Manual Básico para Encantadores de Serpientes

¿Qué flauta necesito?

Para encantar a nuestra particular serpiente tenemos dos herramientas. La principal es el Terminal, que podemos encontrar en cualquier distribución de GNU/Linux, en nuestra lista de programas, con un icono con forma de pantalla en color negro. Usaremos el terminal para hacer que nuestra serpiente se mueva mediante nuestras ordenes.

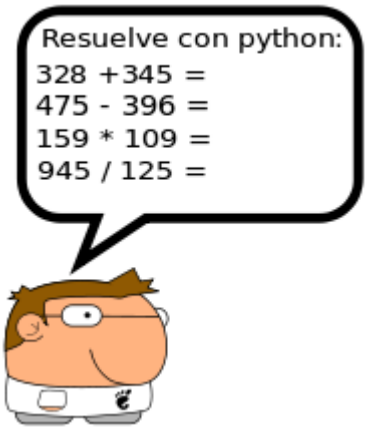
La segunda herramienta, es el editor de texto. En GNU/Linux tenemos varias alternativas: gedit, kate o Geany. Puedes usar el que más te guste. En este editor de texto escribiremos los programas y los guardaremos en ficheros con nombres acabados en .py, por ejemplo flauta.py o mi_programa.py

El editor de texto nos da la ventaja de guardar nuestro trabajo: todas las órdenes que le demos a nuestra serpiente quedaran a salvo y podrás volver a usarlas sin necesidad de escribirlas de nuevo.

Para lanzar el programa que contiene el fichero de texto tendrás que escribir la orden **python flauta.py** en el terminal.

Abrir un terminal, escribir la palabra Python y pulsar la tecla intro. Cuando aparezca el símbolo >>> escribe la orden ***print "Hola Python!"*** *. Después tienes que conseguir el mismo resultado usando la orden en un fichero de texto.*

Hola ! Me llamo Pythonillo



He de presentaros a mi ayudante, a lo largo de este manual te propondrá ejercicios y te ayudará con las órdenes de python.

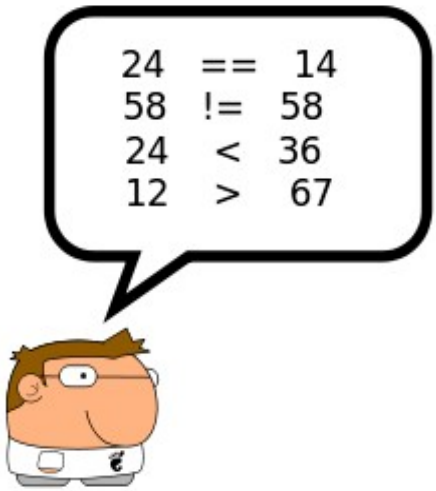
Vamos a seguir conociendo órdenes para nuestra serpiente, comprobaremos que nuestra serpiente es capaz de resolver operaciones matemáticas.

Después de hacer este ejercicio Pythonillo necesita saber cuántos minutos tiene un día. Ten en cuenta que una hora tiene 60 minutos y que un día tiene 24 horas.
Escribe en el Terminal una orden que resuelva este problema.

Cuando quieras realizar varias operaciones (suma,multiplicación...) te será de gran ayuda el uso de paréntesis. Es necesario que lo utilices para establecer qué operación se ejecutará en primer lugar. Por ejemplo, comprueba el resultado de la siguiente operación, con y sin paréntesis.

2 + 2 * 5 (2 + 2) * 5

Blanca y Corchea ¿Son iguales?



Ahora aprenderás los símbolos que utiliza Python para comparar números, palabras o expresiones. Por ejemplo, si queremos saber cual de dos números es mayor, escribiremos en el intérprete de python la siguiente orden:

```
>>> 25 == 46
```

Pythonillo todavía no domina muy bien el español y nos responderá en Inglés. En el caso de que 25 sea igual que 46 responderá **TRUE** y si 25 es diferente a 46 responderá **FALSE**.

Observa los símbolos que ha utilizado Pythonillo. Prueba varios ejemplos con cada símbolo. Recuerda que puedes utilizar tanto números como letras o palabras completas. Al igual que con las operaciones matemáticas, puedes usar paréntesis para ayudarte cuando utilices varios símbolos en una expresión.

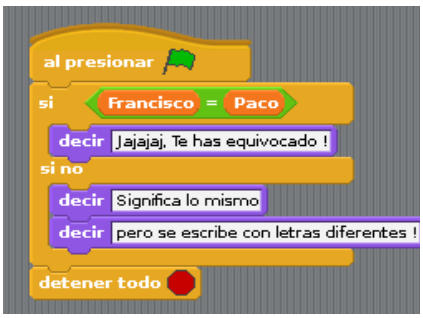
No son iguales... ¿Qué puedo hacer?

Pythonillo tiene un amigo que te va ayudar a responder a esta pregunta. ¿ Conoces un gatito llamado Scratch? Este gatito te va a dar una pista para entender mejor las nuevas órdenes de Python. En los ejemplos anteriores, hemos descubierto que podemos hacerle preguntas a Python y nos dice si es verdadera o falsa.

En este apartado aprenderás a ordenarle a Python lo que quieres que haga, dependiendo de la respuesta que dé a tu pregunta.

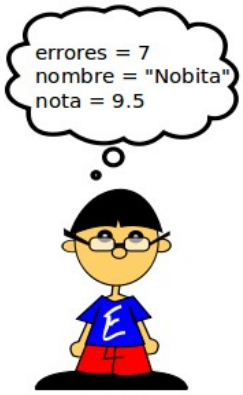
Por ejemplo, vamos a preguntarle: ¿Francisco es igual que Paco? Si responde True, mostraremos la siguiente frase: “Jajajaja, te has equivocado.”. Si responde False: “Es el mismo nombre, pero se escribe con las letras son diferentes.”

A continuación te mostraré como hacer el ejercicio, tanto en Scratch como en Python:




```
if "Francisco" == "Paco":
    print "Jajaja, Te has equivocado!"
else:
    print "Significa lo mismo, pero "
    print "se escribe con diferentes letras"
```

¿Dónde puedo guardar mis notas musicales?



Habitualmente cuando escribimos programas, necesitamos guardar el resultado de expresiones con operadores, como los que hemos visto en ejemplos anteriores. Para esto necesitamos lo que en programación llamamos **variables**. Tendremos que ayudarnos de un nuevo símbolo: " = ", que te servirá para asignar un valor a una variable.

A partir de ahora podrás usar variables y operadores para crear expresiones, de esta forma escribirás y entenderás tu código con más facilidad.

Un uso muy frecuente en programación para las variables es usarlas como un contador. Podemos sumar o restar una cantidad determinada a la variable. Si haces un juego con 10 adivinanzas y quieres que cuente los errores y puntos de cada jugador, para saber quién ha ganado, puedes llevar la cuenta según este ejemplo:

Si el jugador falla una adivinanza: ***errores = errores + 1***

Si el jugador acierta una adivinanza: ***puntos += 5***

total = puntos – errores

Elige un nombre para una variable y guarda tus años. Usa la orden if, en el caso de que tu edad sea mayor que 18 años, Python escribirá "Eres mayor de edad", sino escribirá "Aún no eres mayor de edad". Escribe un mensaje que imprima en pantalla cuántos años te faltan para ser mayor de edad.

! Quiero tocar 5 Corcheas Seguidas !

Nuestro siguiente objetivo es aprender como repetir órdenes de Python, el número de veces que necesitemos para nuestro programa. Existen varias formas de hacerlo, Pythonillo de momento nos ha aconsejado usar la orden "**While expresión :"** significa repetir mientras la expresión sea cierta (True). Veamos un ejemplo.

```
print "Introduce el numero de jugadores:"
jugadores = int (input())

while jugadores >= 4:
      #Mi juego no permite mas de 4 jugadores
      print "Máximo 4 jugadores (1-4) !"
      jugadores = int(input())
```

En este ejemplo, aparece por primera vez la almohadilla (**#**). Este signo se usa cuando el desarrollador quiere hacer un comentario para aclarar el significado de lo que ha escrito en su programa. El comentario que puedes ver en el ejemplo, indica que con esas líneas de programa, se comprueba el

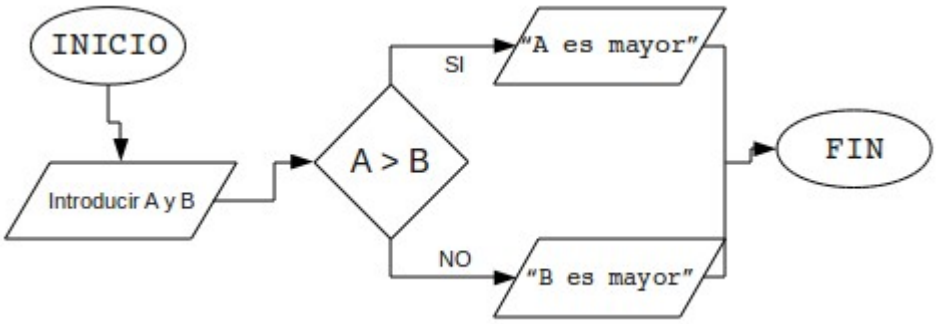
número de participantes en el videojuego y además no pueden ser más de cuatro.
Si te has fijado bien, habrás comprobado que hay un orden que Pythonillo ha querido "colar". ¿Sabes cuál es? ¡ Exacto! La orden **input()** , su función es recoger los datos que el programa pide al usuario, es decir, el número o la palabra que escribimos desde el teclado.

Componer una Canción

Componer es crear algo nuevo. Python te da la oportunidad de crear tus propios programas. Un programa es una secuencia de instrucciones, ya conoces varias instrucciones, has practicado con ellas en el Terminal y escribiéndolas en ficheros de texto. Un programa tiene tres características:

1º Recoger datos de entrada
2º Hacer cálculos con los datos, procesarlos.
3º Producir una salida para esos datos.

Antes de hacer un programa tienes que dedicar un tiempo a pensar en estas características, utiliza lápiz y papel para ayudarte a encontrar los datos que va a procesar tu programa.



Una gran herramienta para crear tus programas es el diagrama de flujo. Se trata de dibujar en un papel el funcionamiento del programa. A la izquierda esta el diagrama que calcula cual de dos números es el mayor. *Haz uno similar para saber cual es el mayor de tres números.*

Mi Primera Canción

Ya conoces varias órdenes y has practicado varios ejemplos escritos en Python. Es el momento de crear tu primer juego. Pythonillo te va ayudar a hacerlo.
El juego consiste en adivinar el número secreto. El ordenador va a elegir un número al azar entre 1 y 20.
El jugador tendrá cinco oportunidades para adivinar el número. El ordenador te dará pistas. Si no aciertas, el ordenador te dirá si el número que has elegido es mayor o menor que el secreto.

Aquí tienes un ejemplo del funcionamiento del juego. En negrita están los datos que escribe el jugador.

```
Hola !! Escribe tu nombre!
Pythonillo
Adelante! Pythonillo, estoy pensando un número entre      1 y 20.
Vamos! Adivínalo !
5
El número secreto es mayor
Vamos ! Adivínalo !
10
El numero secreto es menor
Vamos ! Adivínalo !
8
Enhorabuena Pythonillo, lo has adivinado en 3 !
```

Pythonillo dice que para escribir el código de tu programa necesitarás varias variables para guardar datos. El nombre del jugador, el número secreto, el número de intentos... Además tendrás que usar el operador == para comparar si el número secreto es igual al número que escribe el jugador. Una variable para contar intentos. ¡ A Programar !

Presta atención, te voy a mostrar como guardar un número al azar en una variable con Python:

```
secreto = random.randint(1,20)
```

Mostrar en pantalla un mensaje que contiene una variable (Lo necesitaras para el mensaje final de tu juego):

```
print “Lo has adivinado en ”,intentos, “ intentos !!”
```

Una Colección Especial

Python te da la oportunidad de coleccionar nombres, notas musicales y números. Existen tres tipos de “variables especiales” llamadas tuplas, listas y diccionarios. En primer lugar vamos a usar la tupla. Hasta ahora solo conoces como guardar un número en una variable. Pythonillo nos ha sugerido un ejemplo, para ver como almacenar varios datos en una tupla, ya sean números, letras o palabras:

```
personajes = (“Bob Esponja”,”Patricio”,”Calamardo”,”Arenita”)
mezcla = (“casa”, 'M', 75, 'p', 166.186)
```

En la linea anterior hemos declarado la variable personajes de tipo tupla. En una variable normal puedes cambiar su contenido, la tupla no permite cambios, cuando se define se establece el valor de sus elementos. En ningún momento podrás añadir o quitar elementos de una tupla.

Abre un terminal. Ejecuta la orden Python. Escribe el ejemplo anterior u otro que se te ocurra. Prueba estas ordenes, ¿Para que sirve cada una de ellas?

personajes *personajes[0]* *personajes[1:3]* *personajes[-1]*

De Nota en Nota

La utilidad que tienen las colecciones a parte de guardar múltiples datos, es poder usar esos datos de uno en uno. En el apartado ! Quiero Tocar 5 Corcheas Seguidas i aprendiste a repetir ordenes en Python mediante la sentencia **while**. Existe otra orden en Python: `for variable in secuencia_elementos:`

Como secuencia_elementos puedes utilizar tuplas como las del apartado anterior. Presta atención a estas instrucciones:

```
for valor in (1,2,3,4):
    print “Numero: ”, valor
```

```
for personaje in personajes:
    print personaje, “es un personaje”
```

El número de veces que se repiten las instrucciones que contiene la sentencia for es igual al número de elementos que contiene la secuencia. Si quieres repetir instrucciones un numero determinado de veces tienes que usar la orden **range(numero_de_iteraciones)**. Observa:

```
for variable in range(10):
    print variable
```

```
for variable in range(5,10):
    print  variable
```

```
for variable in range(1,50,5):
    print  variable
```

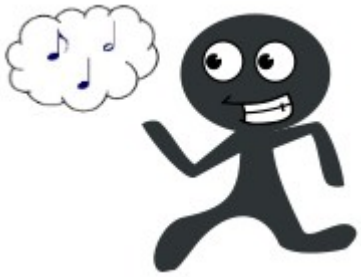
Pythonillo quiere que le ayudes a enseñar las tablas de multiplicar a sus amigos. Usa la orden for para crear un programa que imprima en pantalla la tablas de multiplicar del 9.

Coleccionando Notas Musicales

En apartados anteriores hemos practicado con un tipo de colección llamada tupla, la cual te permite almacenar datos, pero no permite modificar, añadir o eliminar los valores que contiene.

Para poder hacer este tipo de operaciones usaremos la colección "lista".
Veamos como crear una lista:

```
miFamilia = ["Homer", "Marge", "Lisa", "Bart"]
```



A continuación aparece la sintaxis para realizar las distintas operaciones sobre una lista.

```
miFamilia.append("Rodolfo Chiquilicuatre")
```

```
miFamilia.remove("Homer")
```

Si tu intención es modificar un elemento de los que contiene la lista, debemos indicar cual es la posición en la que se encuentra dentro de nuestra lista. Si escribes la orden miFamilia[0] = "Harry" , sustituirás en la primera posición de la lista "Homer" por "Harry".

Las tuplas y las listas usan la misma sintaxis para acceder a sus elementos, repasa el apartado Una Colección Especial para recordar el uso de los corchetes.

Escribe el programa del juego "Adivina el número" pero almacena los números que introduce el jugador en una lista. Muestra un mensaje con todos los números que se han utilizado.

Organiza tus melodías

Una función es un conjunto de ordenes que actúan conjuntamente para conseguir un objetivo. Puedes usar las funciones para organizar tus programas. Una función tiene un conjunto de ordenes de Python. Puedes entender el significado de una función como un pequeño programa que realiza una sola tarea. Las Listas almacenan variables y las funciones almacenan ordenes de Python.

Esta es la sintaxis para crear una función:

```
def mayorDeEdad(anios):
    if anios >= 18 :
        print "Eres mayor de edad"
    else:
        print "Eres un pezqueñin"
```

Una vez que ya has creado la función, cada vez que necesites usarla solo tendrás que escribir el nombre de la función seguido de paréntesis.

```
#Llamada a la función
mayorDeEdad(12)
```

Paso de argumentos; puede que tu función realice una serie de cálculos, pero para hacer esos cálculos la función necesita uno o varios datos. Puedes crear un función para que calcule la tabla de multiplicar y que esa función reciba un argumento que sea el número de la tabla que quieres calcular.

Crearas muchas funciones que realizaran cálculos, para obtener el resultado de esos cálculos mediante la función, necesitaras la orden **return**. Imagina una función que le pases como argumento una cantidad en pesetas y te devuelva la misma cantidad en euros. El resultado de los cálculos puedes asignar a una variable.

```
def peseta_a_euro(pesetas):
    return pesetas / 166.386
```

¿La Flauta es un Objeto ?

En capítulos anteriores has practicado como guardar datos en variables y como guardar ordenes en funciones. El siguiente paso es guardar datos/variables y ordenes/funciones dentro de objetos. Una pelota es un objeto, que tiene unas características determinadas, por ejemplo, color, peso o tamaño. Estas características las podemos almacenar en variables dentro de un objeto pelota en Python.

Por otro lado, con una pelota se pueden hacer muchas cosas: lanzarla, inflarla o golpearla. En Python reflejaremos estas acciones con una función para cada una de ellas, que esta dentro de nuestro objeto. Las funciones también son conocidas como métodos del objeto o clase.

```
class pelota:
    def __init__(self):
        self.color = "rojo"
        self.tamanio = 10
        self.tamanio = 5
    def lanzar(metros):
    def inflar (aire):
    def golpear():
miPelota = pelota()
miPelota.color
miPelota.lanzar(15)
miPelota.inflar(9)
```

La orden para crear un objeto en Python es **class.** Además aparece un método especial, se llama def __init__(self), ha este método se le llama constructor de la clase. Se utiliza para dar un valor inicial a las propiedades de nuestro objeto.

En la columna de la derecha puedes ver las ordenes necesarias para usar nuestro objeto después de haberlo creado. Tendrás que utilizar el nombre de tu objeto seguido de un punto y la propiedad o función que quieras usar. Si te has fijado, lo métodos acaban con paréntesis y las propiedades no.

Serpentario

En un serpentario podemos encontrar muchas serpientes. En Python podemos hacer algo parecido con nuestros objetos. Podemos usar un fichero para escribir todas nuestras clases, a estos ficheros les llamaremos módulos. Cuando necesitemos usar uno de nuestros objetos en un programa usaremos la orden **import** seguida del nombre de nuestro archivo.

Python, tiene una gran variedad de módulos que podemos usar en nuestros programas. En próximos capítulos usaremos los módulos easygui y pygame para que puedas crear tus propios videojuegos. Si quieres obtener ayuda sobre un modulo determinado en el terminal escribe la orden **help()**. Después escribe el nombre del modulo. Prueba con el módulo time y la función sleep.
Escribe un programa donde se use esta función.

```
time.sleep(15)
```