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:

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.

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 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

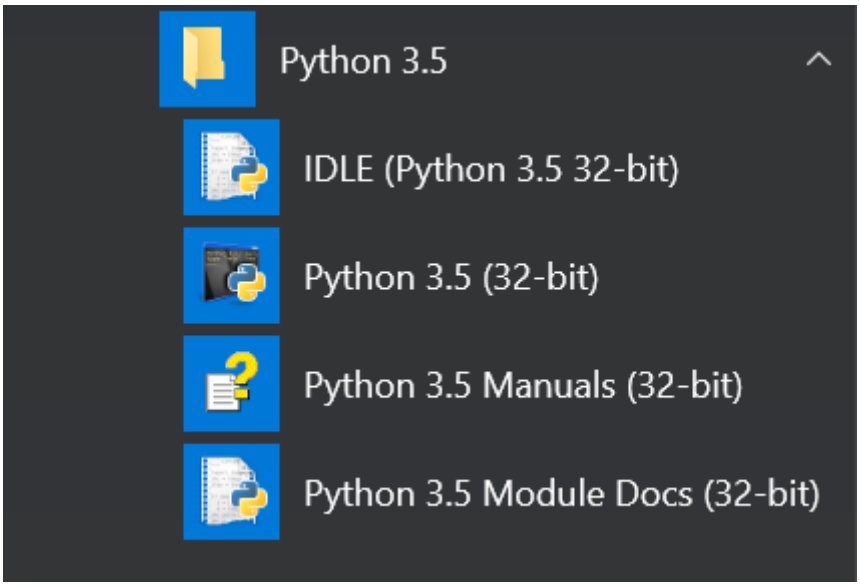
Instalación: Es necesario descargarse el programa de la página web:

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.

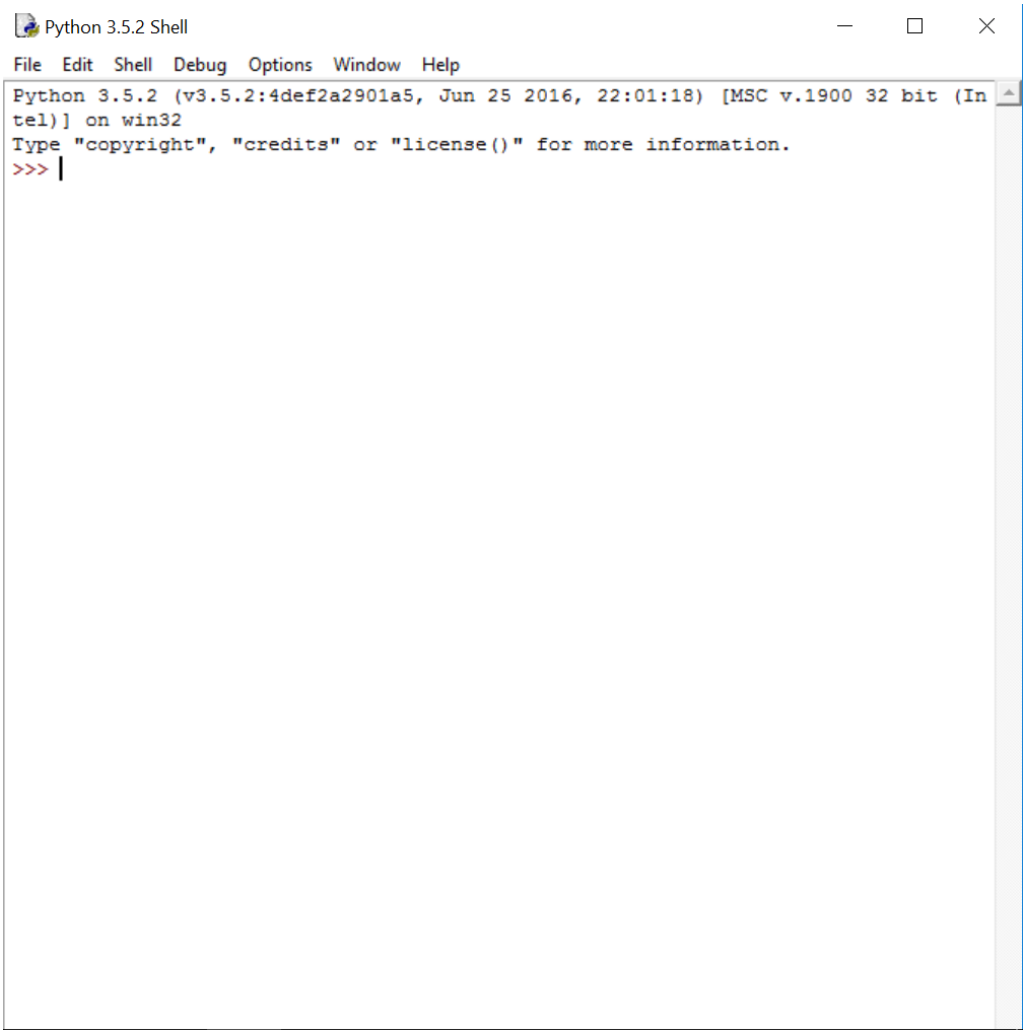
Salimos de la explicación avanzada: Los alumnos rara vez conocen estos problemas y, por tanto, no prestan atención a esto. Convendría dejarles unas pocas ideas claras. Éstas serían:

- 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

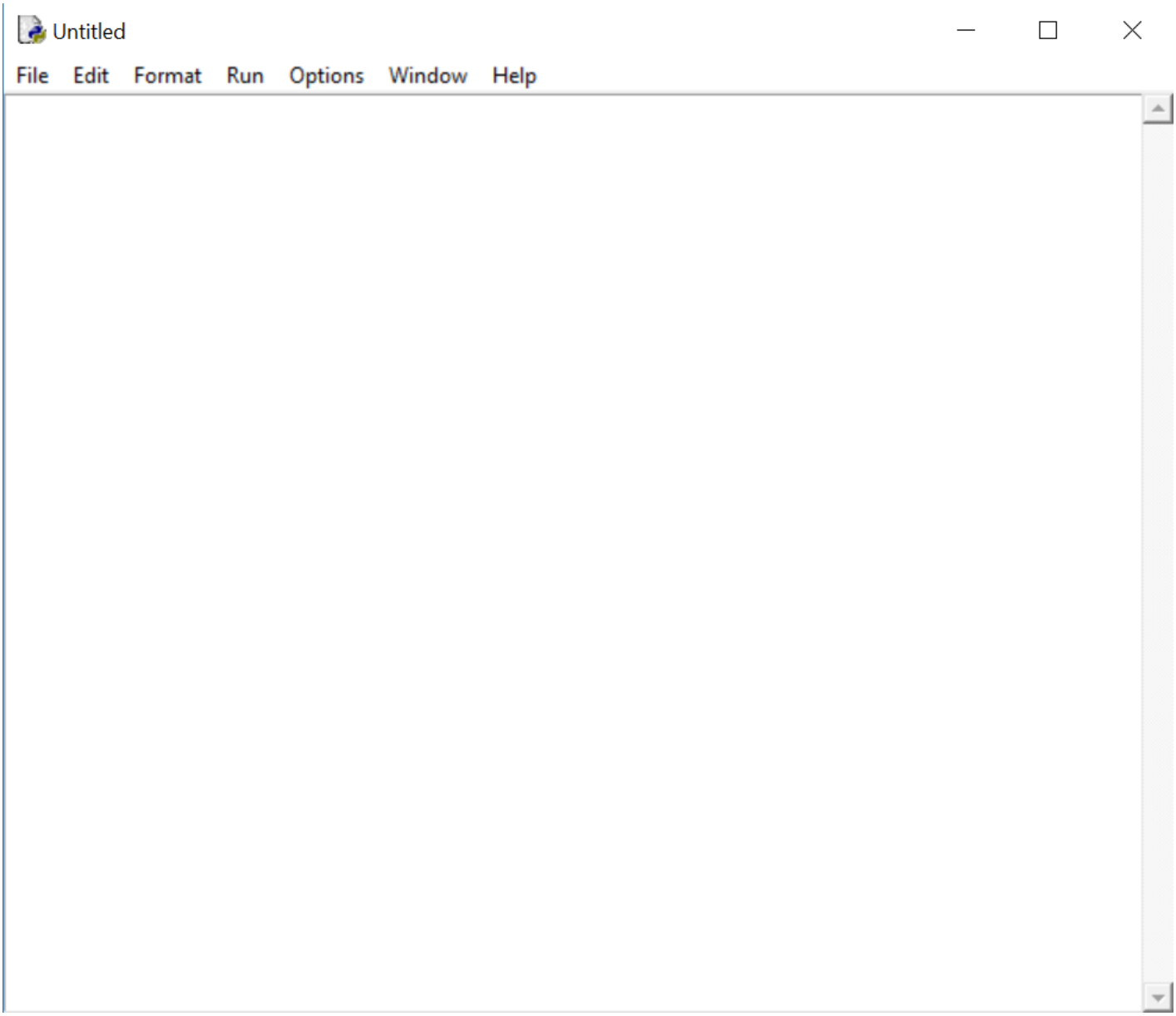
Presentación del sistema de desarrollo:

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.

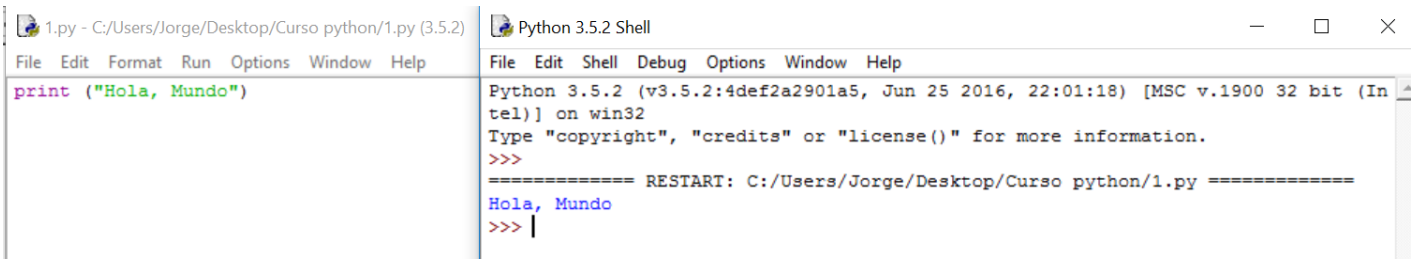
Para ejecutar los programas, es necesario, una vez escritos, ir a **Run** y, una vez allí, elegir **Run Module.**

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

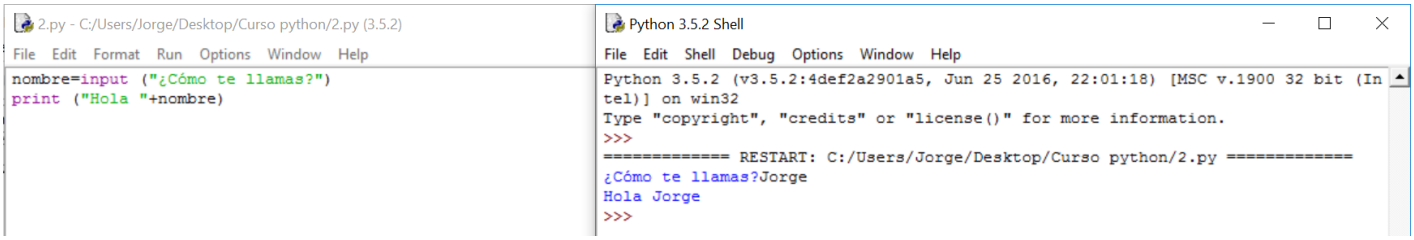
Algoritmo:

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
```

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

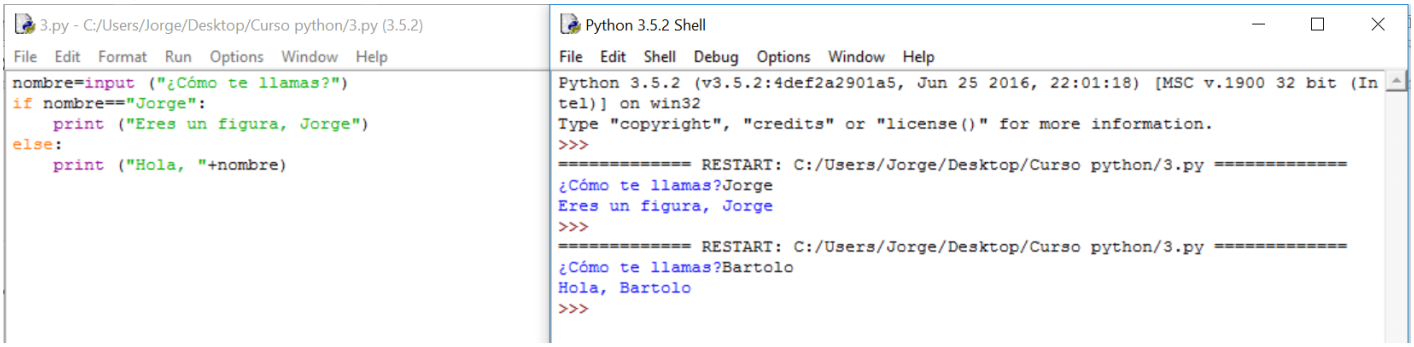
Algoritmo:

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**.

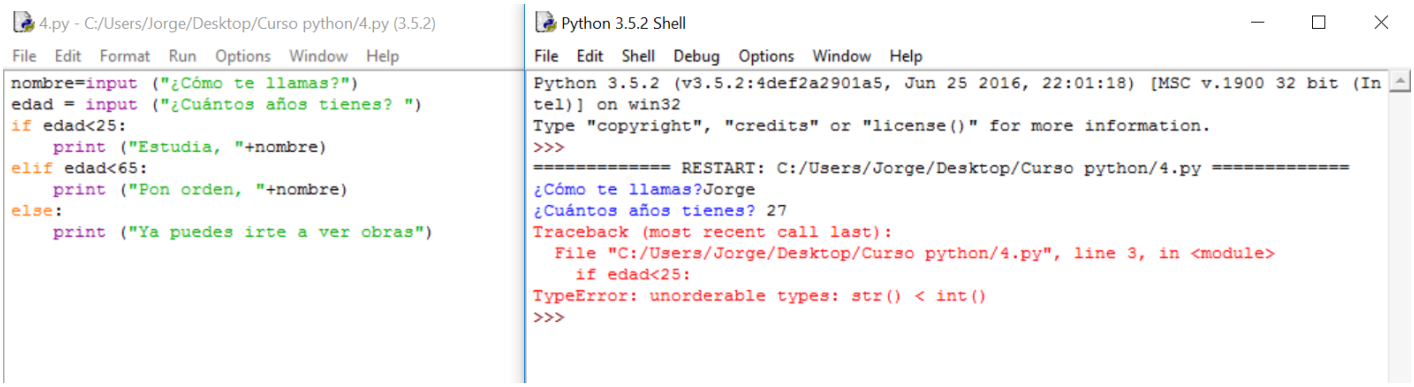
¡Atención! Un problema que se le puede haber ocurrido a alguien es que 24 es menor que 25 y menor que 65. ¿En qué se fijaría en ese momento? La respuesta es sencilla, si está en un if, irá por orden y, si entra en el primero, automáticamente sale de la estructura sin entrar en el siguiente.

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.

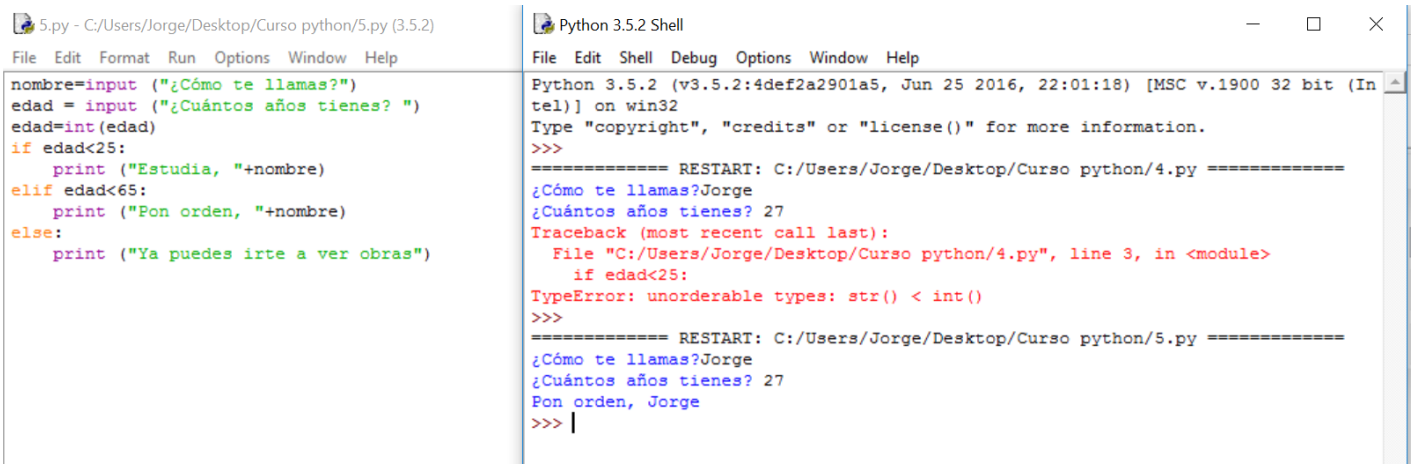
Es muy importante conocer la diferencia entre 2 y “2”. El primero es un número y el segundo es un texto. Algo así como 2 y "dos" para nosotros. Si queremos sumar, no podemos usar la palabra “dos”, debemos usar el número.

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:

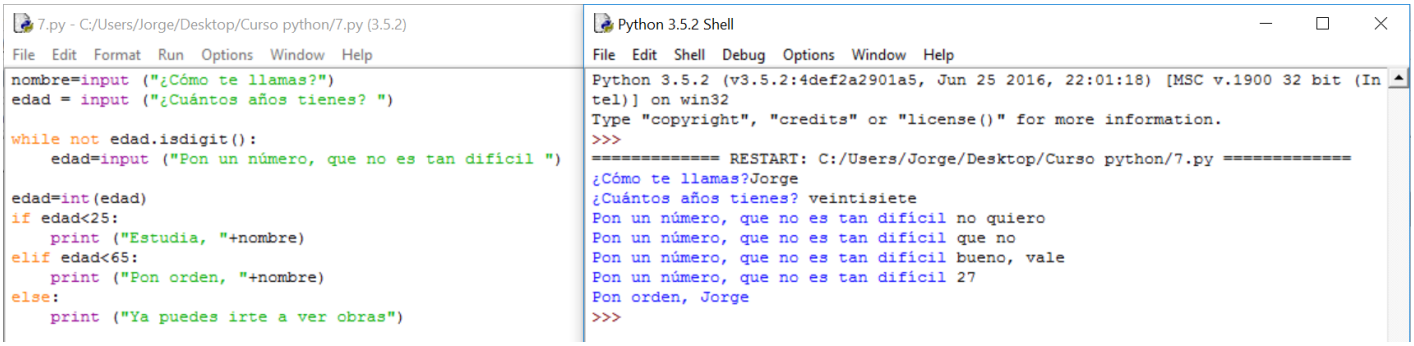
1.- Pido el nombre

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.

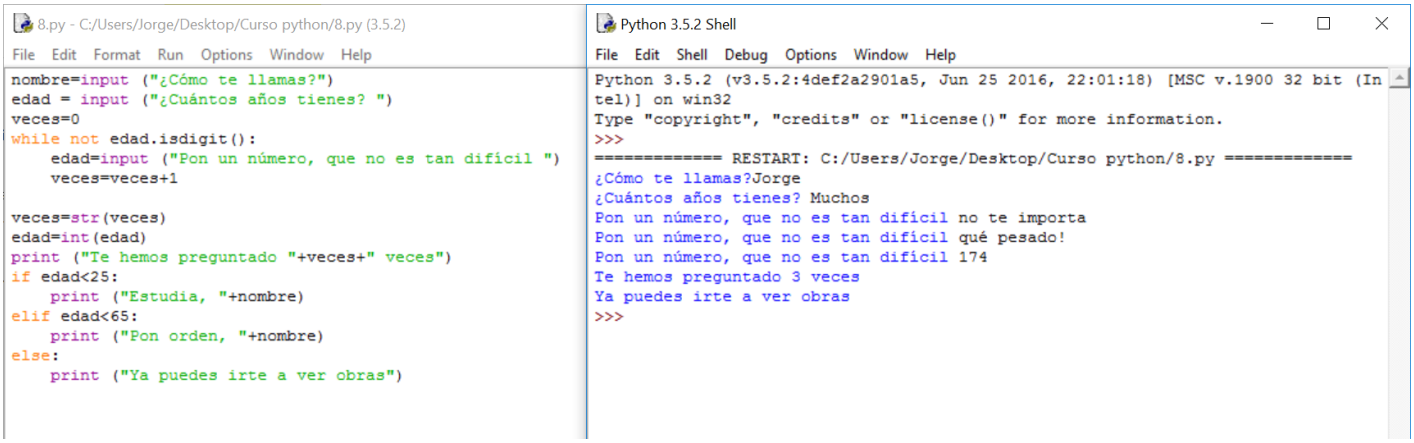
Los **contadores** son variables que almacenan un número que lleva la cuenta de algún proceso... Hay estructuras que veremos un poco más adelante que no pueden existir sin ellos. De momento, quedaos con que es una función tan común para una variable que se le denota de una forma propia.

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.

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
>>>
```

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
>>>
```

Revision #21
Created 1 February 2022 11:41:42 by Equipo CATEDU
Updated 5 July 2023 12:37:59 by Jesús López de Leyva