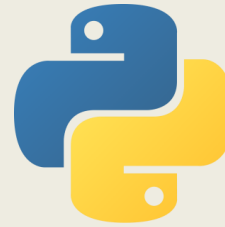
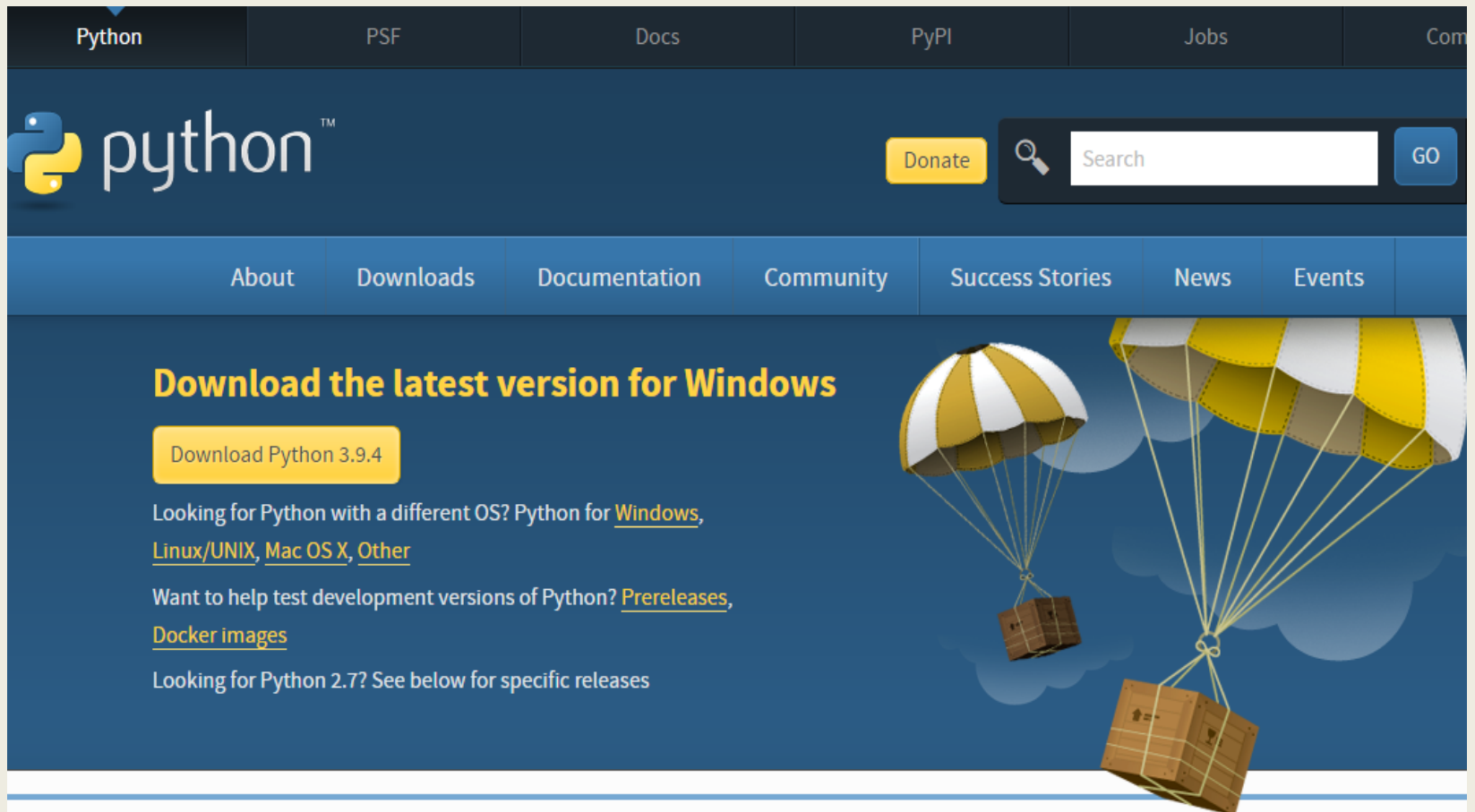


Python



Para descargar Python vamos a la siguiente página <https://www.python.org/downloads/>

A screenshot of the Python.org website. The top navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Com. Below this is a dark blue header with the Python logo, a 'Donate' button, and a search bar with a 'GO' button. A secondary navigation bar contains links for About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a large yellow button labeled 'Download Python 3.9.4' and text links for other operating systems, prereleases, Docker images, and Python 2.7. An illustration of two parachutes carrying boxes is on the right.

Python

PSF

Docs

PyPI

Jobs

Com

python™

Donate

Search

GO

About

Downloads

Documentation

Community

Success Stories

News

Events

Download the latest version for Windows

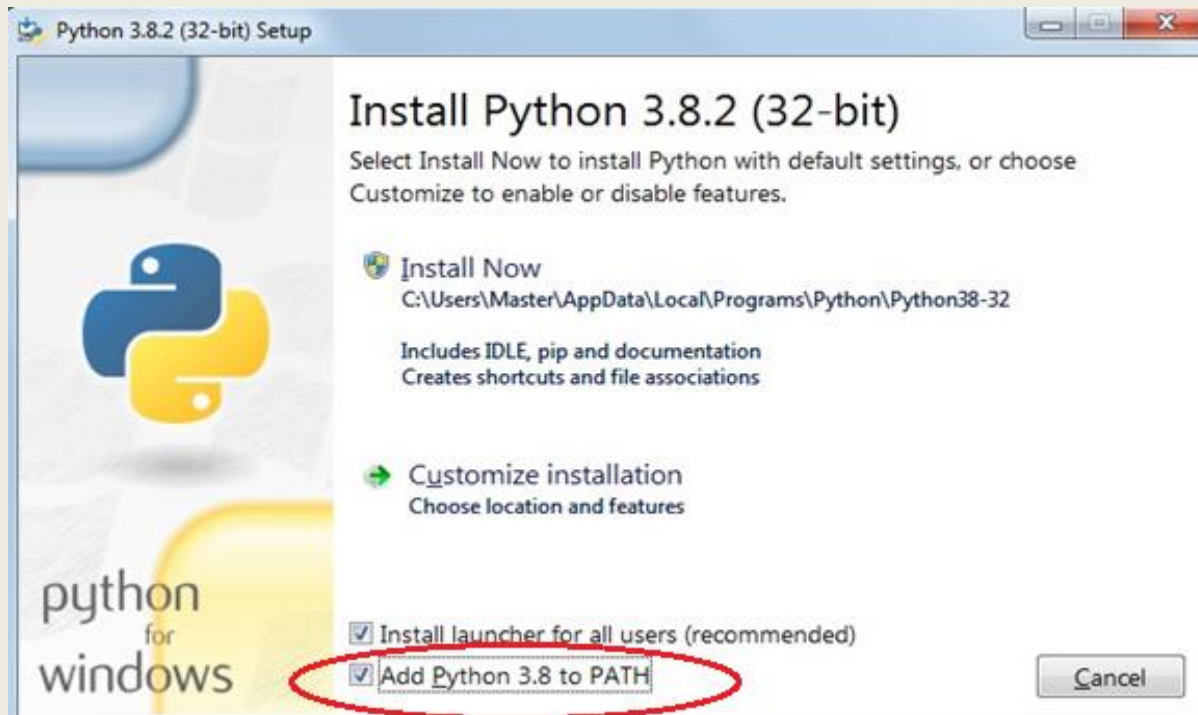
Download Python 3.9.4

Looking for Python with a different OS? Python for [Windows](#),
[Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#),
[Docker images](#)

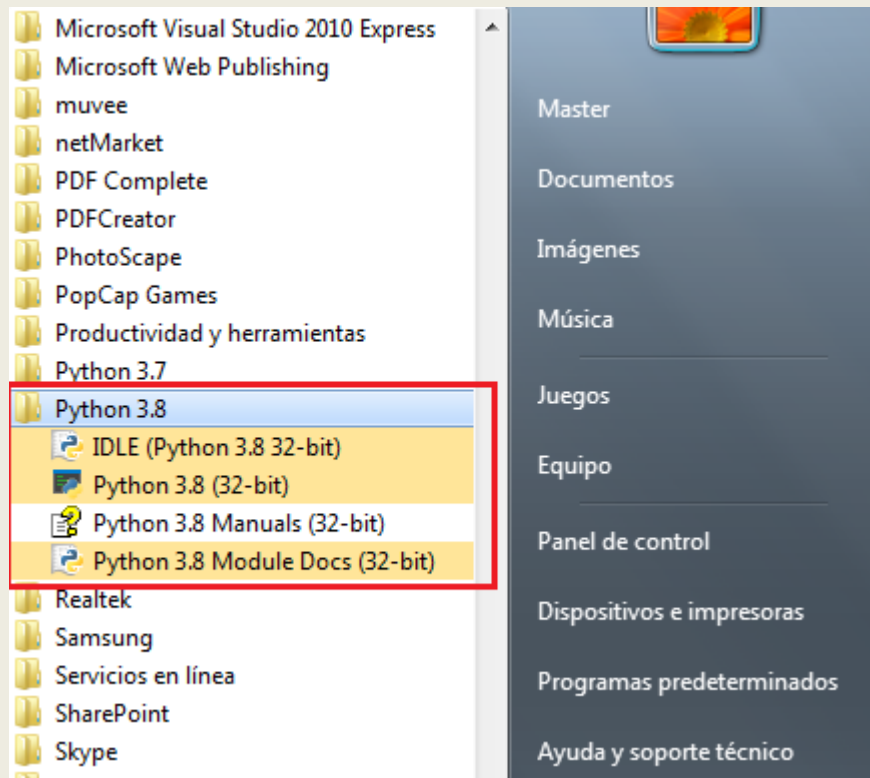
Looking for Python 2.7? See below for specific releases

Una vez completa la descarga lo instalamos teniendo en cuenta lo siguiente:



Es muy importante que al instalar hagamos clic en Add Python x.x to PATH para que luego podamos instalar componentes de Python.

Luego de la instalación nos va a aparecer la carpeta de Python y estos ítems



IDLE Integrated Development and Learning Environment. Entorno de desarrollo Editor y Shell (consola)

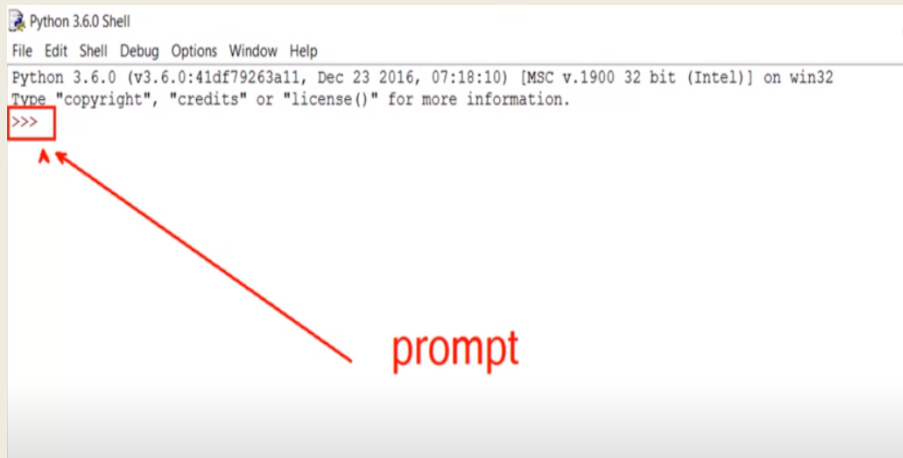
Tiene dos tipos de pantallas la de Shell y la de Editor

El ítem que está abajo en el menú es una especie de Consola.

Para ver explicación sobre IDLE <https://docs.python.org/3/library/idle.html>

Entornos de Desarrollo para Python

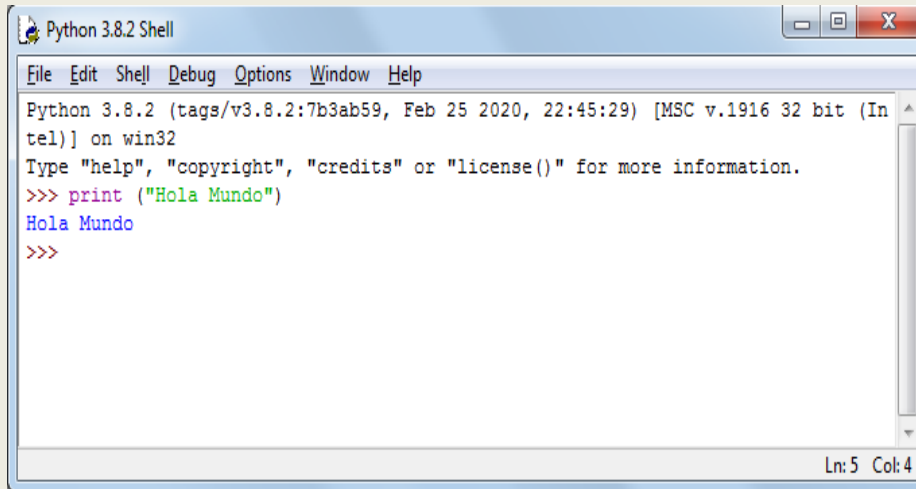
Sublime Text 3 Visual Estudio Code Eclipse Notepad++ o el entorno de desarrollo que quieras



Un **editor** que te ayudará a escribir el código

Una **consola** en la que puedes iniciar tu código recién escrito y detenerlo por la fuerza cuando se sale de control.

Una herramienta llamada **depurador**, capaz de ejecutar tu código paso a paso y te permite inspeccionarlo en cada momento de su ejecución.



Instrucción línea de código

Los finales de instrucción en Python **no llevan ;** sólo pulsar enter y se ejecuta.

Se puede poner más de una instrucción en una línea, pero no es aconsejable. Para hacerlo se debe poner ;

Comentarios:

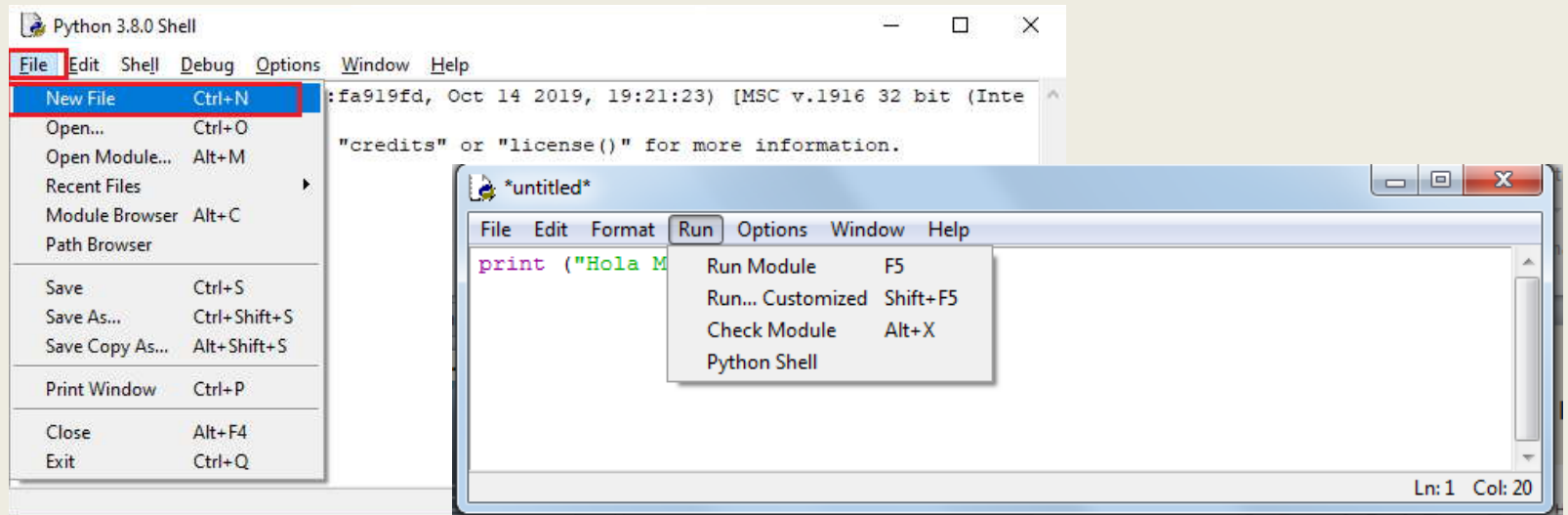
Almohadilla #

barra invertida \ un salto de línea en la escritura pero no en el código. Se puede seguir escribiendo.

Indentación la consola la hace automáticamente. Es muy importante para marcar los bucles y condiciones.

Si van a usar un editor por ejemplo Sublime Text o Visual Estudio Code tienen que instalar las herramientas de Python para que haga las indentaciones

Si quiero guardar el código en lugar de ejecutar directamente



y después ejecuto con la opción Run o F5

También se pueden ejecutar desde una ventana de comandos /modo interactivo/consola

Diferencia entre .py y .pyw

Los programas Python pueden tener dos extensiones: .py y .pyw. La más utilizada es la primera, .py.

Si se ejecutan desde IDLE, no hay diferencia entre ambas extensiones.

Pero si se ejecutan desde un terminal o haciendo doble clic sobre los ficheros, entonces sí que hay diferencias. Los archivos .py son ejecutados por *python.exe*, mientras que los archivos .pyw son ejecutados por *pythonw.exe*.

La *principal diferencia* es que python.exe crea una ventana de terminal (o aprovecha la ventana de terminal desde la que se ejecuta el programa), ventana que permite pedir valores al usuario o imprimir mensajes, mientras que pythonw.exe no crea ninguna ventana de terminal.

Solamente se debe utilizar la extensión .pyw si el programa crea y gestiona su propia ventana de interfaz de usuario o si no queremos ni pedir datos al usuario ni mostrarle ninguna salida del programa. En caso contrario, es mejor utilizar la extensión .py.

Otra diferencia es que python.exe ejecuta los programas de forma síncrona, es decir, que en un terminal no se puede ejecutar un nuevo programa .py hasta que ha terminado el programa anterior, mientras que pythonw.exe ejecuta los programas de forma asíncrona, es decir, que se pueden ir ejecutando nuevos programas aunque los anteriores no hayan terminado de ejecutarse.

PRINT()

Mostrar: textos y variables

`print("¡Hola, Mundo!", nota)` **cadena está delimitada por comillas**

`(\n)` indicar nueva línea `end="\n".`

`end=`

`sep=`

```
print("Mi nombre es", "Python.",  
      end=" ")  
print("Monty Python.")
```

Si escribimos el siguiente código

```
print("Fundamentos", "Programación", "en", sep="***", end="...")  
print("Python")
```

Vamos a obtener

```
Fundamentos***Programación***en...Python
```

Supongamos que se desea mostrar un muy sencillo mensaje:

Me gusta "Monty Python"

¿Cómo se puede hacer esto sin generar un error? Existen dos posibles soluciones.

La primera se basa en el concepto ya conocido del **carácter de escape**, el cual recordarás se utiliza empleando la **diagonal invertida**. La diagonal invertida puede también escapar de la comilla. Una comilla precedida por una diagonal invertida cambia su significado, no es un limitador, simplemente es una comilla. Lo siguiente funcionará como se desea:

```
print("Me gusta \"Monty Python\"")
```

La segunda solución: Python puede utilizar **una apóstrofe en lugar de una comilla**.

Cualquiera de estos dos caracteres puede delimitar una cadena, pero para ello se debe ser **consistente**.

Si se delimita una cadena con una comilla, se debe cerrar con una comilla.

Si se inicia una cadena con un apóstrofe, se debe terminar con un apóstrofe.

Este ejemplo funcionará también:

```
print('Me gusta "Monty Python"')
```


- Hallar la superficie de un cuadrado conociendo el valor de un lado

INPUT()

Ingresar: datos

```
lado=input("Ingrese la medida del lado del cuadrado:")
lado=int(lado)
superficie=lado*lado
print("La superficie del cuadrado es")
print(superficie)
lado=input("Ingrese la medida del lado del cuadrado:")
```

Para el ingreso de un dato por teclado y mostrar un mensaje se utiliza la función **input**, esta función retorna todos los caracteres escritos por el operador del programa:

```
lado=input("Ingrese la medida del lado del cuadrado:")
```

La variable lado guarda todos los caracteres ingresados pero no en formato numérico, para esto debemos llamar a la función int:

```
lado=int(lado)
```

Podemos unir las instrucciones y escribir directamente:

```
lado=int(input("Ingrese la medida del lado del cuadrado:"))
```

Algunas consideraciones

Python es sensible a mayúsculas y minúsculas, no es lo mismo llamar a la función input con la sintaxis: **Input**.

Los nombres de variables también son sensibles a mayúsculas y minúsculas. Son dos variables distintas si en un lugar iniciamos a la variable "superficie" y luego hacemos referencia a "Superficie"

Los nombres de variable no pueden tener espacios en blanco, caracteres especiales y empezar con un número.

Operadores Matemáticos

+ Suma

- Resta

* Multiplicación

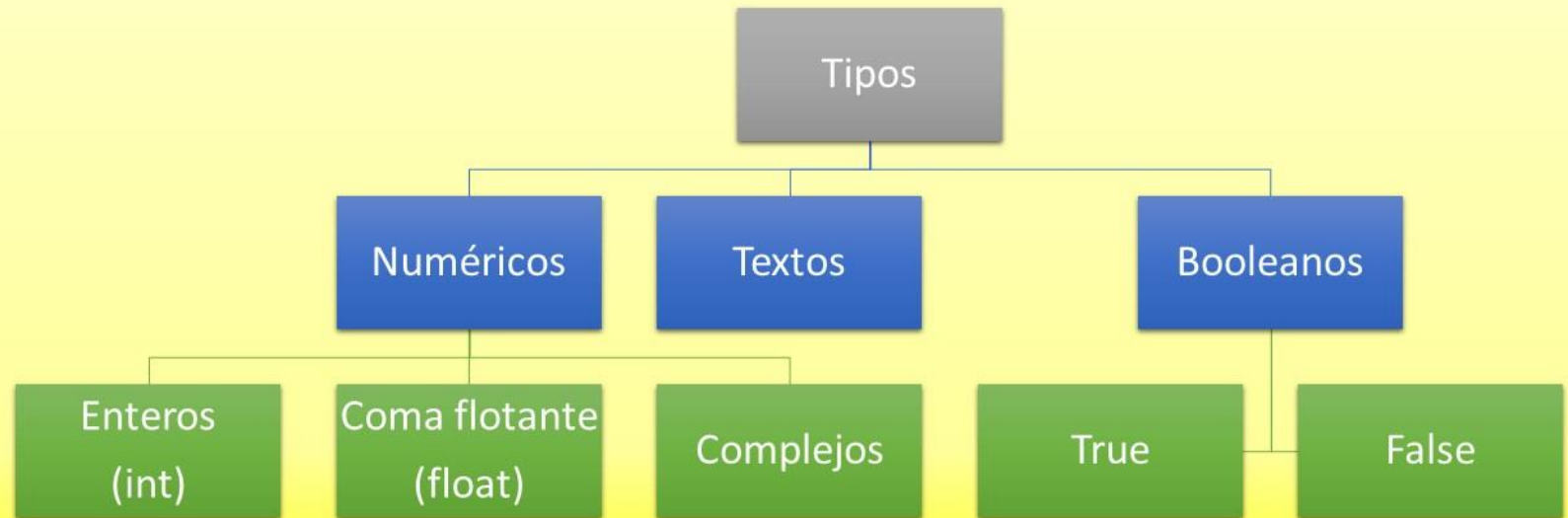
/ División de flotantes

// División de enteros

% Resto de una división

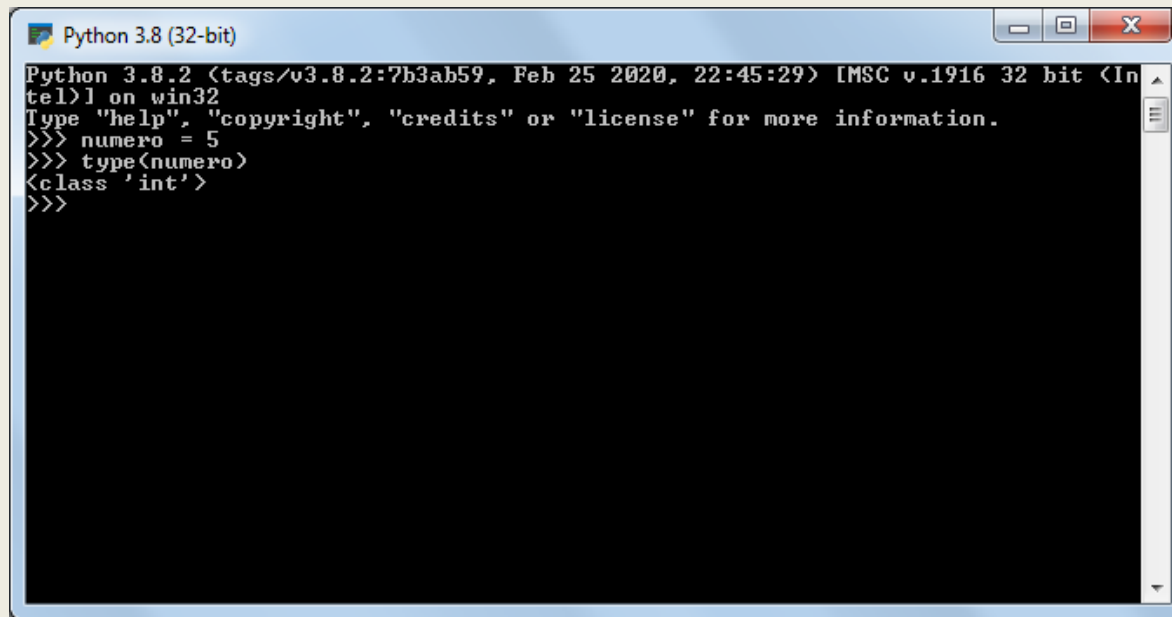
** Exponenciación

Tipos de Variables



Importante el tipo de la variable lo establece el contenido no por el contenedor
Es decir si digo que número = 5 está variable va a ser de tipo numérica entera
También tener en cuenta que en Python todo es un objeto, es un lenguaje 100% objeto
Por ejemplo cuando creamos una variable número = 5 para Python esta variable número es un objeto.

Lo vamos a ver usando una función predefinida de Python que es la **función type()** que nos devuelve el tipo de variable . Y vamos a ver que lo va a hacer con una nomenclatura que usa la palabra class obtenemos los siguiente:



```
Python 3.8 (32-bit)
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit <In
tel>] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> numero = 5
>>> type(numero)
<class 'int'>
>>>
```

Igual para float y para str

str(número)

Concatenación

El signo de + (más), al ser aplicado a dos cadenas, se convierte en **un operador de concatenación**:

Este es un programa sencillo que muestra como funciona el signo + como concatenador:

```
nom = input("¿Me puedes dar tu nombre por favor? ")
ape = input("¿Me puedes dar tu apellido por favor? ")
print("Gracias.")
print("\nTu nombre es " + nom + " " + ape + ".")
```

Replicación

El signo de * (asterisco), cuando es aplicado a una cadena y a un número (o a un número y cadena) se convierte en un **operador de replicación**.

cadena * número

número * cadena

Replica la cadena el numero de veces indicado por el número.

Por ejemplo:

"James" * 3 nos da "JamesJamesJames".

3 * "an" nos da "ananan".

5 * "2" (o "2" * 5) da como resultado "22222" (no 10).

Estructura condicional simple

La palabra clave **if** indica que estamos en presencia de una estructura condicional; seguidamente disponemos la condición y finalizamos la línea con el carácter dos puntos **:**. La actividad dentro del **if** se indenta generalmente a 4 espacios. Todo lo que se encuentre en la rama del verdadero del **if** se debe disponer a 4 espacios corrido a derecha. La indentación es una característica obligatoria del lenguaje Python para codificación de las estructuras condicionales, de esta forma el intérprete de Python puede identificar donde finalizan las instrucciones contenidas en la rama verdadera del **if**.

- Ingresar el sueldo de una persona, si supera los \$30000 mostrar un mensaje en pantalla indicando que debe abonar impuestos.

```
sueldo=int(input("Ingrese cual es su sueldo:"))
if sueldo>30000:
    print("Esta persona debe abonar impuestos")
```

Estructura condicional compuesta

- Realizar un programa que solicite ingresar dos números distintos y muestre por pantalla el mayor de ellos.

```
num1=int(input("Ingrese primer valor:"))
num2=int(input("ingrese segundo valor:"))
print("El valor mayor es")
if num1>num2:
    print(num1)
else:
    print(num2)
```

Operadores Relacionales

== Igualdad

!= Desigualdad

< menor

<= menor o igual

>Mayor

>= mayor o igual

Estructura condicional anidada

Operadores Lógicos

- OR
- AND
- NOT

Problemas propuestos

1. Realizar un programa que solicite la carga por teclado de dos números, si el primero es mayor al segundo informar su suma y diferencia, en caso contrario informar el producto y la división del primero respecto al segundo.
2. Se ingresan tres notas de un alumno, si el promedio es mayor o igual a siete mostrar un mensaje "Promocionado".
3. Confeccionar un programa que pida por teclado tres notas de un alumno, calcule el promedio e imprima alguno de estos mensajes:
Si el promedio es ≥ 7 mostrar "Promocionado".
Si el promedio es ≥ 4 y < 7 mostrar "Regular".
Si el promedio es < 4 mostrar "Reprobado".