Introducción a Python

Índice

Índice de imágenes

1. Conceptos básicos

a. **Computadora:** se trata de un aparato capaz de ejecutar una serie de órdenes que permiten resolver un problema. Una computadora se conoce como una "máquina algorítmica".

b. **Algoritmo:** Opera sobre la base de tres elementos obvios (comienza sobre los datos del problema; suponiendo que se aplican sobre ellos los pasos, procesos o instrucciones de los que consta el algoritmo; obteniendo finalmente los resultados esperados). Los algoritmos tienen una serie de propiedades que son las siguientes:
   i. El conjunto de pasos definido para el algoritmo debe tener un ***final previsible***.
   ii. Una vez se plantea un algoritmo para una tarea o problema dado, ***no es necesario volver a pensar la solución*** del problema cada vez que se desee obtener sus resultados.
   iii. Un algoritmo consiste en una combinación de pasos básicos más pequeños, que eventualmente pueden ser automatizados.

c. **Tipo:** Python es un lenguaje de *tipado dinámico*, lo que significa que una misma variable puede recibir y almacenar valores de tipos diferentes durante la ejecución de un programa, u por lo tanto las variables no deben declararse en base a un tipo específico prefijado antes de ser usadas.
   Una variable en Python se define en el momento en que se le asigna un valor (de cualquier tipo) por primera vez, y el tipo asumido apra esa variable es el que corresponda al valor que se le asignó.

Tipo	Descripción	Rango
bool	Valores lógicos	[True, false]
Int	Números enteros	Parte entera
float	Numeros reales	Hasta 15 decimales
Str	2*nºCaracteres	Unicode

Tabla - Tipos de datos más corrientes en Python 3

d. **Variable:** Grupo de bytes asociado a un nombre o identificador, de tal forma que a través de dicho nombre se puede usar o modificar el contenido de los bytes asociados a la variable.
   Una variable puede cambiar de tipo si se le asigna un valor de un tipo diferente al que tenía inicialmente, pues Python es un lenguaje de tipado dinámico, sin que esto provoque ninguna clase de error, y puede hacerse esto tantas veces como se quiera o necesite el programador.
   Puede cambiar de tipo, pero lo que no se puede hacer es intentar usar una variable sin haberle asignado previamente un valor inicial alguna vez; aunque en algunas situaciones puede esperarse que una variable sea asignada con el valor *none* (equivalente a False), indicando que la variable es válida, pero carece de valor en ese momento.
   El nombre o identificador de una variable es elegido por el propio programador, pero para ello deben seguirse ciertas reglas básicas:
   i. El nombre o identificador de una variable en Python sólo debe contener letras, dígitos y guion bajo (_).
   ii. El nombre de una variable no debe comenzar con un dígito.

iii. Las palabras reservadas del lenguaje Python no puede usarse como nombre de variables.
iv. EL nombre de una variable puede contener cualquier cantidad de caracteres de longitud.
v. Python es *case sensitive*: hace diferencia entre mayúsculas y minúsculas.

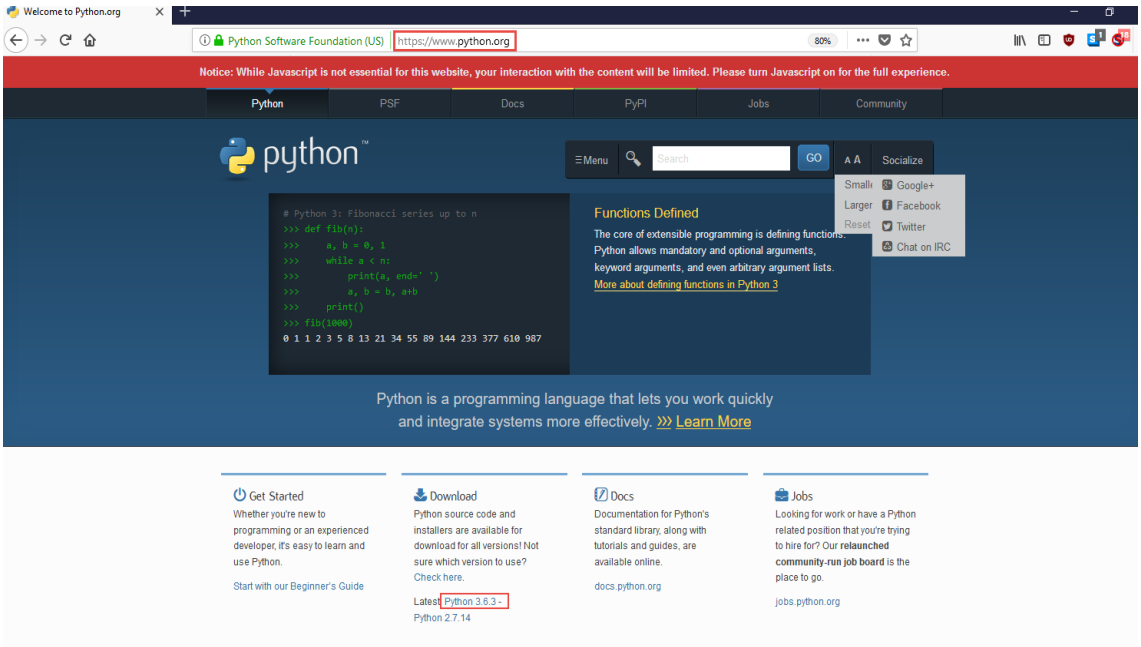
Identificador	Observaciones
n1	Válido en Python 2 y Python 3
nombre_2	Válido en Python 2 y Python 3
sueldo_anterior	Válido en Python 2 y Python 3
x123	Válido en Python 2 y Python 3
año	Válido en Python 3
número	Válido en Python 3

Tabla - Ejemplos de variables válidas en cada versión de Python

e. **Expresión:** Fórmula en la cual se usan operadores (como suma o resta) sobre diversas variables y constantes. Si el resultado de la expresión es un número, entonces la expresión se denomina "*expresión aritmética*". Una variable es el resultado de una expresión.

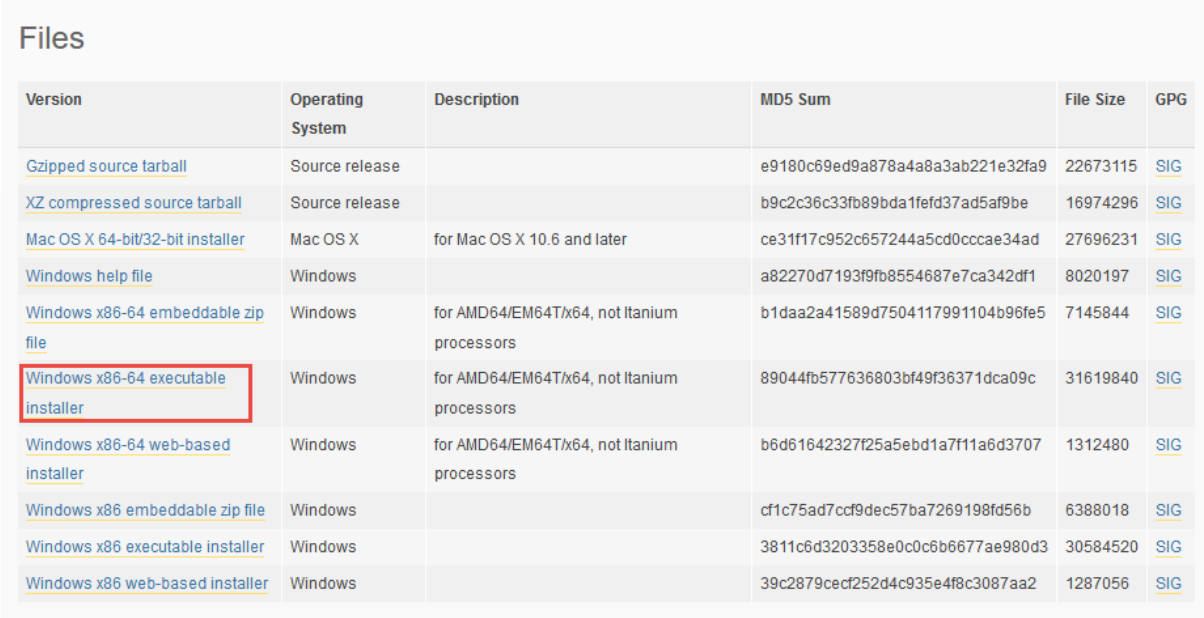
2. Instalación de Python 3

Ir a la página web desde donde se descargará el programa. La dirección web es (https://www.python.org). Una vez en ella, en la página principal, deslizar la barra hacia abajo y en el apartado en el que pone "*Downloads*", seleccionar la opción de "Python 3.6.3-", justo detrás de la palabra "*lastest*" (indicando que es la última versión subida para descargar).



Instalación Python 3(https://www.python.org) - 1

En la nueva ventana que se abre, deslizar la barra del navegador hacia abajo hasta el apartado denominado "*Files*" en el cual están todos los paquetes correspondientes a la versión 3.6.3 de Python en todos y cada uno de los formatos disponibles, se escogerá para esta instalación el paquete llamado "*Windows x86-64 executable installer*" (enmarcado en el cuadro rojo), pues es el ejecutable que lanzará la instalación en el sistema operativo Windows 10 de 64bits sobre el que se instalará.

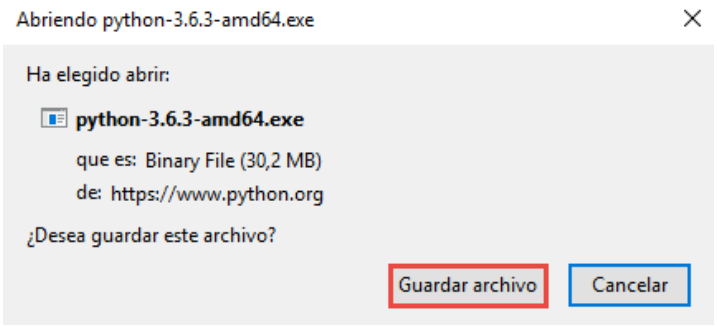


Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		e9180c69ed9a878a4a8a3ab221e32fa9	22673115	SIG
XZ compressed source tarball	Source release		b9c2c36c33fb89bda1fefd37ad5af9be	16974296	SIG
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	ce31f17c952c657244a5cd0cccae34ad	27696231	SIG
Windows help file	Windows		a82270d7193f9fb8554687e7ca342df1	8020197	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64, not Itanium processors	b1daa2a41589d7504117991104b96fe5	7145844	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64, not Itanium processors	89044fb577636803bf49f36371dca09c	31619840	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64, not Itanium processors	b6d61642327f25a5ebd1a7f11a6d3707	1312480	SIG
Windows x86 embeddable zip file	Windows		cf1c75ad7ccf9dec57ba7269198fd56b	6388018	SIG
Windows x86 executable installer	Windows		3811c6d3203358e0c0c6b6677ae980d3	30584520	SIG
Windows x86 web-based installer	Windows		39c2879cecf252d4c935e4f8c3087aa2	1287056	SIG

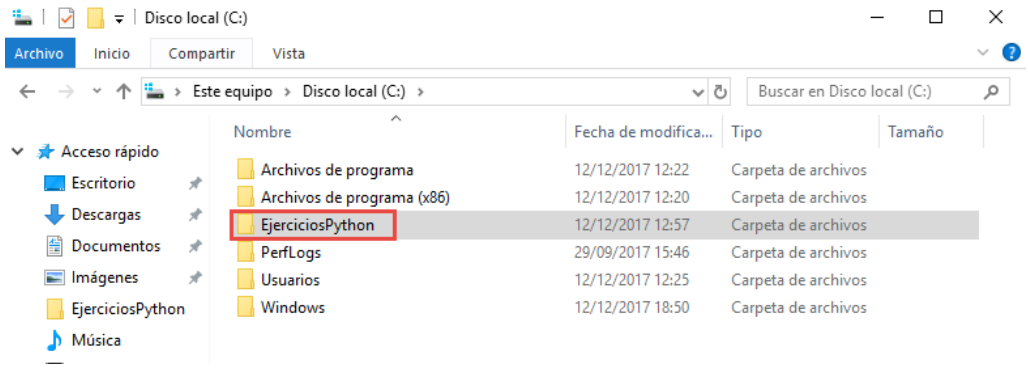
Instalación Python 3 (Elección del paquete) - 2

Se abrirá una ventana en la cual preguntará si se desea guardar o cancelar la descarga del archivo python-3.6.3-amd64.exe que se ha seleccionado para descargar, por lo que se escogerá la opción de "*Guardar archivo*".



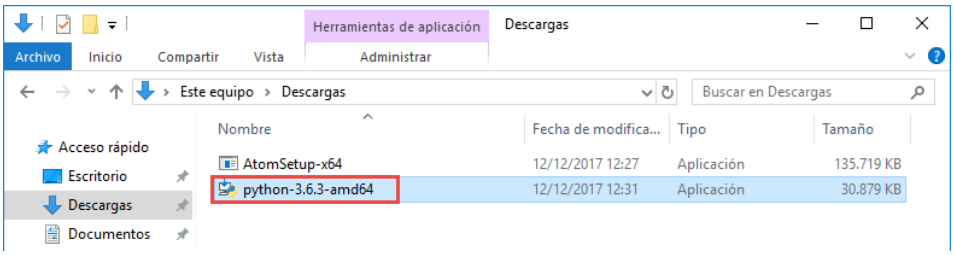
Instalación Python 3(Guardar ejecutable) - 3

Por comodidad para cuando se vayan a lanzar o correr ficheros de Python (.py) desde la terminal de comandos, crear una carpeta en la raíz de la unidad C (C:\), dentro de la cual se almacenarán todos los programas que se creen posteriormente. En este caso, el nombre para la carpeta ha sido "*EjerciciosPython*". (c:\EjerciciosPython).



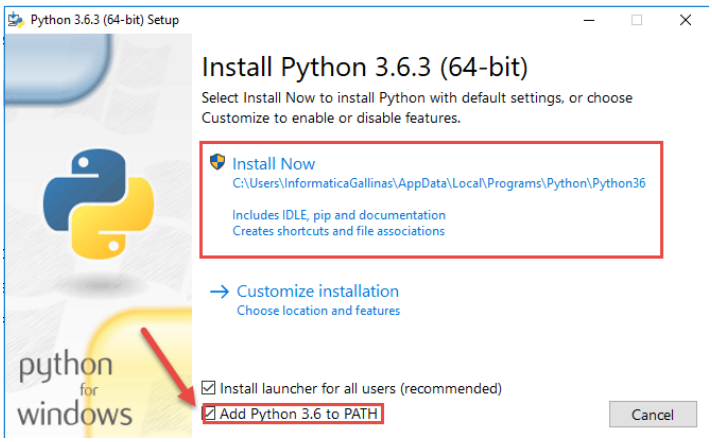
Instalación Python 3 (Crear carpeta ficheros .py) - 4

Creada la carpeta, ir a la carpeta de "Descargas" dentro de "Acceso rápido" en el menú del explorador de Windows, (c:\user\<nombreUsuario>\Descargas). Seleccionar el ejecutable recién descargado (Python-3.6.3-amd64) y con doble click del botón izquierdo arrancará el instalador.



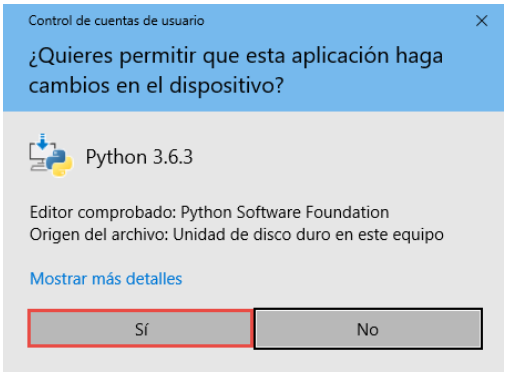
Instalación Python 3(Lanzar el ejecutable) - 5

Seleccionar el checkbox llamado "*Add Python 3.6 to PATH*" (permitirá lanzar Python desde la terminal sin tener que meter la ruta del fichero ejecutable de Python en el PATH) y seleccionar la opción de "*Install Now*".



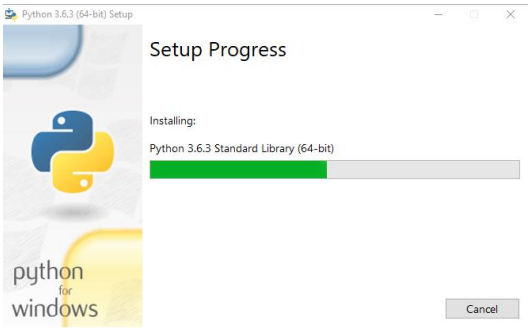
Instalación Python 3 (Inicio Instalación) - 6

Antes de comenzar la instalación se bloqueará la pantalla y aparecerá una ventana en la cual pedirá permisos de administración para instalar el paquete, seleccionar la casilla "Si" y dejar que comience la instalación.



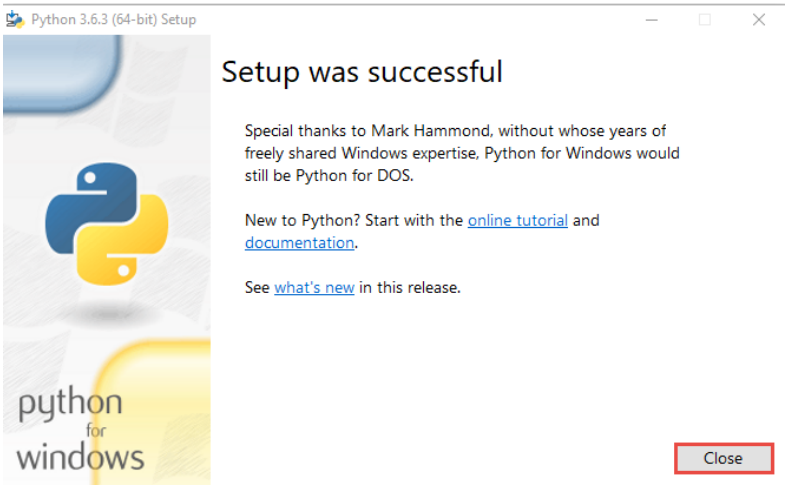
Instalación Python 3 (Permiso administrador para instalar) - 7

Dejar que el proceso de instalación continúe, no suele tardar mucho.



Instalación Python 3(Proceso instalación) - 8

Una vez finalizada la instalación, se podrá acceder a tutoriales online, documentación o noticias de nuevos lanzamientos. Para cerrar la ventana del instalador seleccionar la pestaña "*close*".



Instalación Python 3 (Final instalación) - 9

3. Visualizar por consola y carga de datos por teclado

a. *Visualizar por consola*

La función predefinida ***print()*** permite mostrar en pantalla tanto el contenido de una variable como también mensajes formados por cadenas de caracteres.

Para mostrar el valor de una variable***, el nombre de la misma no lleva comillas ni apóstrofes***, pues en ese caso se tomaría al nombre en forma literal.

print(a) → Muestra el valor de la variable a

print('a') → Muestra literalmente la letra a

print('cadena') → Muestra literalmente la cadena de caracteres entrecomillada.

print('cadena', a) → concatena la cadena de caracteres entrecomillada literalmente seguida del valor de la variable a

Ejercicio 1: Crear un fichero con el nombre holaMundo.py que muestre por pantalla el mensaje "Hola Mundo!!!..." tras su ejecución.

b. *Carga de datos por teclado*

La función ***input()*** permite realizar con comodidad operaciones de carga de datos desde el teclado mientras el programa se está ejecutando.

Permite tomar como parámetro una cadena de caracteres que acompañará en la consola al *prompt* o *cursor* que indique que se está esperando la carga de datos. Lo que sea que el usuario cargue, será retornado por la función *input()* en forma de cadena de caracteres.

nom = input('Inserte su nombre: ') → *Pedirá insertar el nombre por teclado y tras presionar el botón "enter", se asignará esa cadena a la variable nom*

Al igual que cadena de caracteres, se puede realizar carga de datos por teclado de números enteros o números en coma flotante (con decimales), debiendo usarse 2 funciones predefinidas en Python para ello:

a. int(cadena) → Retorna el número entero de la cadena. Así, para pedir que se introduzca un numero por teclado de forma correcta se usaría la función ***int(input('Introduzca un número: '))***
b. float(cadena) → Retorna el número en coma flotante representado por la cadena. Así, para pedir que se introduzca un numero por teclado con decimales se usaría la función ***float(input('Introducir temperatura: '))***

Ejercicio 2: Crear un fichero con el nombre holaNombre.py en el cual se pida introducir por teclado el nombre de la persona que lo está ejecutando y muestre por pantalla un mensaje de saludo igual a "Hola <nombre>".

Ejercicio 3: Crear un fichero con el nombre salNom.py en el cual se pida introducir por teclado el nombre de la persona que lo está ejecutando, la edad que tiene dicha persona y el sueldo que recibe mensualmente con 2 decimales. Mostar por pantalla todo ello con el formato "<nombre> de <edad> años cobra <sueldo> € al mes".

4. Operadores Aritméticos

Operador	Significado	Ejemplo de uso
+	Suma	a=b+c
-	Resta	a=b-c
*	Producto	a=b*c
/	División coma flotante (decimales)	a=b/c
//	División entera	a=b//c
%	Resto de una división	a=b%c
**	Potencia	a=b**c

Tabla - Operadores aritméticos básicos en Python

Los operadores de *suma* (**+**), *resta* (**-**) y *producto* (*****), funcionan tal y como se esperaría.

Ejercicio 4: Crear un fichero con el nombre sumEnt.py en el cual se pida introducir por teclado dos números enteros los cuales se sumen, mostrando el resultado por pantalla con el formato "<num1> + <num2> = <resultado>".

Ejercicio 5: Crear un fichero con el nombre sumDec.py en el cual se pida introducir por teclado dos números decimales los cuales se sumen, mostrando el resultado por pantalla con el formato "<num1> + <num2> = <resultado>".

Ejercicio 6: Crear un fichero con el nombre entSRP.py en el cual se pida introducir por teclado dos números enteros los cuales se sumen, resten y multipliquen, mostrando el resultado por pantalla con el formato "<num1> +|-|* <num2> = <resultado>".

Ejercicio 7: Crear un fichero con el nombre decSRP.py en el cual se pida introducir por teclado dos números decimales los cuales se sumen, resten y multipliquen, mostrando el resultado por pantalla con el formato "<num1> +|-|* <num2> = <resultado>".

Python provee 2 operadores diferentes para el cálculo de la *división*: (**/**) para calcular el cociente entre dos números con decimales; (**//**) para calcular el cociente entre dos números, pero obteniendo sólo la parte entera.

Un operador muy útil es el que permite calcular el *resto* o *módulo* de una división (**%**). En general, se aplica sobre operandos de tipo entero, aunque también se puede aplicar sobre operadores *float*. Para usar el operador resto no es necesario usar antes el operador división, ambos operadores se pueden aplicar sin tener que usar el otro.

Ejercicio 8: Crear un fichero con el nombre divNum.py en el cual se pida introducir por teclado dos números decimales los cuales se dividan, mostrando el resultado de la división con decimales, entera y el resto por pantalla con el formato "<num1> /|//|% <num2> = <resultado>".

El operador *potencia* (**) permite en forma muy simple, calcular el valor de x^y para cualquier valor entero o flotante.

Ejercicio 9: Crear un fichero con el nombre potEnt.py en el cual se pida introducir por teclado dos números enteros los cuales se potencien, mostrando el resultado por pantalla con el formato "<num1> ** <num2> = <resultado>".

Ejercicio 10: Crear un fichero con el nombre potDec.py en el cual se pida introducir por teclado dos números decimales los cuales se potencien, mostrando el resultado por pantalla con el formato "<num1> ** <num2> = <resultado>".

5. Ejercicios propuestos

Son los mismos ejercicios que se han propuesto anteriormente, pero recopilados al final del tema por si se prefiere poner todo en práctica una vez se haya terminado la lectura del tema.

Ejercicio 1: Crear un fichero con el nombre holaMundo.py que muestre por pantalla el mensaje "Hola Mundo!!!..." tras su ejecución.

Ejercicio 2: Crear un fichero con el nombre holaNombre.py en el cual se pida introducir por teclado el nombre de la persona que lo está ejecutando y muestre por pantalla un mensaje de saludo igual a "Hola <nombre>".

Ejercicio 3: Crear un fichero con el nombre salNom.py en el cual se pida introducir por teclado el nombre de la persona que lo está ejecutando, la edad que tiene dicha persona y el sueldo que recibe mensualmente con 2 decimales. Mostar por pantalla todo ello con el formato "<nombre> de <edad> años cobra <sueldo> € al mes".

Ejercicio 4: Crear un fichero con el nombre sumEnt.py en el cual se pida introducir por teclado dos números enteros los cuales se sumen, mostrando el resultado por pantalla con el formato "<num1> + <num2> = <resultado>".

Ejercicio 5: Crear un fichero con el nombre sumDec.py en el cual se pida introducir por teclado dos números decimales los cuales se sumen, mostrando el resultado por pantalla con el formato "<num1> + <num2> = <resultado>".

Ejercicio 6: Crear un fichero con el nombre entSRP.py en el cual se pida introducir por teclado dos números enteros los cuales se sumen, resten y multipliquen, mostrando el resultado por pantalla con el formato "<num1> +|-|* <num2> = <resultado>".

Ejercicio 7: Crear un fichero con el nombre decSRP.py en el cual se pida introducir por teclado dos números decimales los cuales se sumen, resten y multipliquen, mostrando el resultado por pantalla con el formato "<num1> +|-|* <num2> = <resultado>".

Ejercicio 8: Crear un fichero con el nombre divNum.py en el cual se pida introducir por teclado dos números decimales los cuales se dividan, mostrando el resultado de la división con decimales, entera y el resto por pantalla con el formato "<num1> /|//|% <num2> = <resultado>".

Ejercicio 9: Crear un fichero con el nombre potEnt.py en el cual se pida introducir por teclado dos números enteros los cuales se potencien, mostrando el resultado por pantalla con el formato "<num1> ** <num2> = <resultado>".

Ejercicio 10: Crear un fichero con el nombre potDec.py en el cual se pida introducir por teclado dos números decimales los cuales se potencien, mostrando el resultado por pantalla con el formato "<num1> ** <num2> = <resultado>".

Soluciones a los ejercicios propuestos →
https://mega.nz/#!YyJTESAA!PbLr7ogyXbhdIhs0hYHzwOmE6Z_sVr8WMvQgf3Ag2kE