea un montón de paquetes y “librerías” (bibliotecas) para usar y no tener que andar reinventando la rueda cada vez.
  - ▶ Spyder: es un entorno que facilita programar en Python. Es una IDE.
  - ▶ IDE = integrated development environment  $\equiv$  entorno de desarrollo integrado.
- ▶ Respuesta larga: nada de esto es necesario ni importante para aprender a programar o para codear en Python. Pasa más por gustos personales.
  - ▶ Por ejemplo, en vez del Anaconda, se podría bajar Python (<https://www.python.org/>) e ir instalando paquetes.

# Anaconda, Spyder, Python, IDEs...

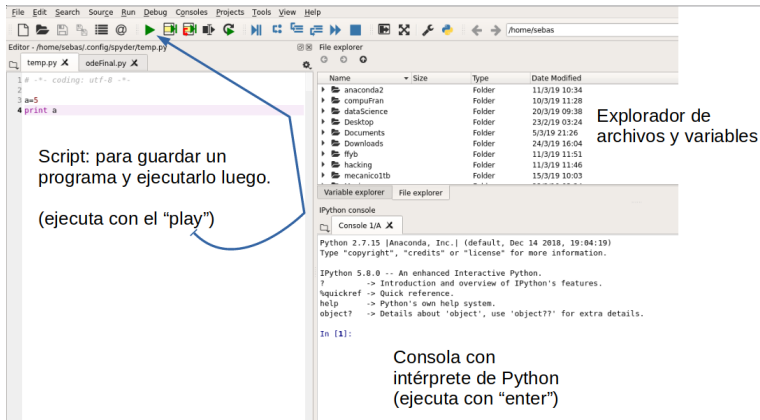
¿Y qué demonios son todas esas cosas...?

- ▶ Respuesta corta:
  - ▶ Anaconda: nuclea un montón de paquetes y “librerías” (bibliotecas) para usar y no tener que andar reinventando la rueda cada vez.
  - ▶ Spyder: es un entorno que facilita programar en Python. Es una IDE.
  - ▶ IDE = integrated development environment  $\equiv$  entorno de desarrollo integrado.
- ▶ Respuesta larga: nada de esto es necesario ni importante para aprender a programar o para codear en Python. Pasa más por gustos personales.
  - ▶ Por ejemplo, en vez del Anaconda, se podría bajar Python (<https://www.python.org/>) e ir instalando paquetes.
  - ▶ En vez de Spyder, para codear se puede usar simplemente un editor de textos, o cualquiera de las IDEs existentes (<https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>)

# Usando el Spyder como IDE para Python

Abriendo la IDE...

Anaconda-navigator → Spyder.



The screenshot displays the Spyder IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar contains icons for file operations and execution. The main editor window shows a Python script with the following code:

```
1 # -*- coding: utf-8 -*-
2
3 a=5
4 print a
```

A blue arrow points from the text "Script: para guardar un programa y ejecutarlo luego. (ejecuta con el 'play')\" to the 'Run' button in the toolbar. The File explorer on the right shows a list of files and folders, including anaconda2, compuFran, dataScience, Desktop, Documents, Downloads, flyb, hacking, and mecanico1tb. The IPython console at the bottom shows the output of the script:

```
Python 2.7.15 [Anaconda, Inc.] (default, Dec 14 2018, 19:04:19)
Type "copyright", "credits" or "license" for more information.

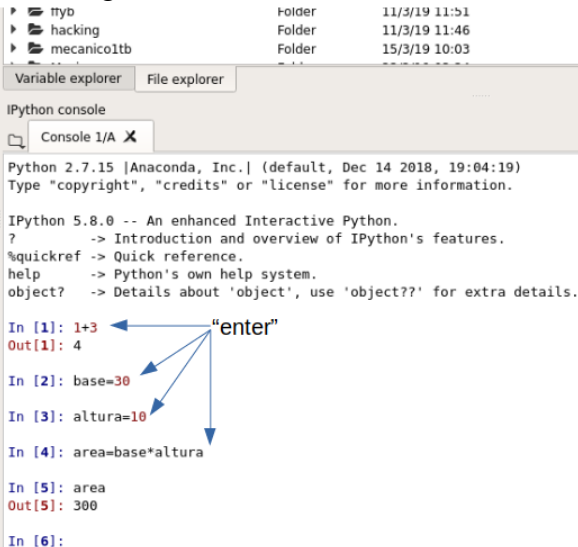
IPython 5.8.0 -- An enhanced Interactive Python.
?      -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help    -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

In [1]:
```

The text "Explorador de archivos y variables" is positioned next to the File explorer, and "Consola con intérprete de Python (ejecuta con 'enter')\" is positioned next to the IPython console.

# Usando la consola intérprete de Python (desde el Spyder)

Probar lo siguiente:



```
Python 2.7.15 [Anaconda, Inc.] (default, Dec 14 2018, 19:04:19)
Type "copyright", "credits" or "license" for more information.

IPython 5.8.0 -- An enhanced Interactive Python.
?                -> Introduction and overview of IPython's features.
%quickref        -> Quick reference.
help             -> Python's own help system.
object?         -> Details about 'object', use 'object??' for extra details.

In [1]: 1+3
Out[1]: 4

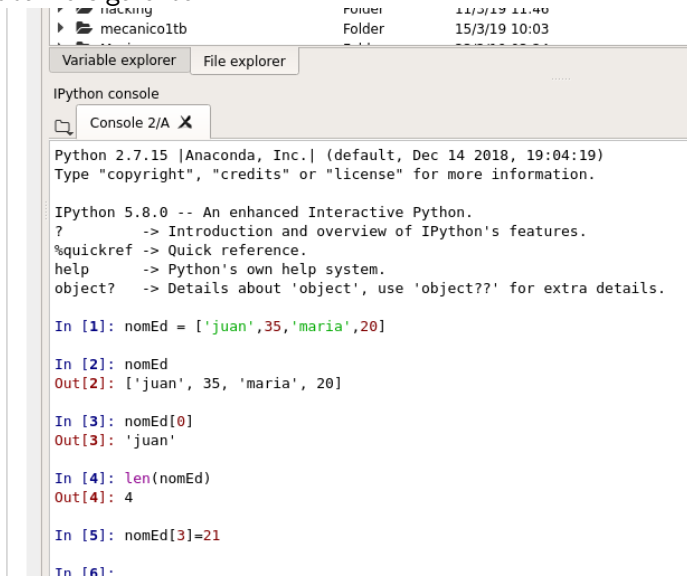
In [2]: base=30
In [3]: altura=10
In [4]: area=base*altura

In [5]: area
Out[5]: 300

In [6]:
```

# Usando la consola intérprete de Python (desde el Spyder)

Probar lo siguiente:



The screenshot shows the Spyder Python IDE interface. At the top, there's a file explorer showing a folder named 'mecanicoltb'. Below it, there's a 'Variable explorer' and a 'File explorer' tab. The main area is the 'IPython console', which displays the following text:

```
Python 2.7.15 |Anaconda, Inc.| (default, Dec 14 2018, 19:04:19)
Type "copyright", "credits" or "license" for more information.

IPython 5.8.0 -- An enhanced Interactive Python.
?                -> Introduction and overview of IPython's features.
%quickref        -> Quick reference.
help             -> Python's own help system.
object?         -> Details about 'object', use 'object??' for extra details.

In [1]: nomEd = ['juan',35,'maria',20]

In [2]: nomEd
Out[2]: ['juan', 35, 'maria', 20]

In [3]: nomEd[0]
Out[3]: 'juan'

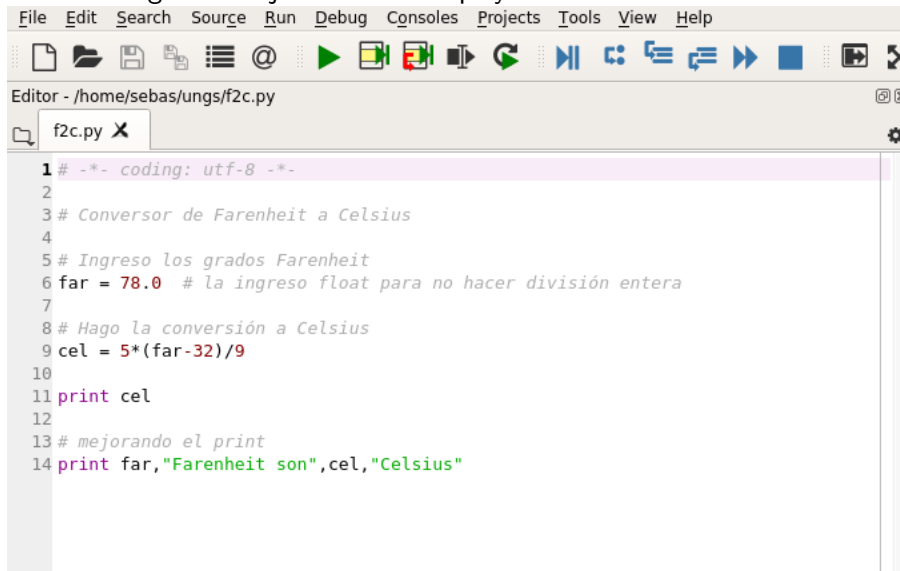
In [4]: len(nomEd)
Out[4]: 4

In [5]: nomEd[3]=21

In [6]:
```

# Usando Python desde un script

Probar lo siguiente. Ejecutar con el “play”.



The screenshot shows an IDE window titled "Editor - /home/sebas/ungs/f2c.py". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar contains icons for file operations and code execution. The code editor displays a Python script with line numbers 1 through 14. The script defines a variable 'far' with the value 78.0 and calculates 'cel' using the formula 5\*(far-32)/9. It then prints the result. The script is encoded in utf-8.

```
1 # -*- coding: utf-8 -*-  
2  
3 # Conversor de Farenheit a Celsius  
4  
5 # Ingreso los grados Farenheit  
6 far = 78.0 # la ingreso float para no hacer división entera  
7  
8 # Hago la conversión a Celsius  
9 cel = 5*(far-32)/9  
10  
11 print cel  
12  
13 # mejorando el print  
14 print far, "Farenheit son", cel, "Celsius"
```

# ¡Manos a la obra!

## Ejercicios:

1. Escribir un script que convierta un valor en millas a kilómetros. Guardar ese script en un archivo de nombre `m2k.py`
2. Modificar `m2k.py` para redondear el resultado a dos decimales después de la coma. Usar la función `round` (ejemplo: `round(XX,d)` redondea el número `XX` a `d` decimales después de la coma). Guardar el script como `m2k_v2.py`
3. Modificar `m2k_v2.py` para que convierta una lista de 3 valores en millas, a otra lista con sus 3 valores correspondientes en kilómetros. Guardar el script como `m2k_v3.py`