Introducción A Python

Dominar un lenguaje de programación lleva años de práctica y nadie puede pensar que en 20 horas se puede convertir en un programador experto. Sin embargo, la evolución del aprendizaje en Python es de...

- Introducción
- 1. Iniciación a Python
  - Inicio
  - Instalación y primeros pasos
- 2. Entrando en materia
  - Entrando en materia
  - Programa 7
  - Programa 8
  - Programa 9
  - Programa 10
  - Ejercicios de autoevaluación
  - Soluciones a los ejercicios de autoevaluación
- 3. Llegando al final
  - Llegando al final
  - Programa 11
  - Programa 12

Copyright 2026

- Programa 13
- Programa 14
- Ejercicios de autoevaluación
- Soluciones a los ejercicios de autoevaluación

- Créditos

Copyright 2026

Introducción

Copyright 2026

1. Iniciación a Python

Copyright 2026

1. Iniciación a Python

Inicio

Estimado alumno, sin quien este curso no tendría sentido:

Añadir todas las "palabras" de cualquier lenguaje de programación a un documento de texto, acompañadas de una breve explicación, no debería llevarnos más allá de 40 páginas. Digamos que así es el "diccionario" del lenguaje pero, como cualquier idioma, no se aprende una vez memorizado el diccionario; es necesario practicarlo para entender su lógica, conocer la fuerza de cada palabra, diferenciar los matices de significado y tantas otras cosas que diferencian a los hablantes especialistas de los simples usuarios del idioma.

Pues bien, un lenguaje de programación tiene mucho en común con un idioma, por eso se recomienda leer el problema y practicar cada ejemplo, antes de leer la solución, para poder exprimir el conocimiento encerrado en él. Si se hace, se podrán entender las bases del lenguaje y facilitar el futuro aprendizaje, si no, simplemente será una lectura que nada dejará tras de sí.

Dominar un lenguaje de programación lleva años de práctica y nadie puede pensar que en 20 horas se puede convertir en un programador experto. Sin embargo, la evolución del aprendizaje en Python es de las más rápidas en los lenguajes de programación y, tras este curso, se parte de unas condiciones óptimas para continuar profundizando, bien de forma autónoma, bien con otros cursos.

Espero que este curso, realizado con el máximo cuidado, sea de tu agrado. El autor te desea muchos éxitos en este nuevo mundo del que se siente halagado al ser tu anfitrión.

Copyright 2026

1. Iniciación a Python

Instalación y primeros pasos

Objetivos

- Instalar Python
- Familiarizarse con el entorno de desarrollo
- Aprender las primeras órdenes y estructuras

Reflexión inicial

Antes de nada, debes mirar bien esta imagen:

Copyright 2026



http://conectablog.blogspot.com/2010/08/humor-usuarios-e-informaticos.html

Si no quieres que tus alumnos tengan esa imagen de ti ni tú de ellos, es necesario que siempre les hables en un idioma que, no sólo entiendan sino que puedan contrastar, les pueda interesar... El riesgo de caer en dos mundos paralelos es alto cuando estamos hablando de algo tan especializado o donde se pueden perder fácilmente.

En este curso, las cuestiones especializadas las explico con lenguaje más elevado pero siempre las "traduzco" a algo más comprensible. Algo que recomiendo encarecidamente a quien lo vaya a hacer en un aula.

Cuestiones Didácticas

Para enseñar un lenguaje de programación se recomienda seguir una metodología donde, poco a poco, se vayan introduciendo problemas que no se puedan resolver con el conocimiento actual.

Una vez que el alumno es consciente de que le falta algo, se le explica la nueva estructura sin hacer referencia a ningún problema en concreto para que sea él quien investigue cómo aplicarla a su caso particular. Con eso conseguimos los siguientes objetivos:

- El alumno ve que hay algo que le falta y está mucho más atento a lo que se le presenta.
- Le queda mucho más clara la razón de ser de esa estructura nueva y es capaz de asociarla a una funcionalidad concreta.
- Va familiarizándose con los algoritmos y, antes de presentarla, ya ha pensado qué paso del algoritmo debe realizar.

Le queda mucho más clara la razón de ser de esa estructura nueva y es capaz de asociarla a una funcionalidad concreta.

Por otra parte, es recomendable que los alumnos manejen unos pocos programas a los que se les vayan añadiendo funciones y no realizar muchos distintos. Con esto conseguimos que los alumnos:

- No tengan que buscar un archivo en medio de una carpeta llena de ellos para ver cómo solucionaron un determinado problema, repasar una estructura determinada o encontrar una orden en concreto.
- Repasen con la vista las estructuras del programa cada vez que lo abran para añadirle elementos, eso es importante como repaso.

Repasen con la vista las estructuras del programa cada vez que lo abran para añadirle elementos, eso es importante como repaso.

Es necesario saber leer un programa. Es decir, poder llevar en papel el control de las variables y de la salida por pantalla. Es muy útil a la hora de buscar errores o elegir el valor de ciertas variables. Para trabajar este aspecto podemos realizarlo de dos maneras:

- Cuando tengan algún problema porque lo mostrado en pantalla no es lo que habían pensado, decirles que “ejecuten” ellos el programa para ver dónde se han equivocado.
- Plantear directamente un programa para ejecutar

Plantear directamente un programa para ejecutar

Como es natural, el mandarles esta tarea debe ir acompañado de la explicación de los objetivos buscados con esta actividad:

- Mejorar la comprensión de las estructuras del lenguaje.
- Reducir el tiempo de detección de fallos.

Reducir el tiempo de detección de fallos.

Copyright 2026

A lo largo de este curso se va a seguir esta metodología. Es tuya la decisión de usarla en un futuro.

Algoritmos

¿Qué es un algoritmo?

Esta pregunta en una clase genera varios problemas, los alumnos no suelen saber qué significa.

Modo avanzado: Un algoritmo es el conjunto de pasos necesarios que, realizados en el orden marcado por el mismo, nos conducirán a la solución del problema.

Modo terrestre: Un algoritmo es una especie de “receta” que nos dice qué debemos hacer y cuándo para realizar algo. Por ejemplo: El algoritmo de freír un huevo sería: “poner aceite en una sartén, calentar el aceite hasta que hierva, cascar el huevo con cuidado, abrir la cáscara encima de la sartén a una altura adecuada, esperar que se fría y sacarlo con una rasera”.

Programar no es sino hacer algoritmos que, posteriormente, se traducen a lenguaje de programación. Es tan importante conocer las órdenes como saber dónde ubicarlas, y quien decide esto último es el algoritmo.

Si se ve la configuración del curso, hay una cantidad de programas que van ilustrando temas del lenguaje y todos tienen la misma configuración:

- Descripción del problema
- Materia nueva
- Algoritmo
- Solución
- Explicación
- Comentarios

Te recomiendo encarecidamente que, una vez leída la descripción y la materia nueva, te esfuerces en obtener el algoritmo y que, una vez obtenido, pases a trabajar con el ordenador y le busques solución; sólo una vez resuelto, o intentado, puedes mirarla. Sin el paso del algoritmo, nadie puede programar y te puedes hacer una idea de la importancia de controlar esto si te digo que hay cursos enteros que sólo se dedican a los algoritmos (puedes buscarlo en internet y verás la oferta que hay). Con ese conocimiento, el traducirlo a uno u otro lenguaje es algo inmediato.

El trabajo del alumno en este curso no se puede controlar de la misma forma que en una clase presencial, es necesario que te tomes tu tiempo para trabajar cada programa. No tiene sentido copiar los programas inmediatamente y ver que funcionan, han sido probados y, aunque pudiera

Copyright 2026

haber alguna mejora, lo hacen.

Al igual que yo lo tuve que hacer en su día, ahora es tu momento de pensar y diseñar algoritmos. No hay otra forma de aprender: El conocimiento y la destreza no admiten atajos al pensamiento y a la práctica.

Instalación

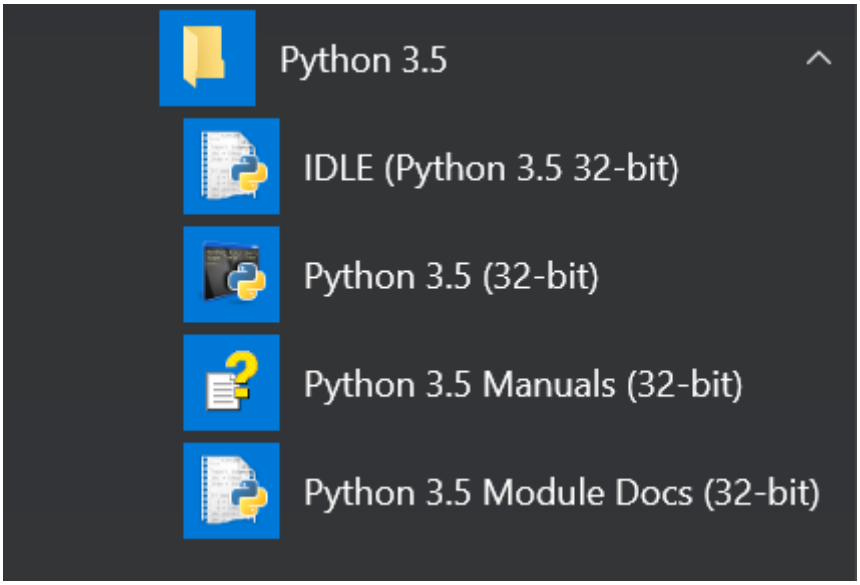
Instalación: Es necesario descargarse el programa de la página web:

www.python.org/downloads

Como se puede ver, hay varias versiones con la misma fecha de actualización. Elegid la 3.5.

Una vez descargado ya tenemos todo disponible para desarrollar.

Presentamos los componentes:



El primero es el que usaremos en este curso

IDLE es el editor de Python por defecto. Podríamos utilizar muchos otros pero es simple y no hay que instalar nada más. Perfecto para comenzar.

Python 3.5 es el intérprete... ¡Un segundo! ¿Intérprete? Sí, es el programa que ejecutará las órdenes que le escribamos. Esto tiene sus ventajas e inconvenientes, como veremos a continuación.

El tercero se refiere a manuales. Es importante leerlos si quieres profundizar más de lo que lo hace este curso. Como es comprensible, en 20 horas se puede dar únicamente una visión del lenguaje y resolver unos cuantos ejercicios simples. Controlar este lenguaje supone años de programación continuada.

El cuarto se refiere a documentos de módulos. En este curso no veremos lo que son por falta de tiempo pero son necesarios porque amplían las posibilidades del lenguaje. Todo lo que son operaciones matemáticas complejas, conexión a internet... vienen por esta vía.

Explicación avanzada:

Existen dos tipos de lenguajes: Los que se interpretan y los que se compilan. Sus diferencias se ven en esta tabla:

Característica	Interpretados	Compilados
Resultado	Programa tipo "texto"	Archivo ejecutable, .exe ...
Privacidad del código	Escasa	Elevada
Multiplataforma	Sí	No. Sólo funcionará en la plataforma donde se haya compilado
Velocidad de ejecución	Baja	Alta

Python es un lenguaje interpretado con un motor muy potente que hace que su velocidad de ejecución sea muy alta y hay varios proyectos para hacer tanto archivos .exe como aplicaciones para Android .apk, el primero funciona bastante bien, el segundo está empezando.

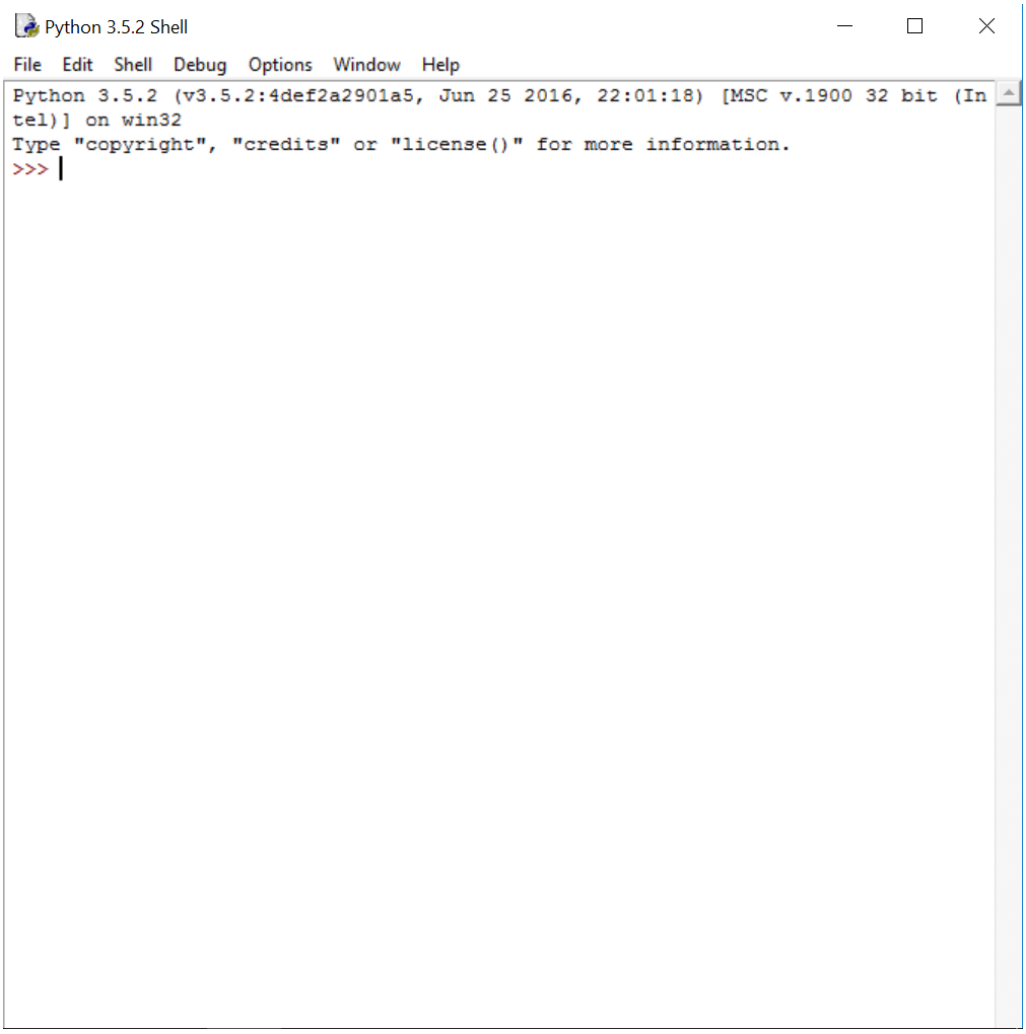
Salimos de la explicación avanzada: Los alumnos rara vez conocen estos problemas y, por tanto, no prestan atención a esto. Convendría dejarles unas pocas ideas claras. Éstas serían:

- Para ejecutar el programa en cualquier ordenador tienen que instalarle Python antes. La ventaja que tienen es que, si lo instalan en un Mac, un móvil, una tablet... también pueden ejecutarlo allí. Es decir, sus programas funcionarán en cualquier sistema.
- No obstante, este lenguaje puede, usando ciertas herramientas, producir aplicaciones para móvil o PC. Lo que sucede es que no es inmediato.

Presentación del sistema de desarrollo

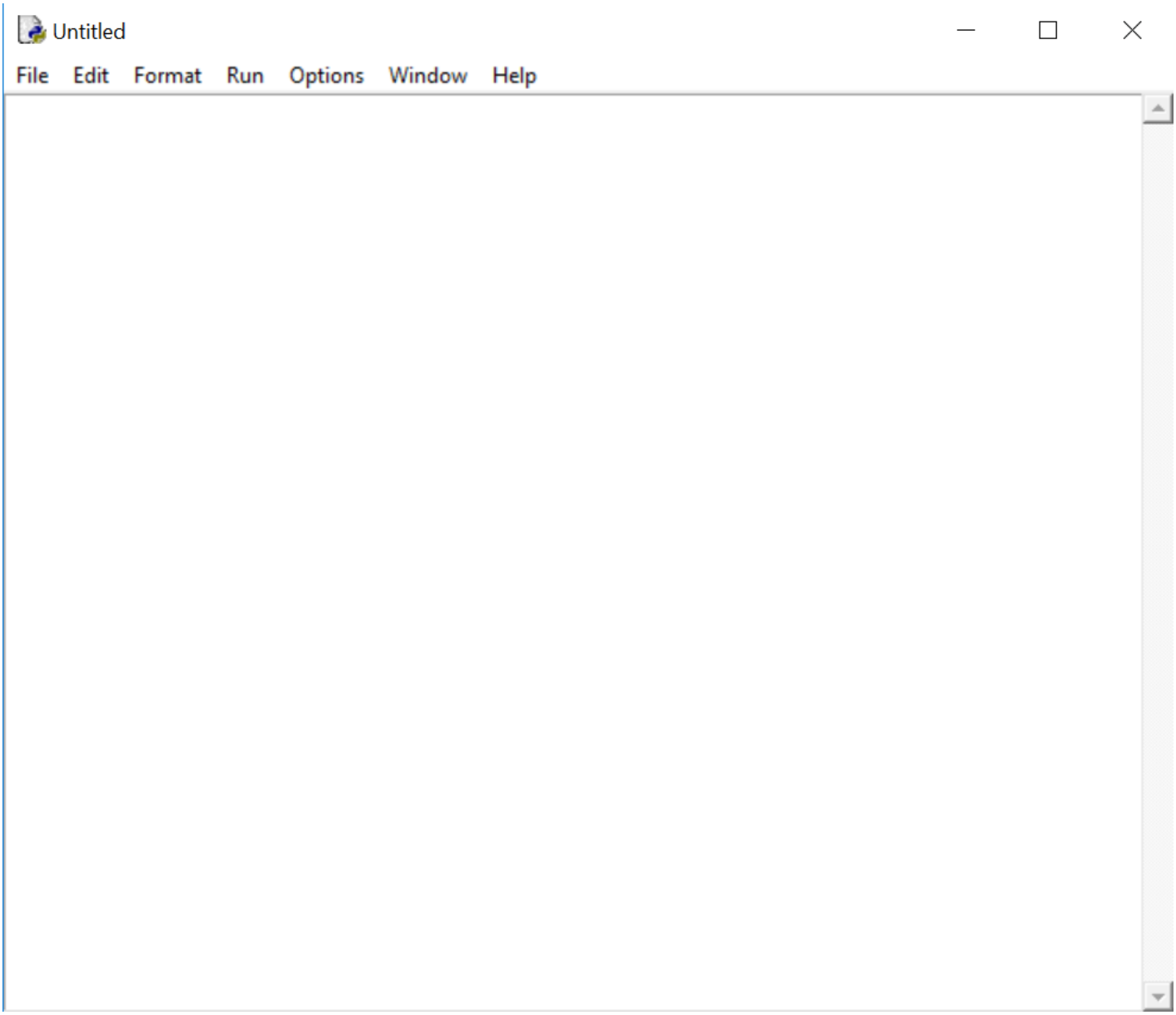
Presentación del sistema de desarrollo:

Basta con entrar en IDLE, del que hemos hablado antes:



Ésta es la pantalla del intérprete, donde se ejecutarán los programas que hagamos. El salto a las pantallas a las que todos estamos acostumbrados con botones... queda fuera del alcance de este curso, no por dificultad sino por tiempo.

Al ir a **File -> New File** se abre el editor de texto, que tiene esta pinta:



La ventaja que tiene es que, a partir de esta pantalla, podemos probar nuestro programa fácilmente con **Run Module** y, al tener un contador de filas, nos dirá dónde hemos cometido los errores, si los hay.

Programa 1

Veamos un primer programa para ver si esto funciona: El famoso “Hola, Mundo”.

Descripción del problema:

Queremos saludar al mundo de la programación en Python sacando por pantalla "Hola, Mundo"

Materia nueva:

print es la orden de sacar por pantalla, puede sacar texto, variables o una combinación de ambas. Si es texto, éste debe ir entrecomillado, si es una variable, no.

Su sintaxis es:

- print ("texto a mostrar") En este caso, sólo mostrará texto
- print (variableamostrar) En este caso, sólo mostrará el valor de la variable
- print("texto a mostrar" + variableamostrar) En este caso mostrará primero el texto y luego la variable.

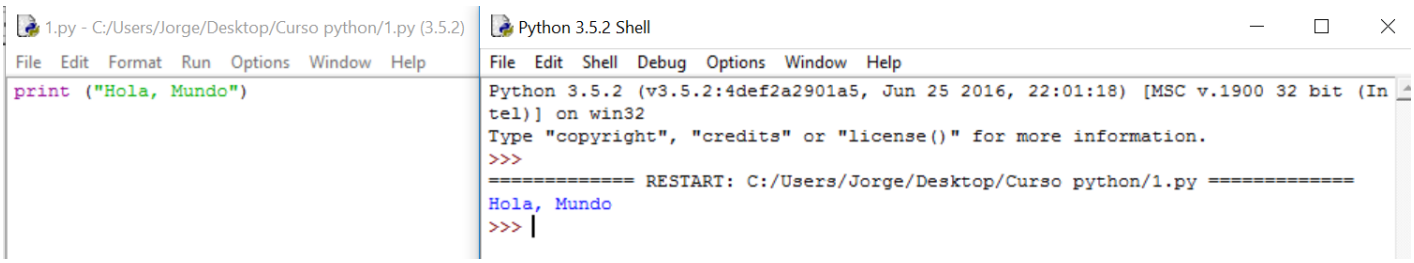
Para ejecutar los programas, es necesario, una vez escritos, ir a **Run** y, una vez allí, elegir **Run Module.**

Solución

Algoritmo:

Mostrar "Hola, Mundo" en pantalla

Solución:



Si todo ha ido bien, ¡Enhorabuena! Ya has dado el mismo primer paso que los más grandes programadores. Si no, bastaría con reinstalar Python, posiblemente habrá habido algún problema al elegir si 32 o 64 bits.

Observa que el propio editor nos ayuda diferenciando mediante colores la función de cada palabra.

Explicación:

print("Hola, Mundo"): Es la orden que hemos nombrado como materia nueva. En este caso tenemos que mostrar un texto, es la primera opción que hemos nombrado de esta orden.

Hola, Mundo es el texto que debe sacar. Por eso está entre comillas, porque tiene que mostrarlo tal cual.

Comentarios:

A veces, como ayuda, en el propio editor sale un texto mientras estamos escribiendo una orden. Es una ayuda que nos indica su sintaxis.

Programa 2

Descripción del problema:

Modificaremos el programa anterior para que nos pregunte el nombre y nos salude de la siguiente forma: "Hola, mengano"

Materia nueva:

Tipos de datos básicos:

Esta materia es necesario darla pero es posible que luego haya que recordarles la existencia de alguno de ellos, sobre todo las booleanas, debido a su escaso uso al principio. De todas formas, sólo se pretende dar una visión general, con programas posteriores se va a ir repasando esto al ir usando los diferentes tipos de variables.

Python 3.5 tiene los siguientes tipos de variables:

- Booleana: Tiene sólo dos valores: True o False, verdadero o falso. Imaginemos que queremos almacenar en una variable
- Numérica: Representa cualquier número pero el lenguaje los divide, como en la vida real en: - Entero - Real - Complejo
- Cadena de caracteres: Representa cualquier símbolo o conjunto de símbolos que se introduzcan por el teclado, aunque luego veremos que hay caracteres especiales, como por ejemplo el que nos indica el final de la línea, final de archivo...

Hay más tipos de datos pero, de momento, es mejor no recargar. Con esto podemos ir haciendo varios programas y, por lo pronto, solucionar el que tenemos entre manos.

Respecto a los tipos de variables:

Modo avanzado: Python es un lenguaje fuertemente tipado.

Modo terrestre: Si una variable es de tipo numérico, no se le puede tratar como cadena de caracteres ni booleana. Python no deja mezclar churras con merinas, así que hay que saber qué tipo tiene la variable que estamos usando para cambiárselo si es necesario.

Otros elementos de teoría necesarios:

Para pedir información existe la función: **input** (“mensaje que se le da al usuario para decirle qué le pedimos”).

Esta función tiene las mismas posibilidades que **print** respecto a mostrar variables o texto. La diferencia entre las dos es que con **print** no esperamos respuesta y con **input** sí, que siempre será una variable tipo Cadena de caracteres.

Para asignarle a una variable esa respuesta, el símbolo de asignación es, como en la vida real, "**=**".

A diferencia de otros lenguajes de programación, Python no requiere que declaremos las variables con antelación, tan sólo hay que usarlas y él se encarga de todo.

Solución

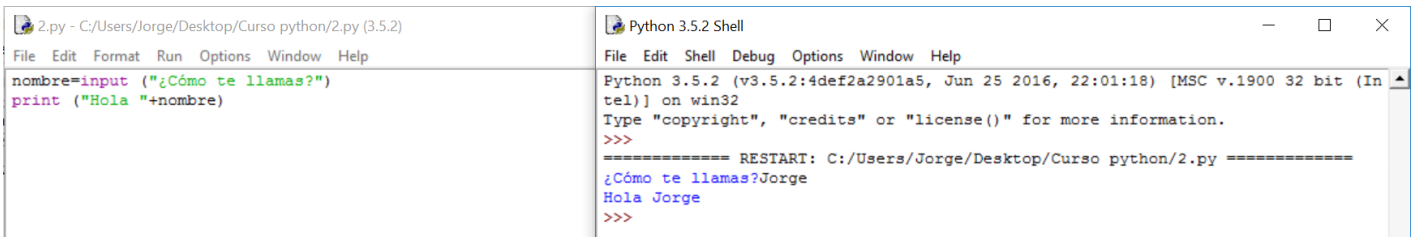
Algoritmo:

1.- Pedimos el nombre. No sabíamos hacerlo hasta ahora, anteriormente sólo hemos mostrado texto sin esperar respuesta, como la necesitamos, usaremos **input** .

2.- Anotamos ese nombre. Si estuviéramos hablando con alguien, lo guardaríamos en un trozo de nuestra memoria y asociaríamos su cara a su nombre. El ordenador hace lo mismo pero no puede asociar caras, así que , en vez de asociar su cara, le pone un nombre. No sabíamos hacerlo hasta ahora pero ya conocemos los tipos de variable y cómo se asignan sus valores.

3.- Saludamos con lo que hemos anotado. Eso ya intuimos que será con la orden **print** porque eso ya sabemos hacerlo.

Solución:



Análisis del programa:

nombre=input("¿Cómo te llamas?") Esta línea es la que hemos empleado para guardar el nombre. En este caso, **nombre** es una variable a la que se le asigna (por medio de **=**) lo que escriba el usuario. **Input** viene del inglés y se traduce como: conjunto de datos que se introducen en un programa o sistema informático. En este caso, **input** pone al ordenador a la espera de recibir texto del teclado. En resumen, la línea entera guarda lo que el usuario escriba en una variable, que ocupará una parte de memoria del ordenador, y a la que nos podremos referir siempre con ese nombre.

print ("Hola, "+nombre) Como vimos en el programa anterior, la función **print** puede sacar por pantalla tanto texto como variables o una mezcla de los dos. Éste es el último caso. Observad que hay un espacio detrás de la "**,**" ya que, si no, lo pondría todo junto. El ordenador no sabe lo que queda bien o mal en un texto.

Comentarios:

- No le hemos dicho de qué tipo de variable estamos hablando. Él, por su cuenta, habrá decidido una. Sabemos que será una Cadena de caracteres por venir de una respuesta a un **input**. La ventaja en Python es que él decide qué tipo de variable utiliza sin que la declaremos y nos ahorra alguna línea que otra. La desventaja es que, para saber de qué tipo es, habrá que preguntarle.

-Debemos elegir los nombres de las variables de forma que nos den una idea de los datos que contienen. Favorecerá la lectura de nuestros programas.

Programa 3

Descripción del problema:

Ahora imaginemos que queremos que queremos modificar el programa anterior para que, una vez preguntado el nombre, si el usario se llama como nosotros, los programadores, le salude de una forma especial: "Eres un figura, mengano". Si no se llama como nosotros, le saludaremos normalmente como antes.

Materia nueva:

Signos de comparación:

Si queremos ver si una variable tiene un valor determinado, mayor, menor... lo normal es usar los signos que usaríamos en matemáticas, aunque hay alguna pequeña sorpresa, y que represento a

continuación:

x==y # x es igual a y. Mucho cuidado con esta comparación, se usan dos iguales seguidos.

x!=y # x es distinto de y

El resto son los tradicionales:

x>y # x es mayor que y

x<y # x es menor que y

x>=y # x es mayor o igual que y

x<=y # x es menor o igual que y

Primera estructura de control: If

If viene del inglés y es un condicional que significa si. El lenguaje usa la siguiente sintaxis:

```
if comparación :
  órdenes
elif comparación:
  órdenes
else:
  órdenes
```

Traduzcamos esto teniendo en cuenta que elif es la abreviatura de else if y que else significa “en otro caso”

```
Si secumplelacondición:
  Haz esto...

En otro caso, si secumpleestaotracondición:
  Haz esto

En el caso de no cumplirse nada de lo anterior:
  Haz esto
```

Copyright 2026

Hay que decir varias cosas:

- Puede haber tantos elif como queramos.
- Las órdenes deben ir identadas o jerarquizadas, es decir, con una tabulación. En cualquier lenguaje, esto siempre se recomendaba para facilitar la lectura del programa pero Python lo exige porque no usa otra forma de agrupar órdenes. Es decir, todo lo que esté en el mismo nivel pertenece a un mismo bloque. Por eso, en Python es común ver la línea izquierda del texto de forma sinuante.
- No te olvides de los dos puntos tras cada comparación o else.

Solución

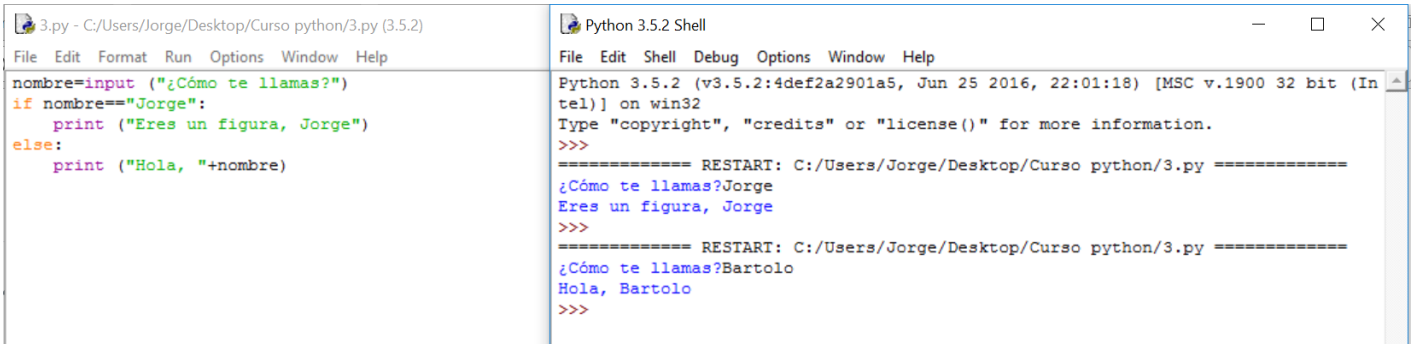
Algoritmo:

1.- Pide el nombre y lo anota. Lo sabemos hacer

2.- Compara ese nombre con el nuestro (que se lo habremos dicho en algún lado del programa, veremos varias posibilidades). No lo sabíamos hacer hasta ahora, usaremos la estructura **if**.

3.- Si coincide, nos saludará de una forma especial, si no, como a todo el mundo. Sabemos hacer la parte del saludo pero, como hay dos formas, intuimos que deberemos ponerle las dos y que elija según el resultado del **if**.

Solución:



Como puede verse en la salida, se ha ejecutado dos veces, una respondiendo con el nombre especial y otra con otro nombre. Cada vez ha entrado en una parte de la estructura if distinta.

Explicación:

La estructura **if** es muy simple en este caso, sólo tenemos que distinguir dos posibilidades, por eso únicamente hay **if** y **else.**

if nombre=="Jorge": le dice: Si lo que la variable nombre contiene es igual (**==**) a la palabra (por eso está entrecomillado) "**Jorge**" entonces sigue lo que te digo tras los dos puntos "**:**"

else: le dice: Si no se ha cumplido la comparación del **if** entonces haz lo siguiente a los dos puntos "**:**".

Comentarios:

Prueba a poner "Jorge", o tu nombre, sin comillas.

Programa 4

Descripción del problema:

Ahora vamos a pedir el nombre y la edad. Si la edad es menor de 25 años, diremos que es un alumno y, si no, que es un profesor. Si la edad es mayor de 65 nos meteremos con él. Como eres tan buen programador y tienes delicadeza con quien usa tus programas (siempre hay que mimar a los usuarios), puedes dirigirte al usuario como quieras pero discriminando si es alumno, profesor o jubilado.

Materia nueva:

En principio no hay, esto es una estructura **if**. Lo que sí sucede es que hay tres opciones y hay que usar una orden **elif**.

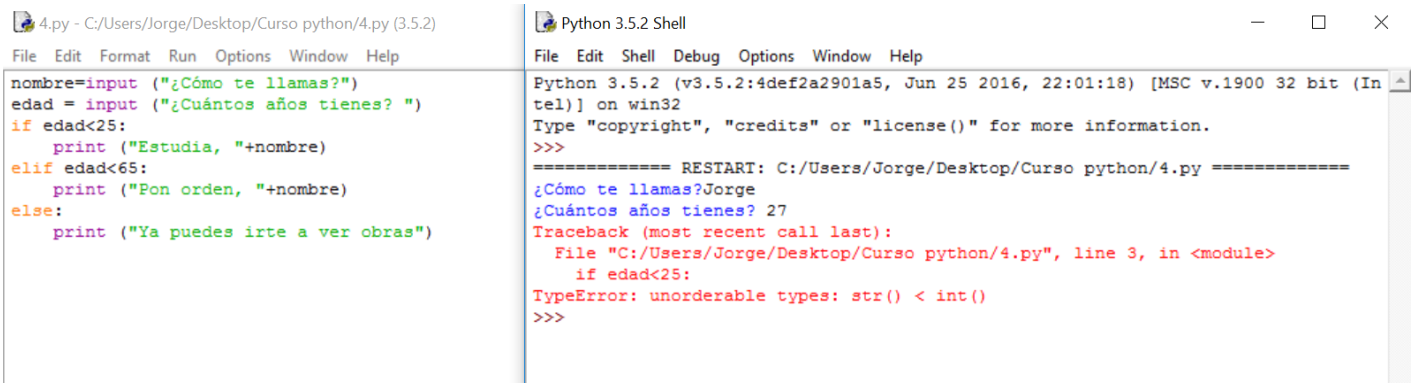
¡Atención! Un problema que se le puede haber ocurrido a alguien es que 24 es menor que 25 y menor que 65. ¿En qué se fijaría en ese momento? La respuesta es sencilla, si está en un if, irá por orden y, si entra en el primero, automáticamente sale de la estructura sin entrar en el siguiente.

Solución

Algoritmo:

1. Pido el nombre
2. Pido la edad
3. Comparo la edad y...
    1. si es menor de 25 le digo que estudie,
    2. si es menor de 65 le digo que ponga orden,
    3. si es mayor o igual a 65 le digo que se vaya

Solución:



¿Qué ha pasado? Sencillamente que quería que vierais un error muy frecuente en programación.

Dijimos que Python era fuertemente tipado y hemos comparado una cadena de caracteres con un número.

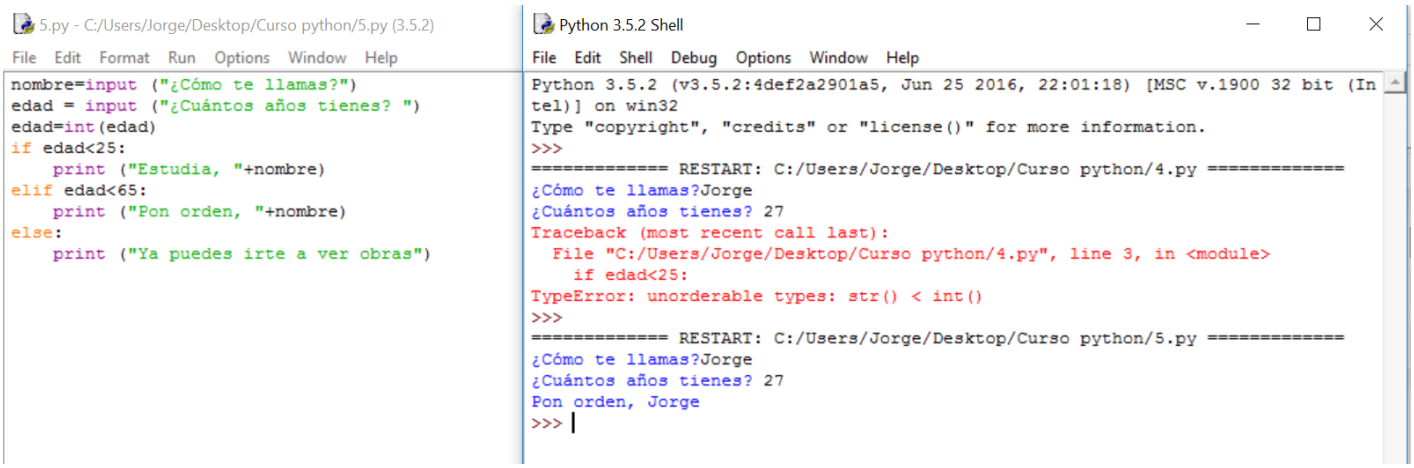
Es muy importante conocer la diferencia entre 2 y “2”. El primero es un número y el segundo es un texto. Algo así como 2 y "dos" para nosotros. Si queremos sumar, no podemos usar la palabra “dos”, debemos usar el número.

Rehagamos el programa pero pasando ese “2” a 2. Para ello, existe una función que vamos a usar:

int(textoquequeramoscambiaranúmero) Si queremos enteros

float(textoquequeramoscambiaranúmero) Si queremos decimales

Cambiemos el programa:



Explicación:

edad=int(edad) es algo muy común en programación si no queremos guardar variables de sobra. En estos casos, se realiza primero la parte derecha y luego guarda el resultado en la variable de la izquierda. En este caso, transforma la variable **edad** en entero y la guarda en el mismo trozo de memoria donde estaba la variable **edad** original. Huelga decir que ese primer valor que tenía lo ha perdido.

Comentarios:

Como ejercicio, prueba también:

- edad=int (input("¿Cuántos años tienes?"))
- edad=float(edad) A ver si hay diferencia si es un entero o no.
- Introduce 24 como edad para comprobar que sólo entra como alumno.
- Introduce "a" como edad, a ver qué pasa.
- Realiza el algoritmo para ajustarlo a esa modificación que hemos realizado después del fallo.

Programa 5

Descripción del problema:

Ahora imaginemos que, en el programa anterior, queremos evitar que pongan una letra en vez de meter un número para evitar fallos y, además, pretendemos seguir preguntándole hasta que lo ponga bien.

Materia nueva:

Para comprobar si puede ser un entero o un decimal, tenemos las siguientes funciones:

cadenaaanalizar.isdigit()

Esta función devuelve True si es un número o False si se trata de texto

Hay muchas funciones más. Para ello es necesario leer los documentos de Python de su página web. Esto queda para quien quiera profundizar.

La siguiente estructura que se va a presentar es la que nos va a permitir hacer algo un **número indefinido de veces:**

```
while condición:
  Órdenes
```

While en inglés significa “mientras”. Por tanto, la traducción sería:

```
Mientras se cumpla la condición:
  Haz esto
```

Observa que se sigue cumpliendo con la identación para el bloque de órdenes.

Solución

Algoritmo:

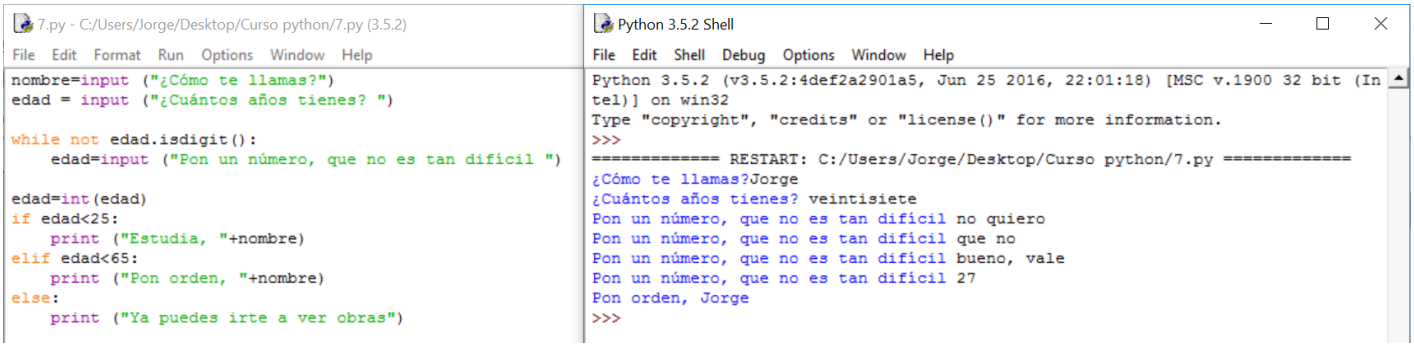
1.- Pido el nombre

2.- Pido la edad

3.- Compruebo que se puede convertir en número y, si no puede, repito la pregunta hasta que pueda. Hasta ahora no lo sabíamos hacer pero en esta frase tenemos la condición que se debe cumplir para que **while** haga o no la pregunta.

4.- Comparo la edad y, si es menor de 25 le digo que estudie, si es menor de 65 le digo que ponga orden y si es mayor o igual a 65 le digo que se vaya

Solución:



Explicación:

En este caso, como la orden que he puesto dentro de la estructura **while** se ejecutará sólo si es cierta y lo que quiero es que lo haga si no es cierta. La he negado con un "**not**". También se podría

haber hecho de otra forma (usando otra variable y una estructura **if**) pero es un engorro. Toma nota de esta negación porque se usa mucho.

El hecho de poner la pregunta dentro de la estructura **while** permite que cada vez haya un valor distinto para comparar. Si no das la oportunidad de que el usuario corrija su error con una nueva respuesta, la comparación contenida en **while** siempre valdrá lo mismo y tendrás un fantástico bucle infinito. Realmente ningún programador puede considerarse tal sin haber realizado varios en su vida. Al final se les pilla cariño, es más emocionante si lo haces en un servidor y puedes tirar tu web y la de todos los que estén alojados allí, pero en un ordenador también es entrañable. Es broma, es deseable que no te pase pero es inevitable.

La idea del bucle infinito sería:

```
while not edad.isdigit():
  print ("No has puesto un número")
```

En este caso, si **edad** no es un dígito, entra en el bucle **while** pero **print** no pregunta y **edad** seguirá sin serlo para cuando vuelva a **while** y, así, indefinidamente. El flujo es: **while** - Comparación - Verdadero - **print** - **while** - Comparación - Verdadero - ...

De la forma que lo hemos puesto: **while** - Comparación - Verdadero - **input** - Nuevo valor de **edad** - **while** - Comparación - Verdadero o falso ya que **edad** ahora es distinto, le hemos dado al usuario una oportunidad de verdad - ...

Comentarios:

Realiza un bucle infinito: Cambia el **input** de la solución por un **print**, al que habrá que quitarle la variable que precedía al **input**.

La forma de salir de algo así es, si no te avisa y no te permite cancelarlo cerrando el intérprete, como siempre en el mundo de la informática: **Ctrl+Alt+Supr**

Programa 6

Descripción del problema:

Vamos a realizar una de las muchas variaciones que se pueden hacerle al programa anterior, por ejemplo, contar las veces que se ha preguntado la edad y decírselas al usuario.

Materia nueva:

Convertir un número en texto se realiza con la orden:

str(númeroaconvertir)

En realidad, no sólo convierte números sino un montón de cosas más. De momento lo emplearemos con números pero os adelanto que lo volveremos a ver.

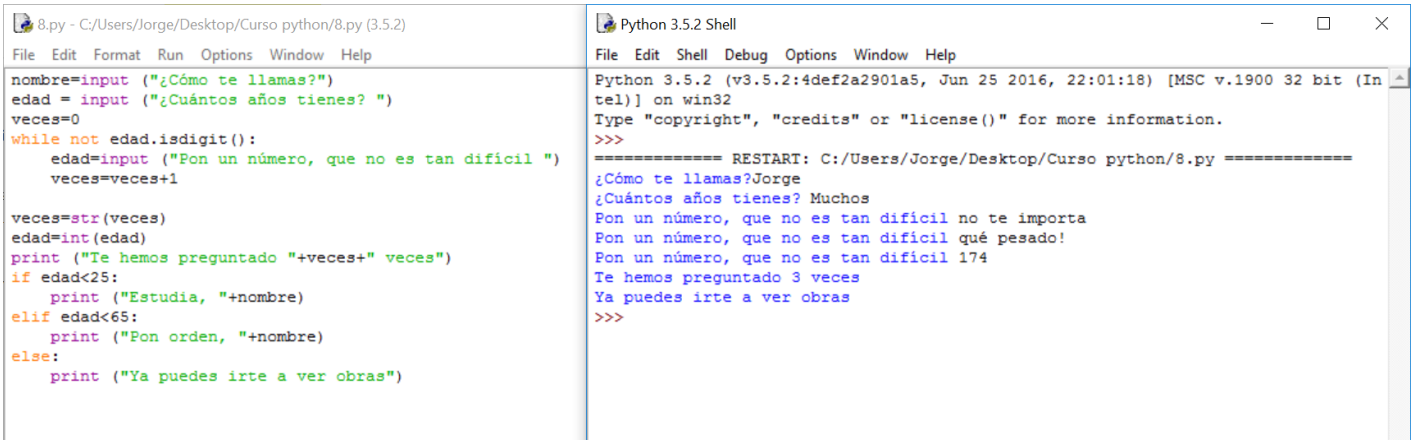
Los **contadores** son variables que almacenan un número que lleva la cuenta de algún proceso... Hay estructuras que veremos un poco más adelante que no pueden existir sin ellos. De momento, quedaos con que es una función tan común para una variable que se le denota de una forma propia.

Solución

Algoritmo:

1. Pido el nombre
2. Pido la edad
3. Compruebo que se puede convertir en número y, si no puede, repito la pregunta hasta que pueda. Voy guardando en una variable las veces que hago esto.
4. Le digo las veces que se lo he preguntado. Como debo convertir el número en texto para sacarlo por pantalla, usaré **str(**variable**)**.
5. Comparo la edad y...
    1. si es menor de 25 le digo que estudie,
    2. si es menor de 65 le digo que ponga orden,
    3. Si es mayor o igual a 65 le digo que se vaya.

Solución:



Explicación:

Recuerda que **veces=veces+1** consiste en que el ordenador hace la parte derecha antes y la guarda como la variable de la izquierda. No hagas el cálculo matemático y despejes: 0=1. Sabes que se tiene que leer:

La nueva **veces** es igual a la antigua **veces** más uno

Comentarios:

Quedan como ejercicios:

- Buscar otra forma de sumarle 1 a una variable en Python más abreviada. En otros lenguajes hay cosas como veces++, busca en internet si hay algo así. Para esto hay una página en concreto que es una referencia en preguntas y respuestas: http://es.stackoverflow.com
- Probar a mostrar el número de veces sin haber transformado la variable en texto.
- Probar qué pasa si ponemos el número sin fallar ninguna vez y solucionar el dato erróneo que da.
- Probar a hacerlo de tal forma que, cada vez que pregunte, diga las veces que lleva. Hay muchas formas de hacerlo pero te puede servir de ayuda el hecho de que str(veces) no convierte la variable "veces" en texto si no se guarda, simplemente devuelve una Cadena de caracteres a partir del valor de "veces".

Probar a mostrar el número de veces sin haber transformado la variable en texto.

Ejercicios de autoevaluación

Felicidades por haber terminado el Módulo 1.

Ahora toca practicar lo aprendido con varios ejercicios que deben resolverse con lo aprendido. Se recomienda encarecidamente emplear tiempo en resolverlo en vez de ir a ver la solución directamente.

1.- Realiza un programa donde se pida dos números y la operación a realizar (a elegir entre suma, resta, multiplicación y división). Posteriormente debe dar el resultado y preguntar si queremos hacer otra operación. Identificar si se quiere realizar una división por 0.

Ten en cuenta que los operadores matemáticos simples son:" + " para la suma," - " para la resta," * " para la multiplicación y " / " para la división. En este programa se recomienda convertir el texto que devuelve **input** a un número tipo **float.**

2.- Realiza un programa donde el usuario piense un número entre 1 y 100 y el ordenador, preguntando si es mayor o menor que otro, debe averiguarlo.

Ten en cuenta que la mejor estrategia es preguntar cada vez por la mitad del rango disponible para eliminar la mayor cantidad posible de números:

Primero preguntaré por 50 y, así elimino a 50 de golpe, una vez ahí, si es mayor, preguntaré por 75...

Información extra: El cociente de la división (o, lo que es lo mismo, el resultado entero de la división) se realiza mediante el operador "//"

5//2 nos dará 2.

Sé muy cuidadoso a la hora de hacer el algoritmo y elegir las variables. Ten en cuenta que los límites superior e inferior deberán cambiar a lo largo de la ejecución del programa.

Este programa puede requerir el uso de una variable booleana. Se declara como sigue:

variable=True o, alternativamente: **variable=False**

3.- Realiza un programa donde se le pidan números al usuario hasta que introduzca 3 impares. Si son pares, seguirá pidiendo indefinidamente. Cada vez que se introduce un impar, el programa nos dice cuántas oportunidades nos quedan antes de terminar salvo al llegar a la tercera vez, debe decir que se termina el programa.

En este caso, diferenciar si es par o impar lo podemos hacer con la operación que nos da el resto de una división. Su sintaxis es: **%**

4 **%** 2 = 0

5 **%** 2 = 1

Se le llama módulo. Nada que ver con el módulo de un vector en matemáticas.

4.- Realiza un programa donde pida números indefinidamente y pare cuando el número que se ha introducido sea menor a la suma de los dos anteriores.

Es un problema donde te van a hacer falta variables que guarden esos números e ir actualizándolos cada vez que incluya uno. Va muy bien cara a practicar el diseño de algoritmos y a empezar a "leer" los programas, algo que se enseñará en el segundo módulo.

Copyright 2026

NOTA: Las soluciones no son únicas. Los programas que realices pueden ser igualmente correctos aunque sean más largos, usen más variables, etc.

Soluciones a los ejercicios de autoevaluación

Autoeval 1.py - C:/Users/Jorge/Desktop/Curso python/Autoeval 1.py (3.5.2)

File Edit Format Run Options Window Help

```
continuar= "S"

while continuar=="S":
    num1=float(input ("Indica el primer número: "))
    num2=float (input ("Indica el segundo número: "))
    operac=input ("Indica la operación que quieres: Suma: s, Resta: r, Multi
    if operac=="s":
        print (str(num1+num2))
    elif operac=="r":
        print (str(num1-num2))
    elif operac=="m":
        print (str(num1*num2))

    else:
        if num2!=0:
            print(str(num1/num2))
        else:
            print ("No se puede dividir por 0")
    continuar=input("¿Quieres volver a operar? S/N ")
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:/Users/Jorge/Desktop/Curso python/Autoeval 1.py =========
Indica el primer número: 3.4
Indica el segundo número: 0
Indica la operación que quieres: Suma: s, Resta: r, Multiplicación: m, División:
 d : d
No se puede dividir por 0
¿Quieres volver a operar? S/N S
Indica el primer número: 3.4
Indica el segundo número: 2.354
Indica la operación que quieres: Suma: s, Resta: r, Multiplicación: m, División:
 d : d
1.4443500424808835
¿Quieres volver a operar? S/N S
Indica el primer número: 23.156
Indica el segundo número: 31.546
Indica la operación que quieres: Suma: s, Resta: r, Multiplicación: m, División:
 d : m
730.4791759999999
¿Quieres volver a operar? S/N S
Indica el primer número: 3.1564
Indica el segundo número: 45.643
Indica la operación que quieres: Suma: s, Resta: r, Multiplicación: m, División:
 d : r
-42.4866
¿Quieres volver a operar? S/N N
>>>
```

Autoeval 2.py - C:\Users\Jorge\Desktop\Curso python\Autoeval 2.py (3.5.2)

File Edit Format Run Options Window Help

```
continuar= True
limitebajo=0
limitealto=100
num=50

while continuar:
    respuesta=input ("¿El número es "+str(num)+"?. Indicar S si es mayor y N
    if respuesta=="S":
        limitebajo=num
        num=(num+limitealto)//2
    elif respuesta=="N":
        limitealto=num
        num=(num+limitebajo)//2
    else:
        continuar=False
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:\Users\Jorge\Desktop\Curso python\Autoeval 2.py =========
¿El número es 50?. Indicar S si es mayor y N si es menor. Si he acertado, indica
rlo con A. S
¿El número es 75?. Indicar S si es mayor y N si es menor. Si he acertado, indica
rlo con A. N
¿El número es 62?. Indicar S si es mayor y N si es menor. Si he acertado, indica
rlo con A. S
¿El número es 68?. Indicar S si es mayor y N si es menor. Si he acertado, indica
rlo con A. S
¿El número es 71?. Indicar S si es mayor y N si es menor. Si he acertado, indica
rlo con A. S
¿El número es 73?. Indicar S si es mayor y N si es menor. Si he acertado, indica
rlo con A. N
¿El número es 72?. Indicar S si es mayor y N si es menor. Si he acertado, indica
rlo con A. A
>>>
```

Copyright 2026

Autoeval 3.py - C:/Users/Jorge/Desktop/Curso python/Autoeval 3.py (3.5.2)

File Edit Format Run Options Window Help

```
errores=0
suma=0
while errores<3:
    numero=int(input("introduce un número par: "))
    if numero%2==1:
        errores=errores+1
        if errores<3:
            print ("Llevas "+str(errores)+" errores. ¡Cuidado!")
        else:
            print ("Ya me he cansado")
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:/Users/Jorge/Desktop/Curso python/Autoeval 3.py =========
introduce un número par: 12
introduce un número par: 34
introduce un número par: 56
introduce un número par: 57
Llevas 1 errores. ¡Cuidado!
introduce un número par: 58
introduce un número par: 59
Llevas 2 errores. ¡Cuidado!
introduce un número par: 45
Ya me he cansado
>>>
```

Autoevaluacion4.py - C:/Users/Jorge/Desktop/Curso python/Autoevaluacio

File Edit Format Run Options Window Help

```
numero1=0
numero2=0
continuar=True
while continuar:
    numero3=int(input("introduce un número: "))
    if numero1+numero2>numero3:
        continuar=False
        print ("Fibonacci debe estar llorando")
    else:
        numero1=numero2
        numero2=numero3
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:/Users/Jorge/Desktop/Curso python/Autoevaluacion4.py ======
introduce un número: 1
introduce un número: 1
introduce un número: 2
introduce un número: 3
introduce un número: 5
introduce un número: 8
introduce un número: 13
introduce un número: 17
Fibonacci debe estar llorando
>>>
```

Copyright 2026

2. Entrando en materia

Copyright 2026

Entrando en materia

Objetivos

- Conocer las diferentes estructuras cíclicas de Python
- Profundizar en los tipos de datos conocidos.

Copyright 2026

2. Entrando en materia

Programa 7

Descripción del problema:

Se va a modificar el programa 6 para saludar a más gente, primero preguntaremos cuánta gente hay e iremos saludando uno a uno comentando su edad como hasta ahora.

Materia nueva:

Cuando sabemos el número de veces que tenemos que hacer algo, aunque podríamos usar **while**, se usa otra estructura:

For significa en inglés “por”. Lo que quiere decir es “por cada uno”. Tiene muchas sintaxis aunque sólo daremos una de momento.

```
for nombredevariableentera in range(númeroinicial, númerofinal):

    Órdenes
```

Hay que decir que, si númeroinicial =0 no hace falta ponerlo y quedaría:

```
for nombredevariableentera in range (númerofinal):
```

¡Atención! Python empieza en el número inicial, imaginemos **range (1,5).** Cuando a cualquier persona le resulta normal contar de uno a cinco de la siguiente manera:

1, 2, 3, 4, 5

A Python no, resulta que él cuenta los números que ha puesto y dice: "No puede ser: 5-1=4. Debo hacerlo cuatro veces" y cuenta de esta manera:

1, 2, 3, 4

He empezado en el 1 por ser range(1,5) . Si hubiera sido **range (**4**)**:

0, 1, 2, 3

Copyright 2026

Como se ve, el último número no lo va a tocar. **Cuidado con esto ya que ocasiona la mayor parte de problemas en los programas**.

Solución

Algoritmo:

1.- Saludo y pregunto cuánta gente hay

2.- Para cada uno de ellos:

2.1.- Pido el nombre

2.2.- Pido la edad

2.3.- Compruebo que se puede convertir en número y, si no puede, repito la pregunta hasta que pueda. Voy guardando en una variable las veces que hago esto.

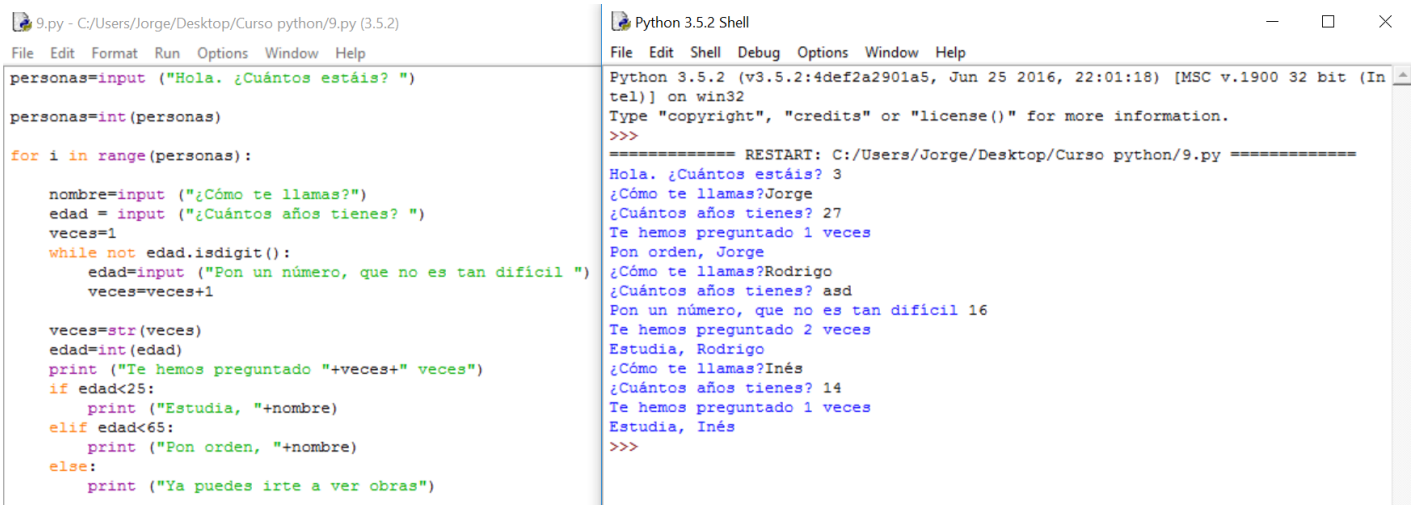
2.4.- Le digo las veces que se lo he preguntado.

2.5.- Comparo la edad y, si es menor de 25 le digo que estudie

Si es menor de 65 le digo que ponga orden

Si es mayor o igual a 65 le digo que se vaya

Solución:



Explicación:

Es necesario darse cuenta de que se ha identado el programa anterior para que forme parte de la estructura for, pues tenía que repetir todo el programa anterior. Menos mal que tenemos la identación para aclarar un poco las diferentes partes que componen el programa.

Si quieres identar una gran cantidad de texto, el editor tiene una opción para identar un montón de líneas a la vez y que no sea necesario hacerlo a mano. Está en **format: indent region**.

Copyright 2026

2. Entrando en materia

Programa 8

Descripción del problema:

Queremos realizar un programa que modifique el número 7. Esta vez sólo queremos saludar pero, en vez de hacerlo uno a uno, lo haremos a todos a la vez, es decir, es necesario que recuerde los nombres y se dirija a ellos con un “Hola, persona1, persona2, persona3... y personaúltima”.

Materia nueva:

Existe otro tipo de datos que se llama List (lista). Una lista es, tal y como se puede pensar, lo mismo que en la vida real salvo que, en Python, el primer elemento siempre es el 0. Imaginemos que queremos preparar una cena. Todo lo que se nos ocurra podemos hacerlo en Python, por ejemplo:

|Vida real|Python| |--|--| |Contar el número de comensales:|len(lista) len es una abreviatura de length| |Decir qué comensal ocupa el número 3, por ejemplo|lista[3] Siempre entre corchetes| |Escoger del 7 al 14|lista[7:14] Tened en cuenta el 0!!| |Añadir uno al final|lista.append(otrocomensal)|

Hay muchas más posibilidades, como se puede ver en:

https://docs.python.org/3/tutorial/datastructures.html

Este tipo de datos hay que declararlos, bien para decir que están vacíos, bien porque los queremos con datos iniciales.

Se declara nombrando la lista y poniendo:

listanueva **[]** Si la queremos vacía para ir llenándola a lo largo del programa.

```
listanueva= ["Lunes","Martes","Miércoles","Jueves","Viernes"]
```

Si queremos los días laborables. Es decir, se introducen los elementos separados por comas.

De momento, nos conformaremos con lo expuesto y resolveremos el problema.

Copyright 2026

Ahora retomaremos a una vieja conocida: **str(**algoqueconvertiratexto**)**

Esta función también puede convertir una lista a texto. Usadlo en este programa: **str (** lista **)**

Solución

Algoritmo:

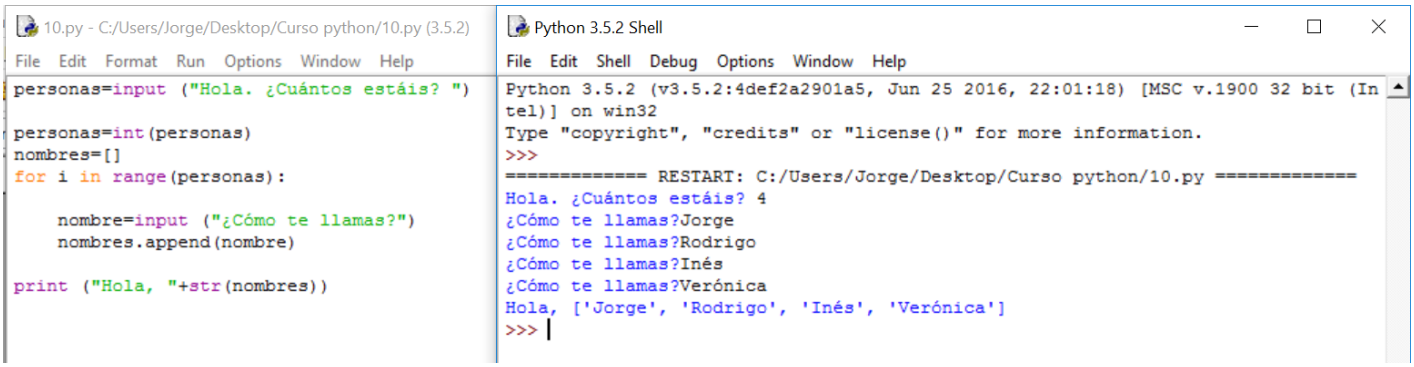
1.- Saludo y pregunto cuánta gente hay

2.- Para cada uno de ellos:

2.1.- Pido el nombre y lo voy almacenando en algún sitio: Una lista.

3.- Saludo a todos poniendo la lista después de "Hola, ".

Solución:



Comentarios:

Como se ve, la transformación a texto de la lista es bastante mala para presentarla en pantalla. A lo largo de los dos ejercicios siguientes vamos a ver cómo solucionarlo.

2. Entrando en materia

Programa 9

Descripción del problema:

Este programa va a solucionar el problema de visualización del programa 8 cambiando la cadena de texto progresivamente. Es decir, una variable de texto va a incluir el saludo y los nombres de cada uno de los presentes. Para hacerlo, aunque hay muchas posibilidades, se va a modificar el programa 8. Esto implica que se van a seguir guardando los nombres en una lista.

Materia nueva:

Aquí hay varias apreciaciones respecto a la estructura **for**.

Si nos paramos a pensar, el conjunto de los números enteros es como una lista. Los días de la semana forman otra lista, y un largo etcétera. Tenemos un elemento inicial y otros siguientes ordenados; eso define una lista.

Pues bien, **for** puede, no sólo recorrer una lista de enteros sino también una de cualquier otra cosa.

Su sintaxis es muy parecida a la que vimos en el programa 7 pero usando la lista dentro de **range**:

```
python
for i in range(lista):
```

Si sólo fuera cuestión de números, podríamos haber simulado una estructura for por medio de un while:

i=0

```
while i<numerofinal:
órdenes
i=i+1
```

Si **for** se ha ganado un puesto en Python es por la potencia que otorga esta posibilidad.

En este programa vamos a hacer referencia a dos situaciones más que van indisolublemente unidas:

- El cambio recurrente en las variables.
- La lectura de programas.

¿Por qué van unidas? Pues muy sencillo, porque para entender bien qué estamos realizando si cambiamos una variable cada vez que se ejecuta un paso en un **for** es necesario tener las ideas muy claras, y eso lo da la lectura de programas; debemos ser capaces de seguir la traza de las diferentes variables a lo largo del programa para saber cómo van a acabar.

¿Qué es un cambio recurrente? Es un cambio que se repite, generalmente dentro de una estructura tipo **while** y **for**.

No es algo del todo nuevo, hemos realizado cambios como veces=veces+1 donde hemos cambiado veces por su siguiente. La cuestión es que ahora lo meteremos en una estructura que va a realizar ese cambio un montón de veces. No hay ningún cambio de sintaxis, esta explicación no va a parar a ninguna orden o estructura nueva, es simplemente para que tomes conciencia del paso tan grande que se va a realizar.

A continuación te presento un método para hacer una lectura de programa. No es la única forma y cada uno puede hacerlo según le parezca, eso sí, el orden es imprescindible.

Se pone: “Situación inicial” y se escriben las variables tal y lo que valen en su inicio.

Cuando hay un cambio, se escribe debajo de la variable el nuevo valor o la salida por pantalla. Así se va realizando hasta el final.

Es necesario poner la cadena de texto entre comillas para señalar que es texto.

Se representa a continuación el del programa 7:

|Situación inicial|||||Pantalla| |--|--|--|--|--|--|--| |Personas (entero)|i|nombre|edad|veces|Hola, ¿Cuántos estáis?| |"3"|||||| |3|||||| ||0||||| ||||||¿Cómo te llamas?| |||"Jorge"|||| ||||||¿Cuántos años tienes?| ||||27||| |||||1|| |||||"1"|| ||||27||| ||||||Te hemos preguntado 1 veces| ||||||Pon orden, Jorge| ||1||||| ||||||¿Cómo te llamas?| |||Rodrigo|||| ||||||¿Cuántos años tienes?| ||||"asd"||| |||||1|| ||||||Pon un número, que no es tan difícil| ||||"16"||| |||||2|| |||||"2"|| ||||16||| ||||||Te hemos preguntado 2 veces| ||||||Estudia, Rodrigo| ||2||||| ||||||¿Cómo te llamas?| |||"Inés"|||| ||||||¿Cuántos años tienes?| ||||"14"||| ||||||Te hemos preguntado 1 veces| ||||||Estudia, Inés| |3|2|"Inés"|14|1|||

Situación final. En el caso de que quisiéramos ver otra situación, nos moveríamos a la línea en cuestión y miraríamos el valor de cada una de ellas. Las celdas en blanco no implican que las variables no tengan valor, su valor es el del último cambio: el primero que esté subiendo por esa

columna desde la fila donde se quiere saber el valor. Simplemente se ha hecho así por limpieza e identificar fácilmente cuál es la variable que está cambiando en cada momento.

Se deja como ejercicio hacerlo para el presente programa.

Solución

Algoritmo:

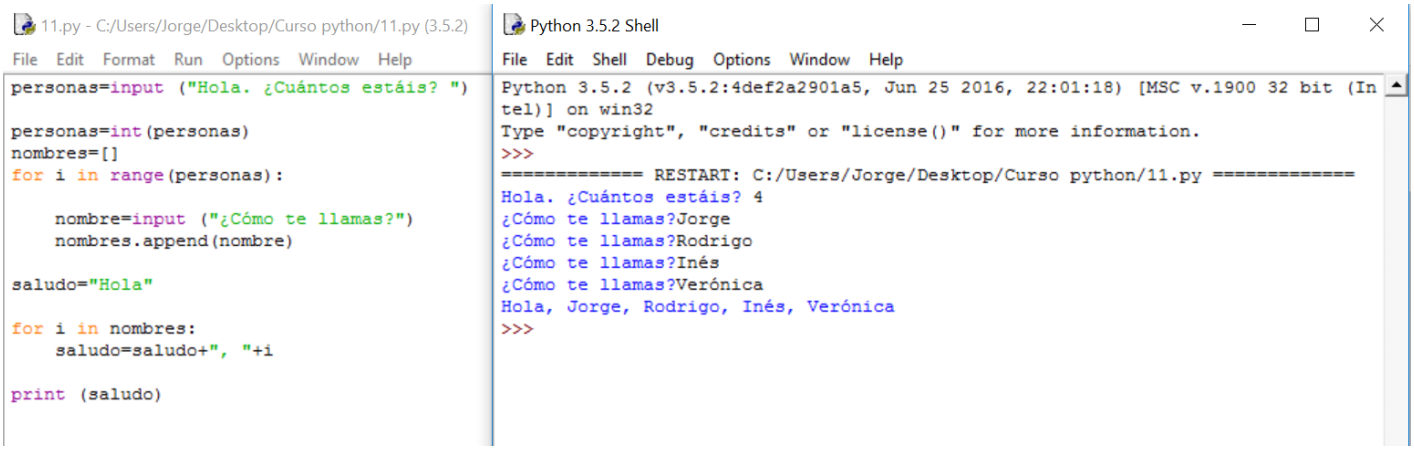
1.- Saludo y pregunto cuánta gente hay

2.- Para cada uno de ellos:

2.1.- Pido el nombre y lo voy almacenando en algún sitio: Una lista.

3.- Modifico la variable donde haya puesto "Hola, " para ir añadiéndole nombres según recorro la lista.

Solución:



Comentarios:

Es necesario hacer la tabla indicada en el apartado de **Materia nueva** para entender qué está pasando con la variable **saludo**. Queda como ejercicio.

Queda como ejercicio conseguir que entre **Inés** y **Verónica** haya una **y.**

2. Entrando en materia

Programa 10

Descripción del problema:

Se va a modificar el problema 8 pero usando una lista y recorriéndola sólo una vez. Es decir, deberemos encontrar una forma de pasar de la lista a una Cadena de caracteres.

Materia nueva:

Una Cadena de caracteres es, sin lugar a dudas, una lista: Tiene un carácter inicial y el resto van ordenados. Es como una lista de caracteres.

Python, debido a lo mucho que se utilizan las Cadenas de caracteres, les ha reservado un tipo de variable especial, como hemos visto a lo largo del curso, y no nos hace trabajar todo el rato con listas para escribir algo, por suerte. Sin embargo, no las ha separado del todo y mantienen varias órdenes en común con las listas. Este programa está hecho sólo para sensibilizar sobre las Cadenas de caracteres y abrir una ventana en algo que se había dado como acotado.

A continuación, se va a presentar una orden referida a las Cadenas de caracteres, necesaria para solucionar el problema que tenemos entre manos, y aprovecharemos para practicar la lectura de la documentación de Python. Tal y como se puede leer

https://docs.python.org/3.1/library/stdtypes.html#string-methods:

str.join (**iterable**)

Return a string which is the concatenation of the strings in the **iterable**. A Type error will be raised if there are any non-string values in **seq**, including bytes objects. The separator between elements is the string providing this method.

Si se ha abierto el enlace, vemos que algunas de las palabras son enlaces que nos llevan a la página donde se explica lo que es. De momento, lo que con este curso se puede leer es poco pero haremos el intento y te pido que lo hagas con **.isdigit()** que también está en ese enlace.

Si traducimos casi literalmente: Devuelve una Cadena que es la concatenación de cadenas presentes en el "iterable" (quiere decir: A container object capable of returning its members one at a time. Un objeto contenedor capaz de devolver sus miembros de uno en uno). Se lanzará un error de tipo (ya vimos uno en los primeros programas antes de introducir **str()**) si no hay valores tipo

Cadena en la secuencia, incluyendo Cadenas de bytes (fuera del alcance de este curso y novedad en Python 3.0). El separador entre elementos es la Cadena sobre la que se le aplica el método (lo primero que se pone antes del punto: **str**).

Lo que, traducido, viene a decir:

separador **.join(**cadenasaañadir**)** Esta orden junta las cadenas que tenga la variable **cadenasaañadir** separándolas con la cadena **separador**.

¿Qué puede ser **cadenasaañadir**? Como se ha visto, cualquier cosa que tenga cadenas, en nuestro caso será una lista pero hay muchas más posibilidades.

De momento, no hay que preocuparse porque la orden la he traducido y no se te va a pedir que uses ninguna otra sin explicarla. De todas formas, era necesario abrir el campo a un aprendizaje posterior, aunque quede sólo para los más osados ya que hay más posibilidades para aprender sin lidiar con semejante redacción, lo normal es que sólo aquellos que tengan un conocimiento avanzado se atrevan.

En los ejercicios de autoevaluación se seguirá trabajando con Cadenas de caracteres. Es un tema muy extenso y es muy necesario trabajarlo.

Solución

Algoritmo:

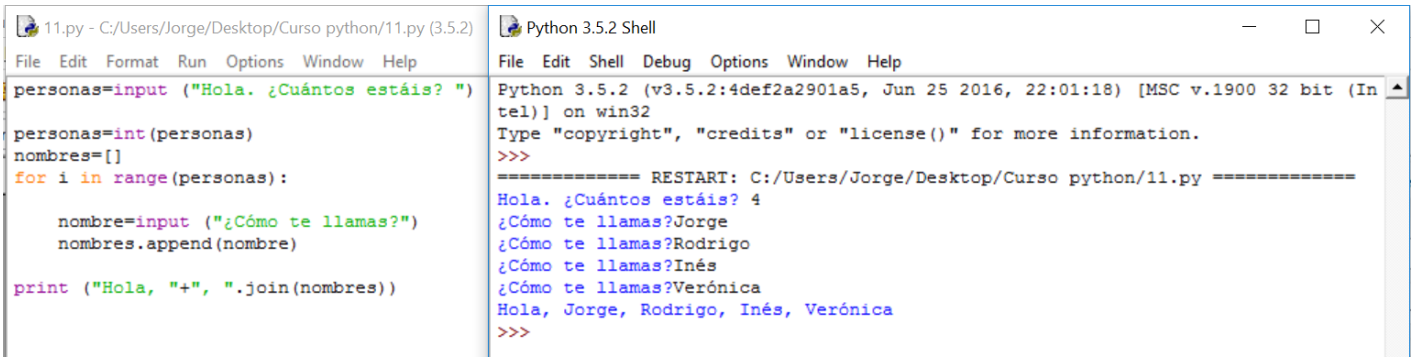
1.- Saludo y pregunto cuánta gente hay

2.- Para cada uno de ellos:

2.1.- Pido el nombre y lo voy almacenando en algún sitio: Una lista.

3.- Convierto la lista en una Cadena de caracteres que poder sumarle al saludo "Hola, ".

Solución:



Copyright 2026

2. Entrando en materia

Ejercicios de autoevaluación

Felicidades por haber finalizado el Módulo 2. A continuación se muestran los programas que servirán de repaso a lo aprendido:

1.- Realiza un programa que "pinte" rectángulos, para ello debe pedir un carácter, la anchura y la altura. El resultado debe ser un rectángulo en la pantalla.

Este programa es complejo. Es necesario diseñar muy bien el algoritmo y pensar muy detenidamente qué es un rectángulo.

Para finalizar cada línea y que salte a la siguiente, debemos introducir un carácter de final de línea. Son caracteres especiales que indican el final de línea, de archivo... El que os hace falta es **"\n"** Ponedlo entrecomillado, al fin y al cabo es un carácter.

2.- Realiza un programa que calcule los divisores de un número que ponga el usuario. Al final, debe preguntar si continúa o no. Si continúa, debe pedir otro número. Debe admitir, para salir, una "n" o una "N".

En este programa, sabemos que basta con dividir por todos los números menores que él hasta su raíz cuadrada pero no nos complicaremos, es mejor dividir por todos.

La parte de la opción de salida "n" o "N" va a alterar la condición de salida. Para incluirle otra condición más en la comparativa de cualquier **if** o **while**:

Si es una "o" lógica: Condición1 **or** Condición2

Si es una "y" lógica: Condición1 **and** Condición2

3.- Realiza un programa en el que el usuario va a introducir un texto y el ordenador va a decirle el número de vocales que tiene.

En este caso, se podrían usar las funciones de **string** de la página que se indicó en el programa 10. Una sería **str.lower(s)**y otra **str.count(** sub, start, end **)**

Se recomienda intentarlo con esas funciones también, pero se pide sin ellas. Si hubiera que buscar conjuntos de letras mayores sí sería necesario.

Copyright 2026

4.- Realiza un programa que guarde los números que el usuario vaya introduciendo; el criterio de finalización es cuando introduzca algo que no sea un número. Una vez que haya terminado, el usuario debe indicar un número que haya introducido y el ordenador deberá decirle el puesto en el que lo metió.

Es necesario realizarlo con una lista. Se podría realizar con una orden de las que aparecen en la página que se indicó en el programa 8: lista**.index(**elemento**)**

Copyright 2026

2. Entrando en materia

Soluciones a los ejercicios de autoevaluación

autoeval2.1.py - C:\Users\Jorge\Desktop\Curso python\autoeval2.1.py (3.5.2)

File Edit Format Run Options Window Help

```
caracter=input ("Elige el caracter del que quieres un
ancho=int(input("Elige la anchura: "))
altura=int (input("Elige la altura: "))
linea=""
rect=""
for i in range(ancho):
    linea=linea+caracter

for i in range(altura):
    rect=rect+linea+"\n"

print (rect)
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:\Users\Jorge\Desktop\Curso python\autoeval2.1.py ========
Elige el caracter del que quieres un rectángulo: *
Elige la anchura: 4
Elige la altura: 7
****
****
****
****
****
****
****

>>>
```

autoeval 2.2.py - C:\Users\Jorge\Desktop\Curso python\autoeval 2.2.py (3.5.2)

File Edit Format Run Options Window Help

```
continua=True
while continua:

    numero=int(input("Introduce un número: "))
    salida=""
    for i in range(1,numero+1):
        if numero%i==0:
            salida=salida+str(i)+"\n"
    print (salida)
    respuesta=input("¿Quieres continuar? (S/N): ")
    if respuesta=="N" or respuesta=="n":
        continua=False
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:\Users\Jorge\Desktop\Curso python\autoeval 2.2.py ========
Introduce un número: 90
1
2
3
5
6
9
10
15
18
30
45
90

¿Quieres continuar? (S/N): s
Introduce un número: 45
1
3
5
9
15
45

¿Quieres continuar? (S/N): n
>>>
```

autoeval 2.3.py - C:/Users/Jorge/Desktop/Curso python/autoeval 2.3.py (3.5.2)

File Edit Format Run Options Window Help

```
vocales=["a","e","i","o","u","A","E","I","O","U","á","é
cuenta=0
texto=input("Introduce un pequeño texto")
for i in vocales:
    for z in texto:
        if i==z:
            cuenta=cuenta+1
print ("El número de vocales es "+str(cuenta))
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:/Users/Jorge/Desktop/Curso python/autoeval 2.3.py ========
Introduce un pequeño textoJueqoejaczéü
El número de vocales es 7
>>>
```

autoeval 2.4.py - C:/Users/Jorge/Desktop/Curso python/autoeval 2.4.py (3.5.2)

File Edit Format Run Options Window Help

```
continua=True
numeros=[]
while continua:
    numero=input("Introduce un número: ")
    if numero.isdigit():
        numeros.append(numero)
    else:
        continua=False
busca=input("Introduce el número del que quieres saber algo: ")
posicion=0
for i in numeros:
    posicion=posicion+1
    if i==busca:
        lugar=posicion
print("Lo metiste en la posicion: "+str(lugar))

"""Esto es un comentario, se debe introducir con
tres comillas al inicio y al final. Van muy bien
para explicar lo que se hace, ya no para el resto
sino para uno mismo. Estas líneas ni se ejecutan
ni salen por pantalla"""

"""Si se hubiera hecho con las funciones propias
de las listas, nos habriamos ahorrado:
posicion=0
for ...

Dejo la orden para que se vea: """

print("Lo metiste en la posición: "+str(numeros.index(busca)+1))
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:/Users/Jorge/Desktop/Curso python/autoeval 2.4.py ========
Introduce un número: 8
Introduce un número: 9
Introduce un número: 4
Introduce un número: 7
Introduce un número: 12
Introduce un número: 45
Introduce un número: 2
Introduce un número: t
Introduce el número del que quieres saber algo: 7
Lo metiste en la posicion: 4
Lo metiste en la posición: 4
>>>
```

Copyright 2026

3. Llegando al final

Copyright 2026

Llegando al final

Objetivos

- Aprender a manejarse con tipos de variable avanzados.
- Aprender a utilizar funciones.

Copyright 2026

3. Llegando al final

Programa 11

Descripción del problema:

Se desea realizar un programa que guarde el nombre, la edad y lo que se les recomendaba hacer, como teníamos antes, para un número de presentes que el usuario dirá. Después de preguntar la información de cada uno de ellos, el programa debe preguntar un número y dar la información referida a la persona que se introdujo en esa posición.

Materia nueva:

Los Diccionarios son un tipo de datos que puede guardar para una sola variable un conjunto de campos. Pongamos un ejemplo: Imaginemos un alumno. Existe un montón de información que podríamos querer guardar sobre él y hacer una tabla como:

|Nombre|Apellido 1|Apellido 2|Altura|Nº Calzado|Color de pelo| |--|--|--|--|--|--| |Jorge|Bes|Tuán|2 m|47|Negro|

Cuando alguien nos preguntara por este alumno, lo haría diciéndonos: Dime el primer apellido del alumno, el color de pelo... Es decir, nos diría el campo que quiere. Tras esto, nosotros iríamos a la celda y responderíamos con la información allí contenida.

Bueno, pues esto es un Diccionario para Python. Tiene unos valores que se llaman Key que en este caso sería la fila superior (Nombre, Apellido1...) y otros que se llaman Value que serían los de la fila inferior.

Una duda que suele asaltar es: ¿Por qué sólo una fila? ¿Y si queremos guardar un conjunto de 1000 alumnos? Pues muy sencillo, se hace una lista de diccionarios, que es lo que vamos a hacer en nuestro programa. Al fin y al cabo, una lista es un conjunto de cosas, nadie ha dicho que haya algo que no se pueda meter ahí.

La sintaxis para declarar un diccionario vacío es:

diccionario **={}**

Y para guardar cualquier dato es:

diccionario[“Key1”]=datoaguardar

Copyright 2026

Aquí hay que decir que, a diferencia de las listas, donde había que añadir un nuevo componente con una orden, aquí es tan fácil como poner una nueva "Key" para que la añada al conjunto de ellas. Es decir, es como tener una tabla dinámica donde podemos añadir columnas sobre la marcha.

Un diccionario tiene muchísimas más posibilidades en Python, como se puede ver en:

https://docs.python.org/3/library/stdtypes.html#typesmapping

No obstante, en este curso no se va sino a introducir este tipo de variable. Queda demostrada la potencia de Python y el nivel de abstracción de los datos y se anima a los alumnos a completar su formación conforme lo vayan necesitando para sus proyectos.

NOTA: Si vas a introducir cambios en un diccionario dentro de una estructura como **for** o **while** , que los ejecutará varias veces, declara esa variable dentro de la estructura y no fuera. Más adelante se explica el por qué.

Solución

Algoritmo:

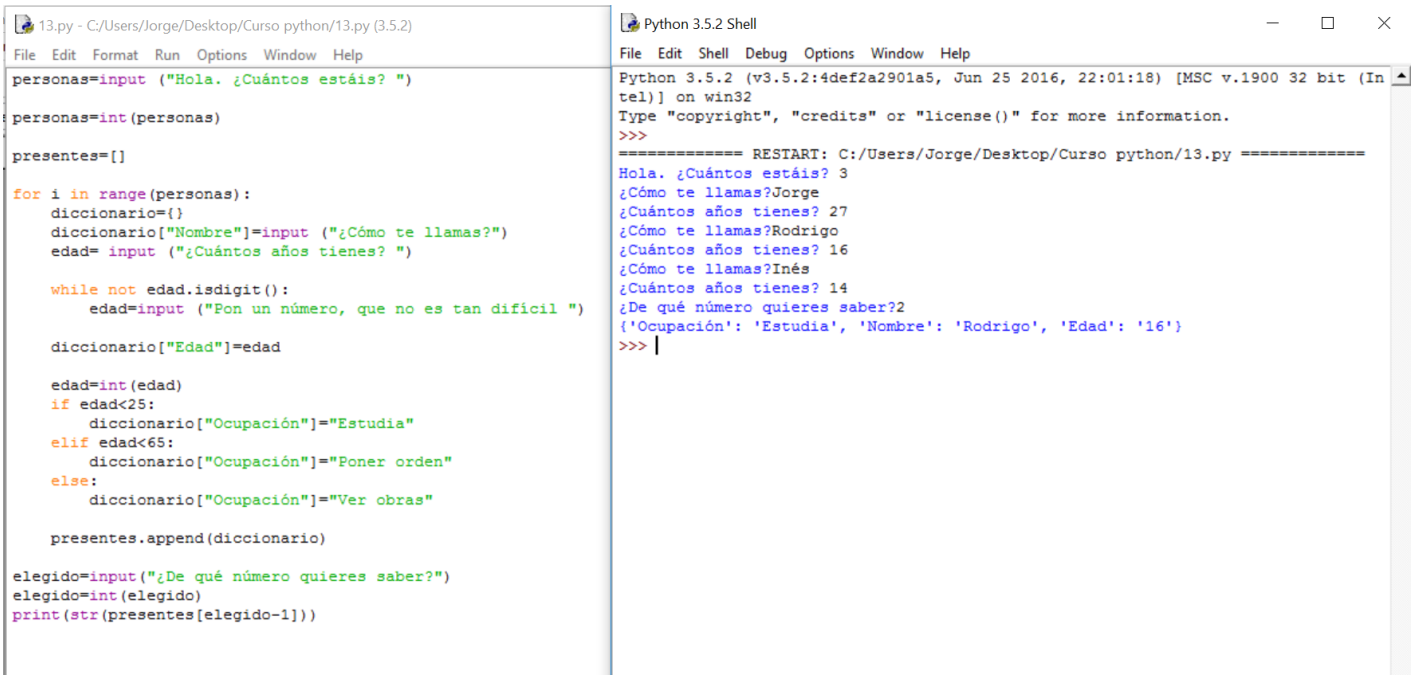
1.- Preguntaremos cuánta gente hay

2.- A cada uno le preguntaremos su nombre y su edad. Guardaremos las dos variables anteriores y la tarea que tienen que hacer (poner orden, estudiar y mirar obras) en un diccionario.

3.- Pediremos que nos diga el número de persona que quiere que le mostremos.

4.- Lo mostraremos en pantalla.

Solución:



Comentarios:

Aquí hay que comentar un error bastante curioso que tiene más que ver con cómo hace las cosas Python que con la programación en sí. Prueba a cambiar la declaración diccionario={} a justo antes del **for** y usa la siguiente orden al final del programa:

print (str(presentes**))** Para que saque por pantalla toda la lista que hemos introducido.

Verás que ha guardado el último tantas veces como gente había. Esto es porque, cada vez que se declara, hace uno nuevo, si no, sólo se está alterando el mismo todo el rato.

No hace falta que lo entiendas, realmente es un error de Python, aunque conozco que se da en varios lenguajes más. Simplemente, quédate con que las variables, para que se puedan usar varias veces, es necesario que tengan su espacio en memoria diferente, y eso lo haces declarándolas de nuevo ya que Python les busca un nuevo hueco en memoria aunque las llame igual.

Es necesario también destacar el recurso de restar 1 a la variable que nos dice qué persona queremos para ajustar el que el ordenador empiece a contar por el 0 y nosotros por el 1.

Copyright 2026

3. Llegando al final

Programa 12

Descripción del problema:

Si miramos el programa número 11, nos damos cuenta de que es todo como un spaghetti, empieza en la primera orden y es todo secuencial hasta el final. Además, a la hora de la lectura, eso de ver el trozo donde se mira la ocupación con tres declaraciones de dicha parte del diccionario no la facilita, como se ve a continuación.

```
edad=int(edad)
if edad<25:
    diccionario["Ocupación"]="Estudia"
elif edad<65:
    diccionario["Ocupación"]="Poner orden"
else:
    diccionario["Ocupación"]="Ver obras"
```

Este tipo de programación se le llama Programación Spaghetti. El programa que hemos hecho es sumamente simple pero, si imaginamos un programa de 2000 líneas, el problema de lectura puede ser tan serio que retocar algo del mismo nos suponga un dolor.

Para evitar esto existen las funciones, que no es otra cosa que sacar código del hilo principal para llamarlo cuando nos convenga. De esta forma, tendremos el **programa** muy claro para leer y retocar.

El programa 12 consiste en extraer la parte de la ocupación del hilo principal.

Materia nueva:

Una función se define en Python de la siguiente manera:

```
def nombredelafuncion (variablesquevaausar)
Ordenes
return(variableadevolver)
```

Copyright 2026

Para recibir el dato que devuelve (**return** significa devolver) es necesario guardarlo en algún sitio. Lo habitual es asignarlo a una variable:

```
variable=nombredelafuncion (variablequeleentregamos)
```

Respecto a la ubicación de las funciones, lo habitual es declararlas antes de que empiece el programa. Por lo tanto, las primeras líneas se dedican a escribir funciones y el hilo principal del programa viene después.

Una vez definida, la llamada es tan simple como poner el nombre de la función y decirle qué variables va a usar.

¡Cuidado! Las variables que va a usar no son aquellas con las que se le llama. Digamos que la función recibe unas variables pero, para no modificar las del programa, hace una copia cambiándoles el nombre y usa las que ha copiado.

NOTA: Si te has dado cuenta, a lo largo de todo el curso he hablado de "órdenes", pues bien, era mentira: son "funciones". Se ha hecho así porque era demasiado duro empezar a hablar de algo que quedaba muy lejos en el progreso de aprendizaje; mejor ver primero para qué sirven y luego ya se explicará el por qué.

Solución

Algoritmo:

1.- Preguntaremos cuánta gente hay

2.- Para cada uno de ellos se les pregunta y:

- 2.1.- En un diccionario iremos guardando nombre y edad.
- 2.2.- A la hora de elegir ocupación haremos una llamada a una función que lo calculará *Problema

3.- Pediremos que nos diga el número de persona que quiere que le mostremos.

4.- Lo mostraremos en pantalla.

Solución:

14.py - C:/Users/Jorge/Desktop/Curso python/14.py (3.5.2)

File Edit Format Run Options Window Help

```
def obtenerocupacion (numero):

    if numero<25:
        curro="Estudia"
    elif numero<65:
        curro="Poner orden"
    else:
        curro="Ver obras"
    return curro

personas=input ("Hola. ¿Cuántos estáis? ")

personas=int(personas)

presentes=[]

for i in range(personas):
    diccionario={}
    diccionario["Nombre"]=input ("¿Cómo te llamas?")
    edad= input ("¿Cuántos años tienes? ")

    while not edad.isdigit():
        edad=input ("Pon un número, que no es tan difícil ")

    diccionario["Edad"]=edad

    edad=int(edad)
    diccionario["Ocupación"]=obtenerocupacion(edad)
    presentes.append(diccionario)

elegido=input("¿De qué número quieres saber?")
elegido=int(elegido)
print(str(presentes[elegido-1]))
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/Jorge/Desktop/Curso python/14.py ==========
Hola. ¿Cuántos estáis? 3
¿Cómo te llamas?Jorge
¿Cuántos años tienes? 27
¿Cómo te llamas?Rodrigo
¿Cuántos años tienes? 16
¿Cómo te llamas?Inés
¿Cuántos años tienes? 14
¿De qué número quieres saber?3
{'Edad': '14', 'Nombre': 'Inés', 'Ocupación': 'Estudia'}
>>>
```

Copyright 2026

3. Llegando al final

Programa 13

Descripción del problema:

Se desea realizar un programa que devuelva el máximo común divisor de dos números. Se debe realizar por medio de funciones.

Materia nueva:

Una función puede devolver tan sólo un único valor. La solución es devolver una lista que, a fin y al cabo, es una única variable.

En este programa, devolveremos los divisores de cada número en una lista.

Solución

Algoritmo:

1.- Pedimos los dos números

2.- Calculamos sus divisores

3.- El mcd se obtiene comparando las dos listas. El mayor divisor que esté en las dos listas será el máximo común divisor. La forma más rápida es:

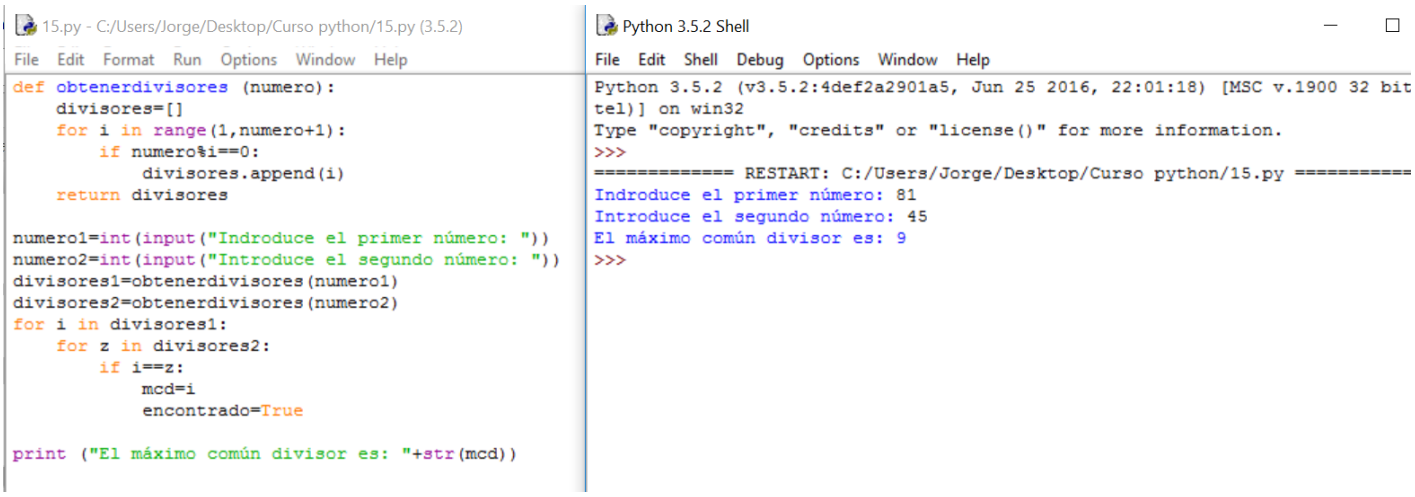
3.1.- Comparar el más pequeño del primer número con todos los del otro número. Lo guardo si coincide.

3.2.- Comparo el siguiente del primer número con todos los del otro número. Lo guardo si coincide.

3.3.- Cuando haya terminado, el último número que tenga guardado será el mcd.

Solución:

Copyright 2026



Comentarios:

El mayor problema de este programa es realizar un algoritmo que calcule el mínimo común divisor. Si las dificultades vienen de ahí, no pasa nada, se irá alcanzando la suficiencia algorítmica con el tiempo y la práctica.

Copyright 2026

3. Llegando al final

Programa 14

Descripción del problema:

Se desea realizar un programa en el que el usuario vaya introduciendo números hasta que introduzca algo que no lo sea. Una vez introducidos, se añadirá a cada número la información de si es primo o no, el número de cifras que tiene y si es el mínimo o el máximo. Al final, sacará todo por pantalla. Es necesario usar un diccionario, listas y funciones.

Materia nueva:

En este caso, nos falta por saber cómo conocer la longitud de una cadena (es una de las múltiples funciones que nos quedan en el tintero). Dicha función es:

len(string**)** Aquí hay una curiosidad. Si la mayor parte eran string**.función**, ¿por qué no hay ningún punto? Pues sencillamente porque esta función se puede usar para un montón de cosas aparte de una Cadena de caracteres. De momento la usaremos para conocer cuántos dígitos tiene el número que el usuario ha introducido.

NOTA:

- **Modo avanzado:** Recuerda que, para cada variable nueva tipo diccionario, es necesario declararla para que no apunte a la misma dirección de memoria.
- **Modo terrestre:** Mete la declaración en el bucle para que se declare y sea nueva cada vez que se usa.

Solución

Algoritmo:

1.- Pedir números hasta que haya una introducción que no se pueda convertir en uno. Ir guardándolos en una lista

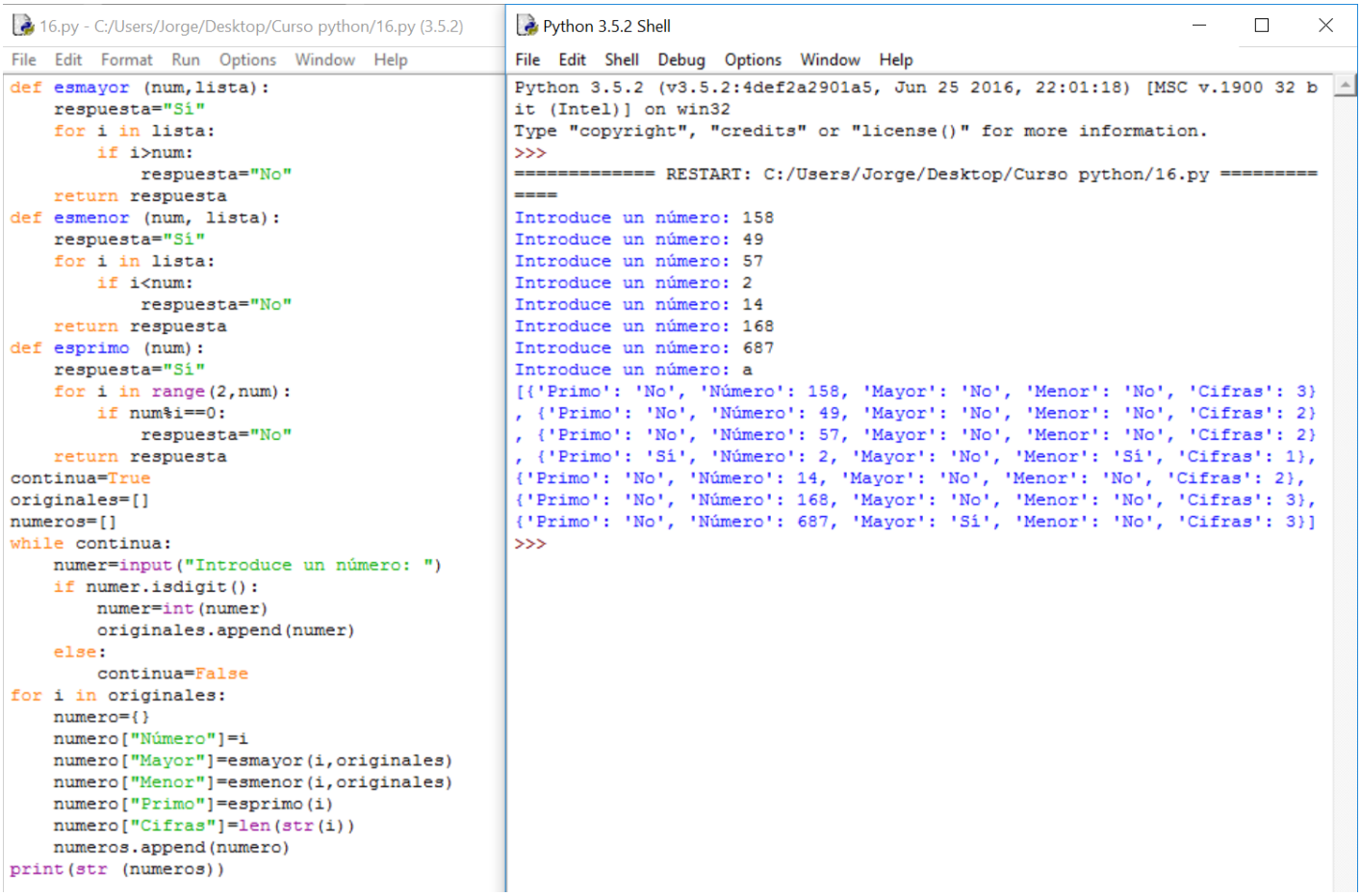
2.- Para cada uno de ellos:

Copyright 2026

- 2.1.- Guardarlo bajo la "Key": "Número"
- 2.2.- Comprobar si es menor, mayor y primo y guardarlo bajo las "Keys": "Menor", "Mayor" y "Primo"
- 2.3.- Contar con la función **len(** s **)** la cantidad de caracteres del número pasado a Cadena de caracteres y guardarlo bajo la "Key": "Cifras"

3.- Sacar la lista de diccionarios por la pantalla.

Solución:



Explicación:

En este caso, en las órdenes:

numero["Mayor"]=esmayor(i,originales) y esmenor(i,originales) podemos ver que se ha trasladado a la función una lista entera.

Podría parecer que sólo admitiría variables simples pero, realmente, no existe límite de datos o variables a la hora de pasar a las funciones.

Sin embargo, cada función usa una lista diferente. Sí, en Python no se permite usar el original de ninguna variable enviada a una función. Si nos pusiéramos puristas, esto duplicaría la cantidad de memoria reservada en el ordenador a nuestro programa; es verdad, pero la cantidad de memoria y la capacidad de cálculo han dejado de ser un problema hace mucho tiempo.

Respecto a la función **len(**s**)**, ha habido que pasarlo primero a Cadena de caracteres para poder realizarlo.

Copyright 2026

3. Llegando al final

Ejercicios de autoevaluación

Felicidades por haber finalizado el Módulo 3. A continuación se muestran los programas que servirán de repaso a lo aprendido:

1.- Realiza un programa que calcule el mínimo común múltiplo de dos números.

Como ayuda: Habrá que realizar una lista con los múltiplos de cada número hasta el producto de los dos que el usuario no dé, ya que será el máximo valor del mcm.

Por lo demás, el proceso es igual que en el mcd pero teniendo en cuenta que, si empezamos buscando desde el valor más bajo, una vez que encuentra la primera coincidencia en las dos listas, ya no querremos que vuelva a cambiar ese valor. Hay una orden que se llama **break** y permite salir del bucle **for** o **while** donde está metido. En este caso no nos ayudará porque estamos metidos en dos. Es necesario usar una booleana que registre si ya ha sido localizado o no y, por tanto, permitir o no, el cambio ante más coincidencias.

Si se hace la lista de multiplicadores empezando por el más alto, nos ahorramos ese pequeño tinglado.

2.- Realiza un programa que, dada una fracción, obtenga la fracción irreducible (la más simplificada posible). El usuario introducirá el numerador y denominador y la salida será un diccionario con "Numerador" y "Denominador" como Keys.****

3.- Realiza un programa que, dadas dos fracciones, obtenga las fracciones equivalentes que tengan el mismo denominador siendo éste el mínimo posible.

4.- Realiza una calculadora de fracciones. Debe poder sumar, restar, multiplicar y dividir dos fracciones. El resultado debe ser una fracción irreducible.

En este caso, como el resultado puede ser negativo, debemos usar una función matemática que nos dé el valor absoluto de un número ya que, si usamos los bucles que hemos realizado hasta ahora, tendremos problemas. Dicha función es **abs(**número**)**

En la solución se han cambiado ciertas funciones que había en el resto de programas para más eficiencia. No es preocupante si os sale con más líneas.

Copyright 2026

NOTA: Estos 4 programas simbolizan la máxima de la programación: Dividir en problemas más pequeños. Aplícalo siempre en tus programas porque, aunque en este curso no ha habido grandes complicaciones, según se quieran resolver problemas más complejos, no es que sea una máxima sino el único camino.

Mi más cordial enhorabuena por haber terminado el curso. Espero que te sea útil y haya sido el primer paso hacia la maestría en programación.

Copyright 2026

3. Llegando al final

Soluciones a los ejercicios de autoevaluación

Autoevaluacion 3.1.py - C:/Users/Jorge/Desktop/Curso python/Autoevaluaci

File Edit Format Run Options Window Help

```
def obtenermultiplos (numero,elotro):
    multiplos=[]
    for i in range(elotro,0,-1):
            multiplos.append(i*numero)
    return multiplos

numero1=int(input("Indroduce el primer número: "))
numero2=int(input("Introduce el segundo número: "))
multiplos1=obtenermultiplos(numero1,numero2)
multiplos2=obtenermultiplos(numero2,numero1)
for i in multiplos1:
    for z in multiplos2:
        if i==z:
            mcm=i
            encontrado=True

print ("El mínimo común múltiplo es: "+str(mcm))
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Jorge/Desktop/Curso python/Autoevaluacion 3.1.py =====
Indroduce el primer número: 12
Introduce el segundo número: 18
El mínimo común múltiplo es: 36
>>>
```

Autoevaluacion 3.2.py - C:\Users\Jorge\Desktop\Curso python\Autoeval

File Edit Format Run Options Window Help

```
def obtenerdivisores (numero):
    divisores=[]
    for i in range(1,numero+1):
        if numero%i==0:
            divisores.append(i)
    return divisores

numero1=int(input("Indroduce el numerador: "))
numero2=int(input("Introduce el denominador: "))
divisores1=obtenerdivisores(numero1)
divisores2=obtenerdivisores(numero2)
for i in divisores1:
    for z in divisores2:
        if i==z:
            mcd=i
            encontrado=True

fraccion={}
fraccion["Numerador"]=numero1/mcd
fraccion["Denominador"]=numero2/mcd
print (str (fraccion))
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Jorge\Desktop\Curso python\Autoevaluacion 3.2.py =====
Indroduce el numerador: 95
Introduce el denominador: 245
{'Numerador': 19.0, 'Denominador': 49.0}
>>>
```

Autoevaluacion 3.3.py - C:\Users\Jorge\Desktop\Curso python\Autoevaluacion 3.3.py (3.5.2)

File Edit Format Run Options Window Help

```
def obtenerdivisores (numero1,numero2):
    divisores=[]
    for i in range(1,numero1+1):
        if numero1%i==0 and numero2%i==0:
            divisores.append(i)
    return divisores

fraccion1={}
fraccion2={}
fraccion1["Numerador"]=int(input("Indroduce el numerador1: "))
fraccion1["Denominador"]=int(input("Introduce el denominador1: "))
fraccion2["Numerador"]=int(input("Indroduce el numerador2: "))
fraccion2["Denominador"]=int(input("Introduce el denominador2: "))
""" Se va a conseguir denominador común multiplicando el denominador
de una fracción por la otra y viceversa"""

fraccion1["Numerador"]=fraccion1["Numerador"]*fraccion2["Denominador"]
fraccion2["Numerador"]=fraccion2["Numerador"]*fraccion1["Denominador"]
fraccion1["Denominador"]=fraccion1["Denominador"]*fraccion2["Denominador"]
fraccion2["Denominador"]=fraccion1["Denominador"]
""" Ahora dividiré por el mayor divisor común a ambas fracciones"""
print (str (fraccion1)+ "\n"+str(fraccion2))
divisores1=obtenerdivisores(fraccion1["Numerador"],fraccion1["Denominador"])
divisores2=obtenerdivisores(fraccion2["Numerador"],fraccion2["Denominador"])

for i in divisores1:
    for z in divisores2:
        if i==z:
            mcd=i

fraccion1["Numerador"]=fraccion1["Numerador"]/mcd
fraccion1["Denominador"]=fraccion1["Denominador"]/mcd
fraccion2["Numerador"]=fraccion2["Numerador"]/mcd
fraccion2["Denominador"]=fraccion1["Denominador"]
print (str (fraccion1)+ "\n"+str(fraccion2))
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 I
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Jorge\Desktop\Curso python\Autoevaluacion 3.3.py
Indroduce el numerador1: 12
Introduce el denominador1: 18
Indroduce el numerador2: 72
Introduce el denominador2: 24
{'Numerador': 288, 'Denominador': 432}
{'Numerador': 1296, 'Denominador': 432}
{'Numerador': 2.0, 'Denominador': 3.0}
{'Numerador': 9.0, 'Denominador': 3.0}
>>>
```

En este caso, hay un paso intermedio que se muestra. Conforme los programas se complican, es necesario decirle que saque por pantalla ciertos valores en determinados momentos para controlar que se está haciendo lo que queremos. Imagina si hubiera 2000 líneas.

Los valores correctos son los últimos.

Autoevaluacion 3.4.py - C:/Users/Jorge/Desktop/Curso python/Autoevaluacion 3.4.py (3.5.2)

File Edit Format Run Options Window Help

```
def mcd (numero1,numero2):
    for i in range(1,numero1+1):
        if numero1%i==0 and numero2%i==0:
            maxcomdiv=i
    return maxcomdiv
def suma(fracc1,fracc2,opera):
    resul={}
    if opera=="+":
        resul["Numerador"]=fracc1["Numerador"]*fracc2["Denominador"]+fracc2["Numerador"]*fracc1["Denominador"]
    else:
        resul["Numerador"]=fracc1["Numerador"]*fracc2["Denominador"]-fracc2["Numerador"]*fracc1["Denominador"]
    resul["Denominador"]=fracc1["Denominador"]*fracc2["Denominador"]
    return resul

fraccion1={}
fraccion2={}
resultado={}
fraccion1["Numerador"]=int(input("Indroduce el numerador1: "))
fraccion1["Denominador"]=int(input("Introduce el denominador1: "))
operacion=input("Introduce la operación que deseas ")
fraccion2["Numerador"]=int(input("Indroduce el numerador2: "))
fraccion2["Denominador"]=int(input("Introduce el denominador2: "))
if operacion=="+" or operacion=="-":
    resultado=suma(fraccion1,fraccion2,operacion)
elif operacion=="*":
    resultado["Numerador"]=fraccion1["Numerador"]*fraccion2["Numerador"]
    resultado["Denominador"]=fraccion2["Denominador"]*fraccion1["Denominador"]
else:
    resultado["Numerador"]=fraccion1["Numerador"]*fraccion2["Denominador"]
    resultado["Denominador"]=fraccion1["Denominador"]*fraccion2["Numerador"]

mcdiv=mcd(abs(resultado["Numerador"]),abs(resultado["Denominador"]))
resultado["Numerador"]=resultado["Numerador"]/mcdiv
resultado["Denominador"]=resultado["Denominador"]/mcdiv

print (str (resultado))
```

Ln: 31 Col: 4

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 20
1)] on win32
Type "copyright", "credits" or "license()" f
>>>
===== RESTART: C:/Users/Jorge/Desktop/Curso
Indroduce el numerador1: 2
Introduce el denominador1: 3
Introduce la operación que deseas +
Indroduce el numerador2: 4
Introduce el denominador2: 5
{'Denominador': 15.0, 'Numerador': 22.0}
>>>
===== RESTART: C:/Users/Jorge/Desktop/Curso
Indroduce el numerador1: 2
Introduce el denominador1: 7
Introduce la operación que deseas -
Indroduce el numerador2: 14
Introduce el denominador2: 21
{'Denominador': 21.0, 'Numerador': -8.0}
>>>
===== RESTART: C:/Users/Jorge/Desktop/Curso
Indroduce el numerador1: 3
Introduce el denominador1: 5
Introduce la operación que deseas *
Indroduce el numerador2: 4
Introduce el denominador2: 10
{'Denominador': 25.0, 'Numerador': 6.0}
>>>
===== RESTART: C:/Users/Jorge/Desktop/Curso
Indroduce el numerador1: 2
Introduce el denominador1: 3
Introduce la operación que deseas /
Indroduce el numerador2: 1
Introduce el denominador2: 8
{'Numerador': 16.0, 'Denominador': 3.0}
>>>
```

En este caso se han optimizado ciertos algoritmos: Se ha utilizado la función **suma** para sumar y

restar y se ha cambiado la función para obtener el máximo común divisor. Esto es lo que suele pasar cuando se le dan varias vueltas al mismo programa. La solución óptima no es exigible a nuestro nivel pero irás viendo que cada vez eres capaz de obtener soluciones más eficientes.

Gracias por haber seguido el curso, te deseo muchos éxitos.

Copyright 2026

Créditos

Autor: Jorge Pérez Ariza

Cualquier observación o detección de error en soporte.catedu.es

Los contenidos se distribuyen bajo licencia **Creative Commons** tipo **BY-NC-SA** excepto en los párrafos que se indique lo contrario.