

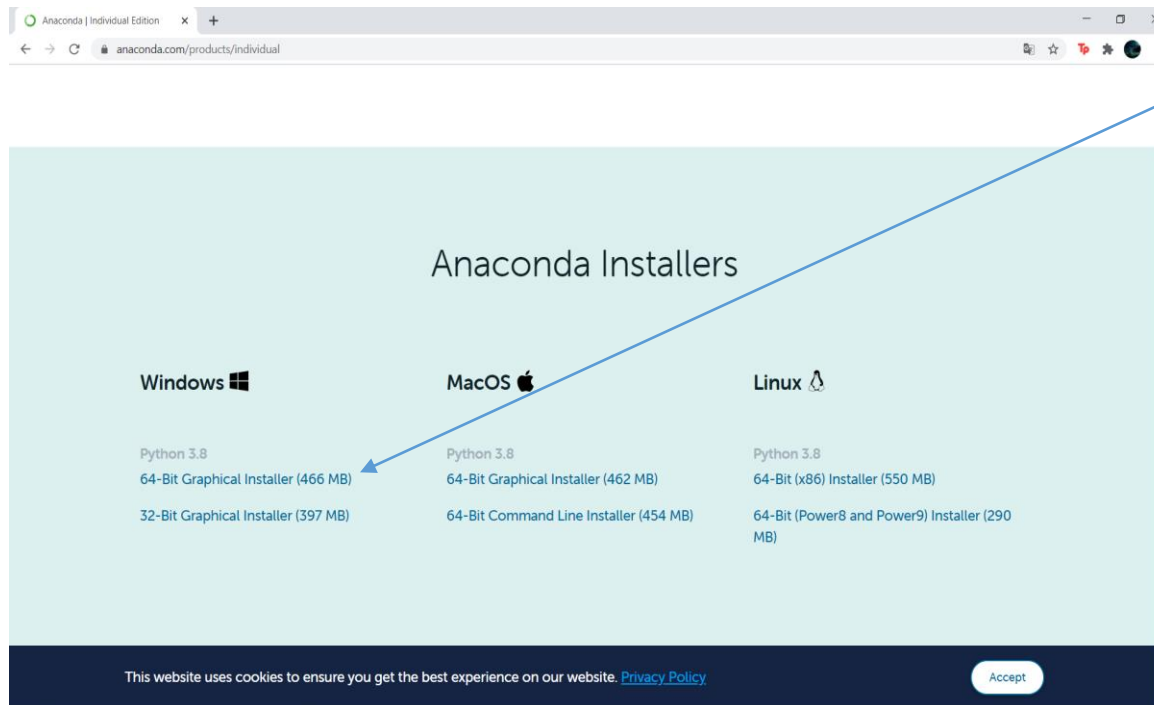
Introducción a la programación con Python

Alexis Gómez Chimeno

2020

Instalación de Python

Vamos a usar la distribución llamada Anaconda que contiene Python, un amplio juego de herramientas y la mayoría de los paquetes software más comunes.

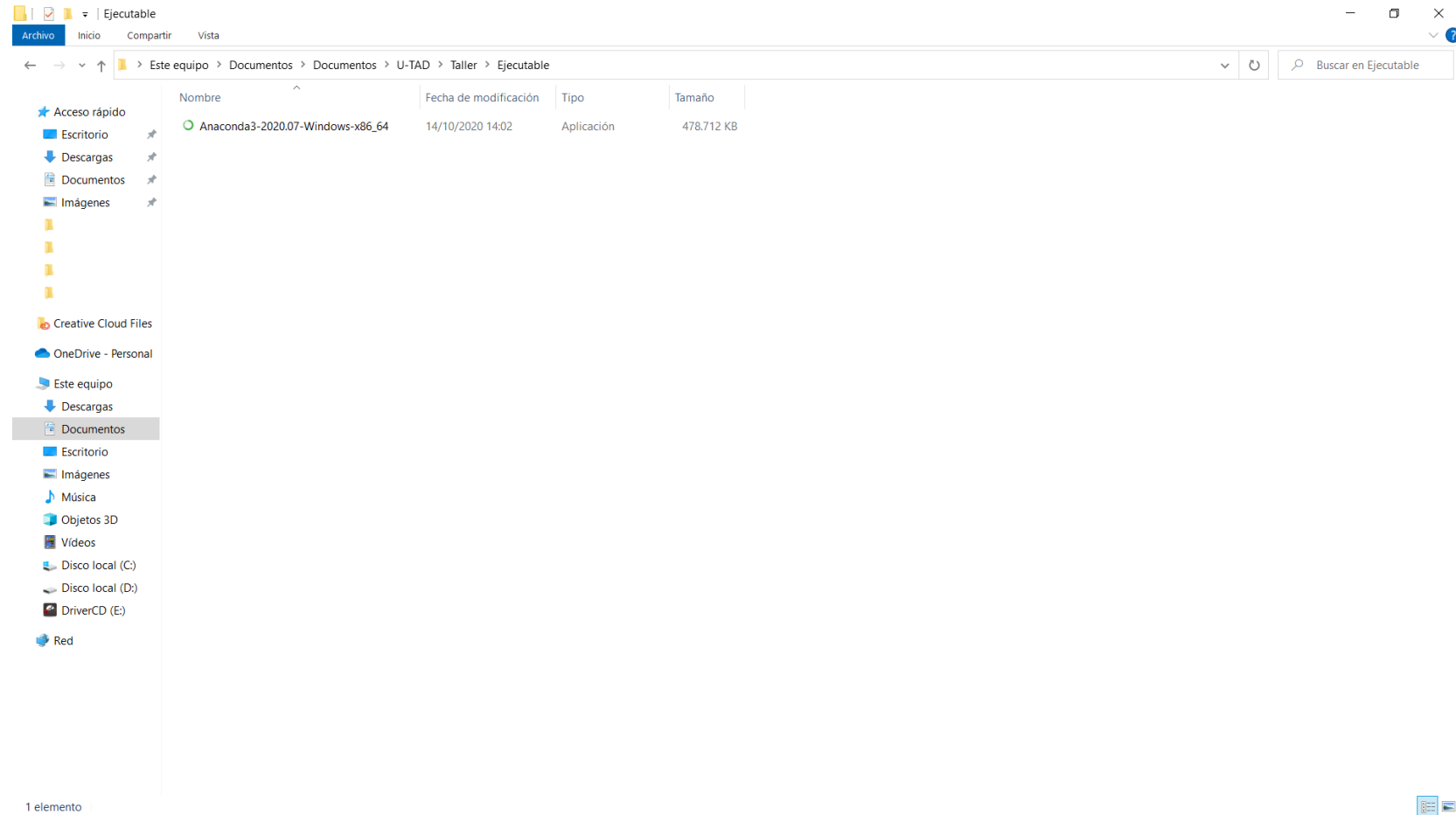


Ve a

<https://www.anaconda.com/download/> y

haz click sobre el botón Download

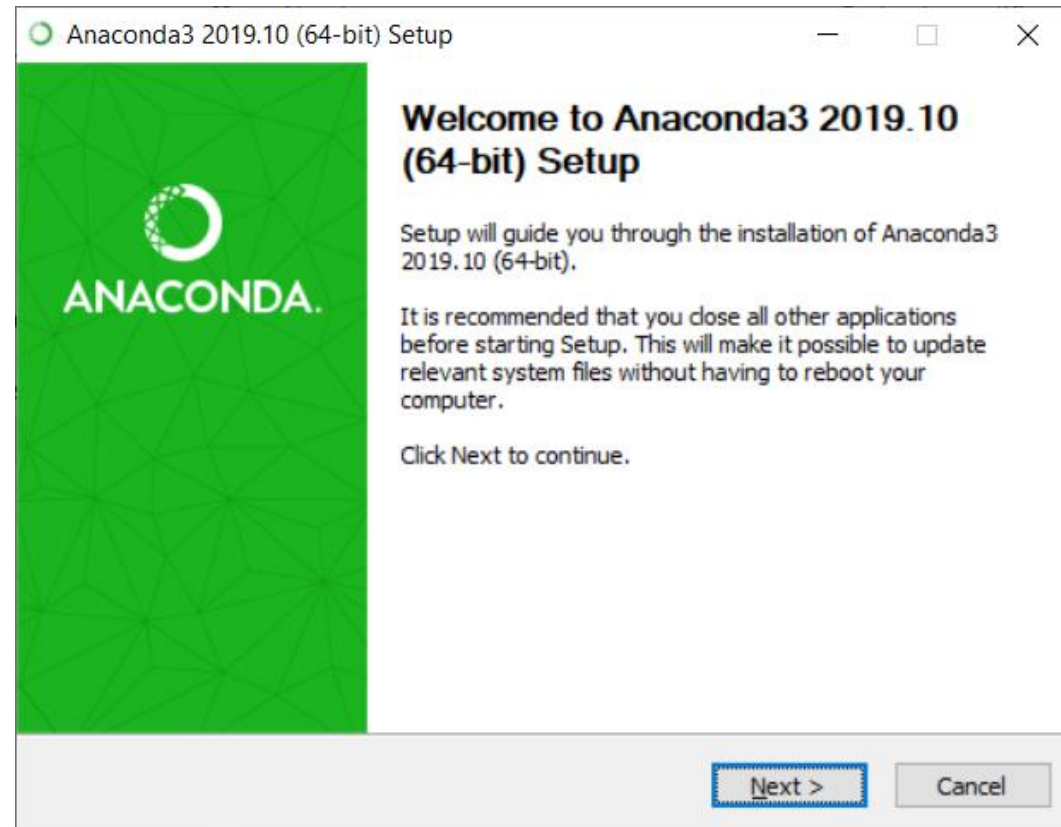
Instalación de Python



Ve a la carpeta en la que se ha descargado el fichero, haz click con el botón derecho del ratón y selecciona “Ejecutar como administrador”

Instalación de Python

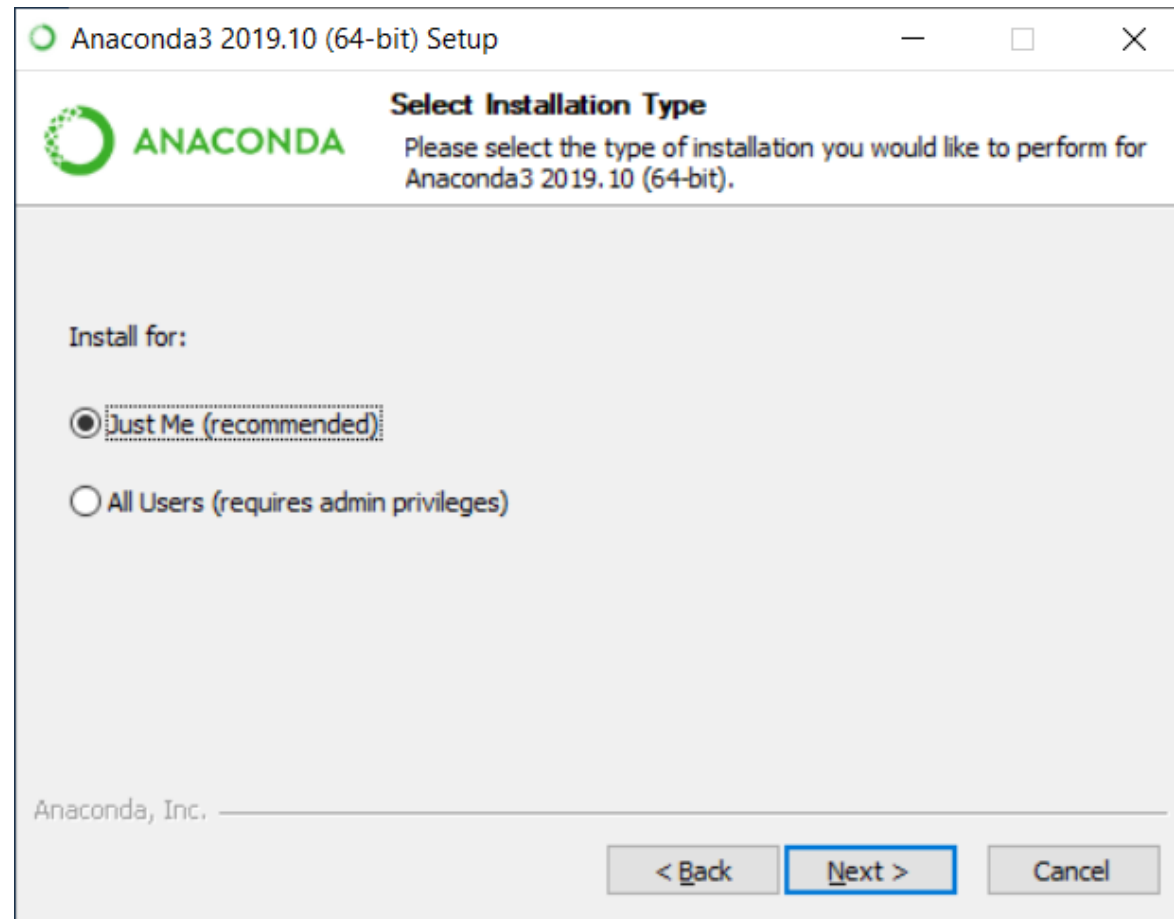
Es posible que aparezca una advertencia de seguridad de Windows indicando que el programa quiere realizar cambios en tu máquina. Selecciona SI



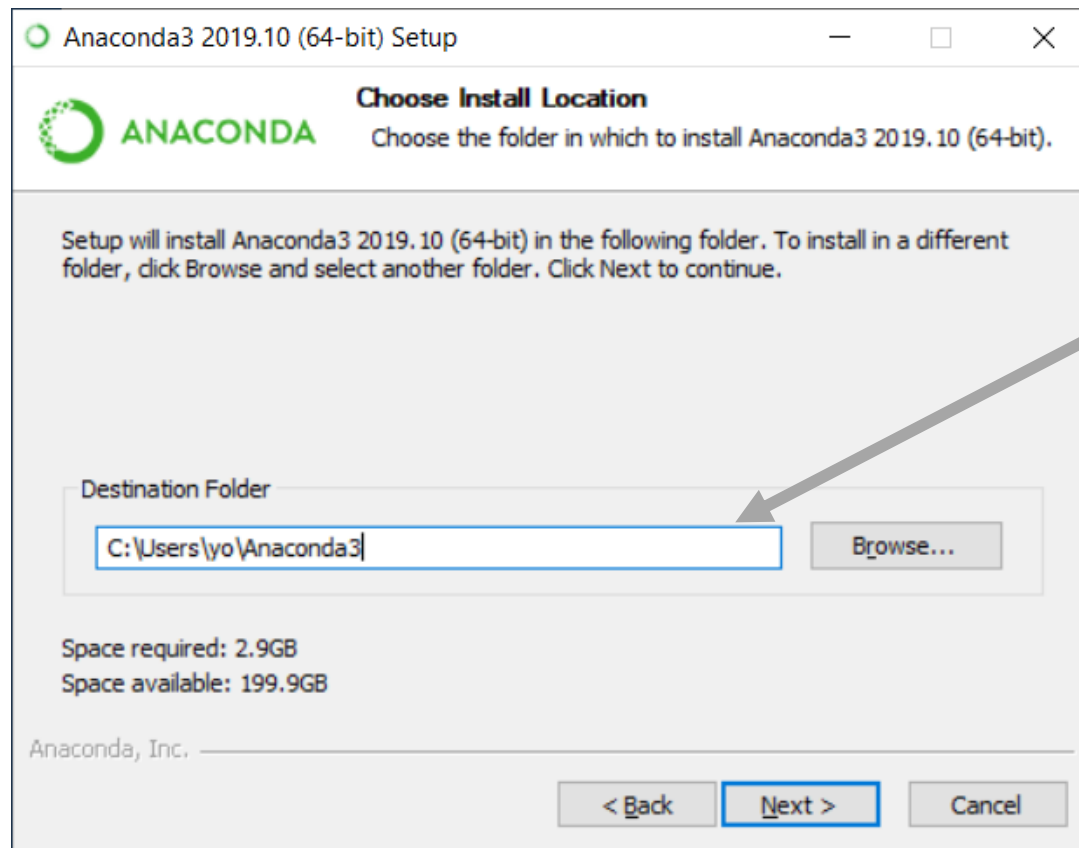
Lo siguiente que verás es esta pantalla. Haz click en el botón Next

Instalación de Python

En este otro cuadro selecciona “Just Me”

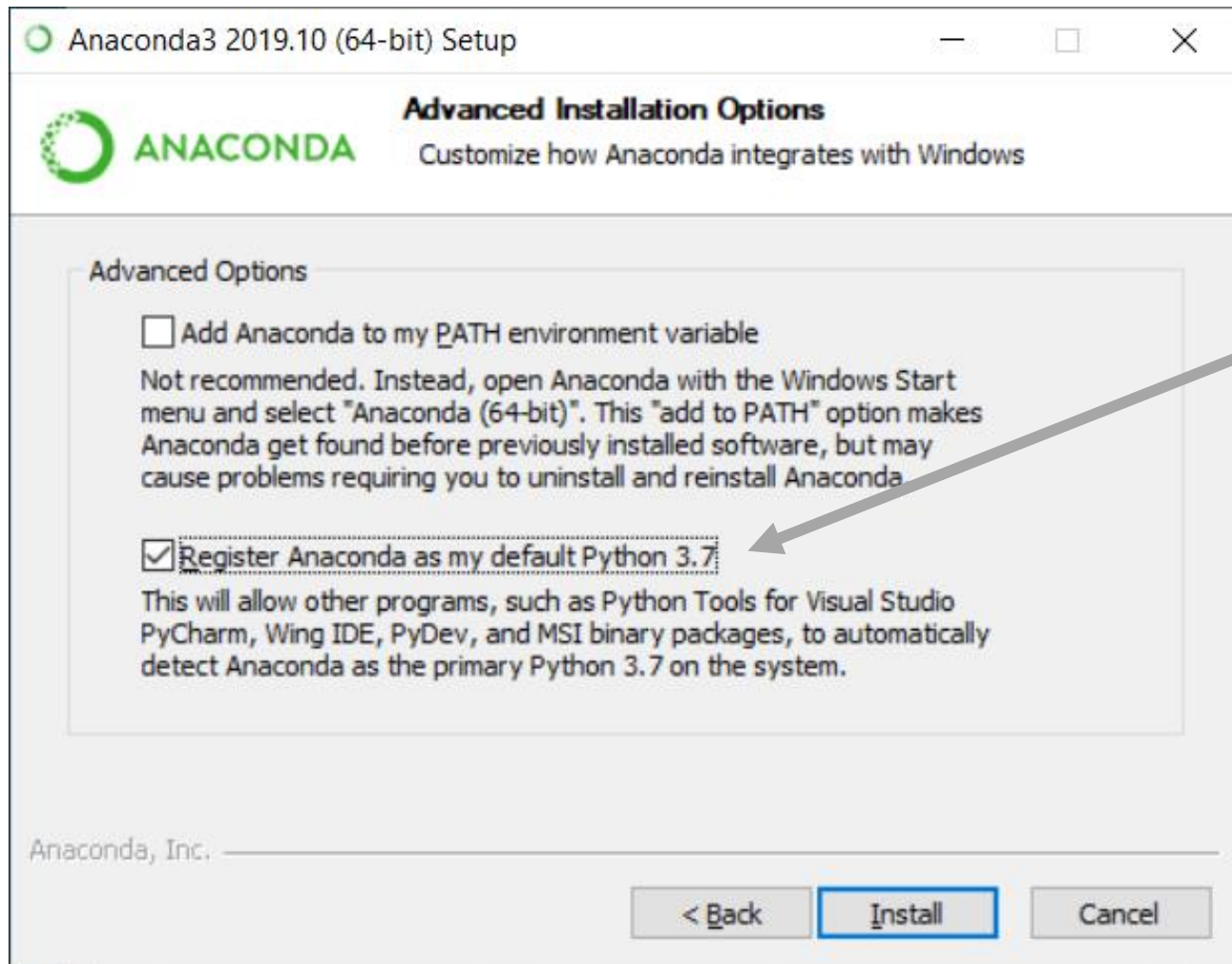


Instalación de Python



Aquí indica la carpeta donde quieres que se instale. El programa te dirá una por defecto, puedes dejar esa misma. Es importante que anotes cual es la carpeta de instalación porque luego te va a hacer falta saberlo

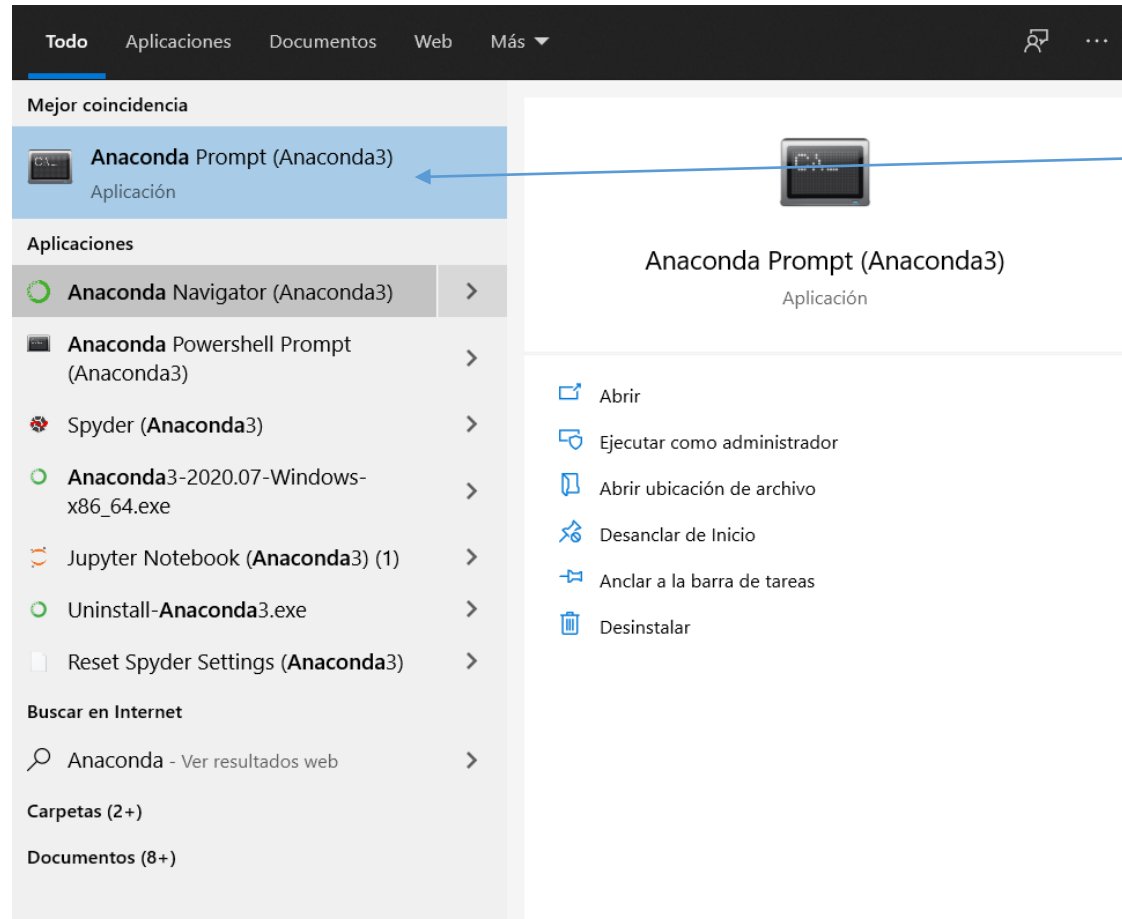
Instalación de Python



Marca esta casilla si no lo está y después haz click en *Install*.

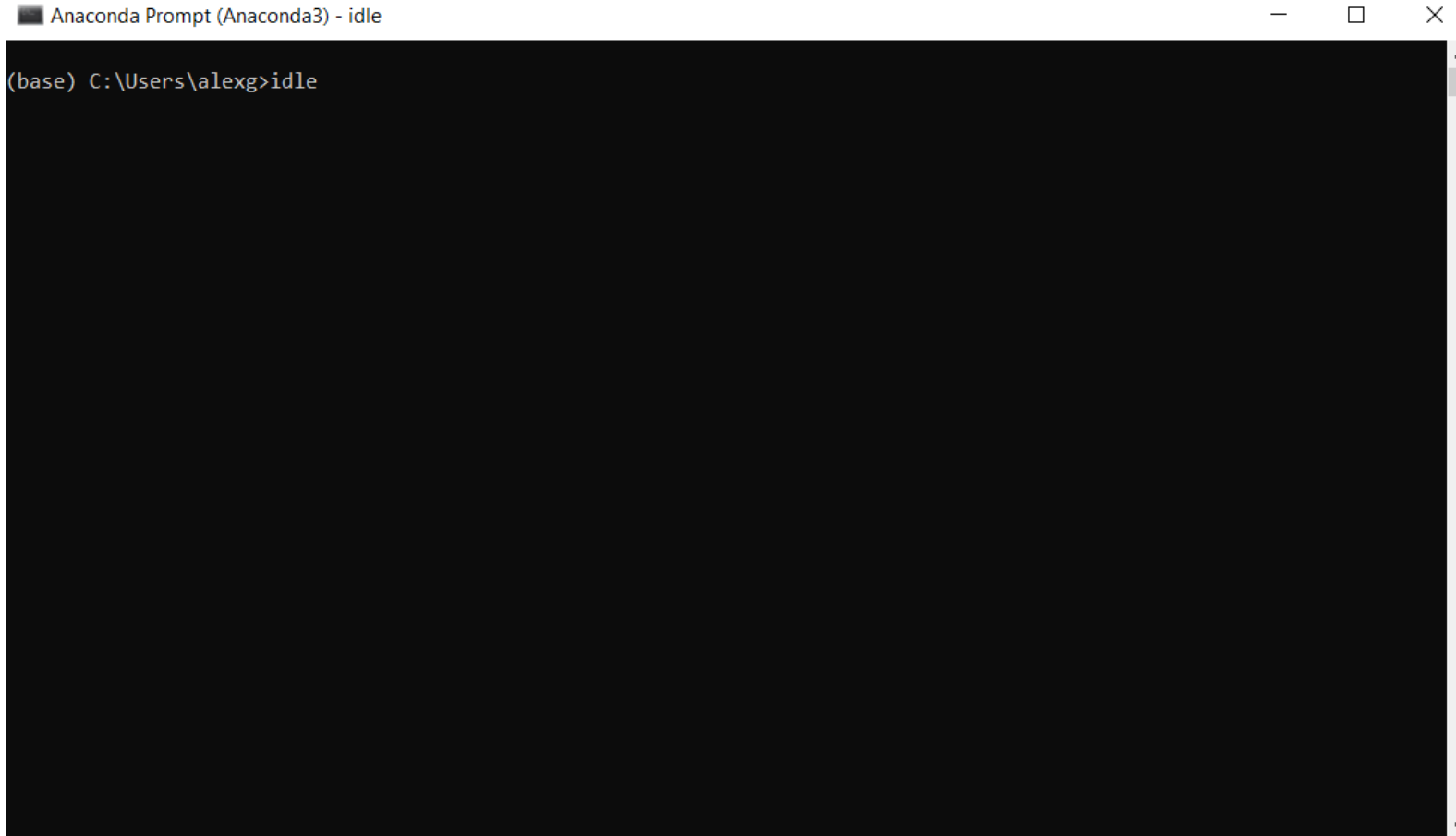
El proceso llevará unos minutos. Al finalizar ya tenemos Python en nuestra máquina

Instalación de Python



Pulsad aquí
para ejecutar
la consola de
comandos

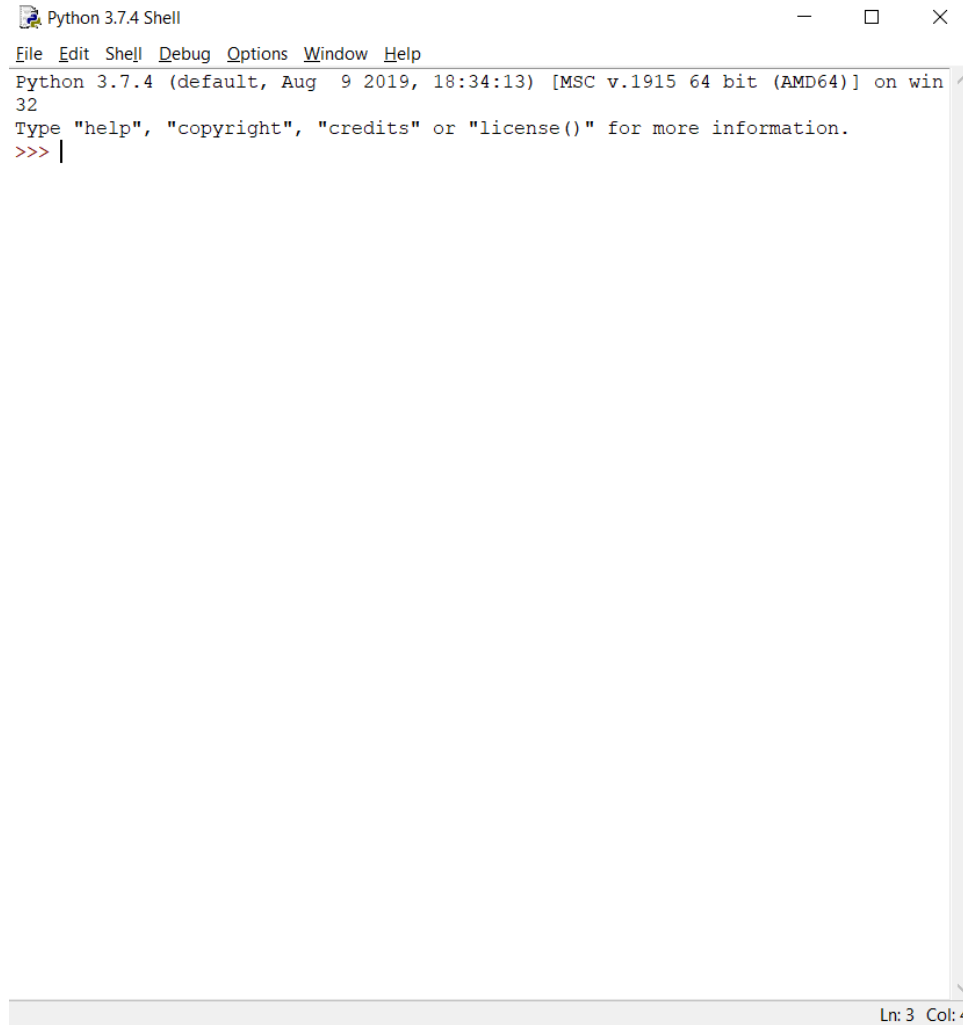
Instalación de Python

A screenshot of the Anaconda Prompt (Anaconda3) - idle window. The window has a title bar with standard Windows window controls (minimize, maximize, close). The terminal area is black with white text. The prompt shows the current directory as C:\Users\alexg and the command 'idle' has been entered.

```
Anaconda Prompt (Anaconda3) - idle
(base) C:\Users\alexg>idle
```

Escribir el comando “idle” para abrir el IDE.

Instalación de Python



A screenshot of a Windows application window titled "Python 3.7.4 Shell". The window has a standard menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following information: "Python 3.7.4 (default, Aug 9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] on win32", followed by "Type 'help', 'copyright', 'credits' or 'license()' for more information.", and the interactive prompt ">>> |". The status bar at the bottom right indicates "Ln: 3 Col: 4".

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (default, Aug 9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] on win
32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Ln: 3 Col: 4

Se debe haber abierto
la siguiente ventana

HELLO WORLD

Comandos que se deben utilizar: `print()`

Explicación: la función `print()` muestra por pantalla tanto las variables como el texto que queráis. Para escribir cualquier texto pondremos: `print("Este es mi primer texto")`.

Variables

Las variables son una serie de contenedores que permiten guardar valores que especifiquemos. Es decir, cada variable es una caja en la que solo se puede guardar una cosa. No obstante, si podemos trabajar con lo que guardamos en esas cajas y modificar lo que contienen.

Python utiliza el tipado dinámico. Es decir, no hace falta especificar de qué tipo es nuestra variable pues lo reconoce por sí solo.

Tipos de variables:

- int -> entero -> 3
- char -> carácter -> 'a'
- float -> número real (en coma flotante) -> 3.142512
- string -> cadena de caracteres -> "palabra"
- boolean -> true or false (verdadero o falso)

Operadores

Los operadores son símbolos reservados por el lenguaje de programación que especifican operaciones matemáticas.

Operadores básicos:

- '+' -> operador de suma
- '-' -> operador de resta
- '*' -> operador de multiplicación
- '/' -> operador de división
- '**' -> operador de exponenciación

Operadores de comparación:

- '<' -> menor que
- '>' -> mayor que
- '>=' -> menor o igual que
- '<=' -> mayor o igual que
- '==' -> igual que
- '!=' -> distinto que

Operadores lógicos:

- '&&' o 'and' -> equivale a la conjunción 'y'
- '||' o 'or' -> equivale a la conjunción 'o'

En Python se pueden hacer cosas muy interesantes como sumar dos strings o multiplicar una string por un valor entero. ¡Pruébalo!

Input

La función input permite al usuario que ejecute el programa introducir valores para que estos se utilicen en el mismo.

Esta función admite pasarle una cadena de texto para que se la muestre al usuario antes de que él/ella introduzcan el valor pedido. En cualquier caso, esta función devolverá el valor introducido

Función: input()

Sintaxis: a= input("Quiero un número entero: ")

Ejemplo de funcionamiento: Quiero un número entero: 3.
A partir de ahora nuestra variable a será igual a 3.

Listas

Las listas son unos contenedores en las que puedes guardar varias cosas. A diferencia de las variables, las listas se utilizan para guardar 1 o más cosas y cada cosa que guardes en la lista tiene un índice que especifica su posición.

En los lenguajes de programación la posición inicial donde se encuentra el primer objeto es el 0.

De esta manera los índices irán desde el 0 hasta el número máximo de objetos de la lista menos 1.

Tamaño de la lista = n

n es cualquier número entero

Índices = $0, 1, 2, 3, \dots, n-1$

Sintaxis: lista = $[0, 1, 2, \text{"hola"}, \text{true}, \text{'a'}]$

Condicionales

Los condicionales son una serie de sentencias que se ejecutan en función de una condición.

Por ejemplo: si mi hermana tiene los ojos azules, yo tengo los ojos marrones; pero si mi hermana los tiene verdes, yo los tengo azules; si no los tiene ni azules ni verdes, entonces los tengo verdes.

Sintaxis: if(condición), elif(segunda condición), else(en caso contrario)

Demostración de uso:

```
a = 10
b = 20
if(a<b):
    print(b)
elif(a==b):
    print(a+b)
else:
    print(a)
```


Bucles

Un bucle es una secuencia que ejecuta repetidas veces las sentencias que le especifiquemos.

En Python existen dos maneras de hacer bucles:

1. El bucle for
2. El bucle while(condicion)

Los bucles **FOR** hacen uso de listas para ejecutarse un determinado número de veces, cuando queremos crear una lista ordenada de números utilizando la función 'range(numero)'. También admite la siguiente sintaxis: range(inicio intervalo, fin intervalo) y range(inicio intervalo, fin intervalo, incremento). Se ha de destacar que el final del intervalo es abierto y no cerrado. Es decir, recorre hasta el anterior numero especificado.

Sintaxis:

```
for i in range(6):  
    print(i)
```

Los bucles **WHILE** hacen uso de los condicionales para repetir su ejecución.

Sintaxis:

```
a=10  
while(a<20):  
    print(a)  
    a=a+1
```

Función factorial

Para explicar las funciones, vamos a crear nuestra propia función para calcular números factoriales.

La definición más simple de un número factorial es:

$$n! = n * (n - 1)!$$

$$\forall n \in N \text{ tal que } 0! = 1! = 1$$

¿Qué significa esto?

Simplemente, un número factorial es aquel que equivale a la multiplicación de todos sus números anteriores hasta llegar al 1.

Es decir:

$$7! = 7 * 6 * 5 * 4 * 3 * 2 * 1$$

Teniendo esto en cuenta y utilizando los condicionales y los bucles. Es el momento de crear nuestra función factorial.

Sintaxis:

```
def nombreFuncion(argumento):  
    bucles/condicionales
```