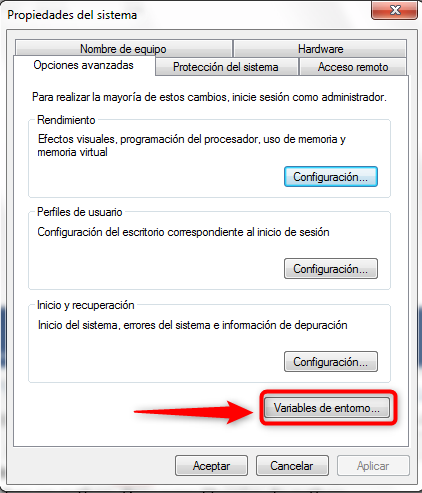
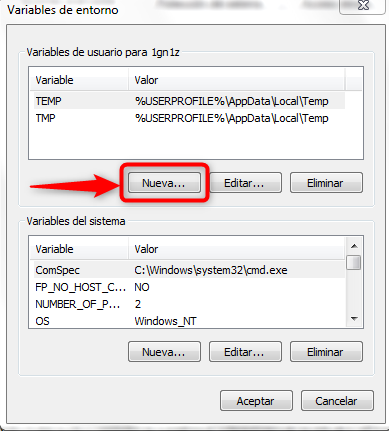
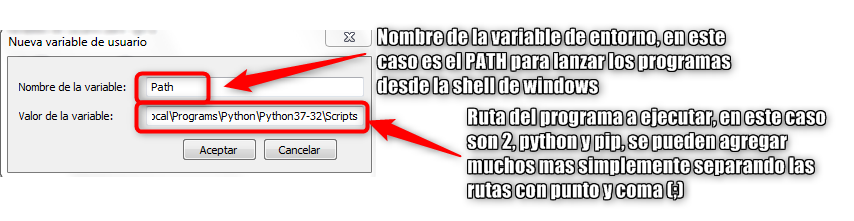
**NOTAS DEL CURSO DE PYTHON IMPARTIDO POR UDEMY**

**PIP** es un manejador de paquetes, lo unico que hace PIP es descargar e instalar programas hechos en python. Descargar librerias de python

**VARIABLES DE ENTORNO**

En mi caso no tenia ninguna de las variables de entorno de Python, para agregarla simplemente escribi **variables de entorno** en el buscador de Windows y luego entrar a la opción **variables de entorno**:





Para añadir más de una variable de entorno al PATH de W7 simplemente la agregamos, separandola de la(s) otra(s) con un punto y coma (**;**). Ejemplo:

C:\Perl64\site\bin**;**C:\Python27

Ejemplo personal con Python 3.7.3 y PIP en la variable **Path**:

C:\Users\1gn1z\AppData\Local\Programs\Python\Python37-32\**;**C:\Users\1gn1z\AppData\Local\Programs\Python\Python37-32\Scripts

**NUMEROS Y OPERADORES ARITMÉTICOS**

En python tenemos suma, resta, multiplicacion y division. Ademas, como en la vida real tenemos lo que se denomina como **Precedencia Aritmetica**. Por ejemplo, cual es el resultado de esta operación:

8 + 2 \* 10

La respuesta NO es 100, la respuesta a esto seria 10\*2+8= 28.

Primero que todo se toman las operaciones que estén entre paréntesis () si es que los hay, es lo primerísimo que se toma en cuenta.

Luego, se toma la multiplicación y la división conforme aparezcan de izquierda a derecha.

Y finalmente la suma y resta, igual conforme aparezcan de izquierda a derecha.

Python, al realizar una división **SIEMPRE** devuelve un flotante, osea, **un numero decimal**.

Si queremos que la división en Python devuelva un **entero** en una división podemos usar **doble diagonal //**.

19 / 2 = 9.5 **19 // 2 = 9**

El numero que devuelve Python al usar la doble diagonal, es redondeado, **siempre devuelve solo la parte entera del resultado**.

Tambien tenemos el signo de **potencia**. Por ejemplo si queremos hacer 6\*6\*6 = 216

Para usar la potencia en Python simplemente usamos **doble signo de asterisco \*\***

6\*6\*6 = 216 **6\*\*3= 216**

**VARIABLES**

En el mundo real, cuando vemos problemas matemáticos como:

X = 10 z = 5 velocidad = 30kmps

Estan serian las variables, x, z, altura.

Se llaman variables por que su valor **varia**, la velocidad de un auto varia, el valor de x varia, etc.

En Python, para delcarar una variable simplemente escribimos

El **nombre de la variable**, y el **valor de la variable**, por ejemplo:

X = 2

Osea, el valor de la izquierda (2), se guarda en la variable de la derecha (X).

Con las variables, **el valor queda almacenado**, pudiendo acceder a esa variable en cualquier momento. A diferencia de las operaciones aritméticas simples que vimos anteriormente.

Tambien podemos asignarle una operación aritmética a una variable, por ejemplo:

**>>>** y = 6 \*\* 3

**>>>** y

216

**>>>**

Ahora, no hay que confundir valor con **asignación**, por ejemplo:

x = 2 esto imprime 2

si ponemos x + 2 = 4

pero si volvemos a imprimir x, vale nuevamente 2. Ya NO vale 4.

Esto pasa xq hicimos una suma solamente, **no asignamos con un nuevo valor la variable**.

Para que x valga 4, podemos hacerlo asi:

**x = x + 2**

Ahora si, x vale el valor de x (2) + 2. Esto se le conoce como asignación.

Ahora bien, si hacemos una operación como:

**z = 19/2 = 9.5**

A este resultado se le conoce como **Punto Flotante o FLOAT**, asi se le conoce en Python a todo **numero decimal**.

Tambien tenemos el tipo de dato **Cadena o String**, que simplemente son **cadenas de texto**, por ejemplo:

a = “Hola Mundo”

En Python, a diferencia de otros lenguajes, no tenemos que declarar que tipo de variable vamos a usar (String, Float, Int, etc.) Simplemente escribimos el nombre de la variable y la asignación o valor de la misma.

Tambien podemos **asignar variables a variables**.

**>>> x = 10**

**>>> y = 20**

**>>> z = x + y**

**>>> z**

**30**

Podemos usar variables para operaciones aritméticas y asignarla a otras variables.

**CADENAS**

Las cadenas se usan para muchas cosas, por ejemplo, generar contraseñas y usarlas, guardar el nombre del servidor al que vamos a accesar, guardar nombres de tablas en bases de datos, para llaves de un diccionario, etc.

En Python NO hay ninguna diferencia entre usar comillas dobles o simples para asignar una cadena.

**>>> nombre = "1gn1z"**

**>>> nombre**

**'1gn1z'**

**>>> nombre = '1gn1z'**

**>>> nombre**

**'1gn1z'**

Pero, en algunos casos muy particulares, tal vez necesitemos usar ambas comas, por ejemplo:

**>>> frase = "Ella me dijo: 'Hola' "**

**>>> frase**

**"Ella me dijo: 'Hola' "**

**>>> frase = 'Ella me dijo; "Hola" '**

**>>> frase**

**'Ella me dijo; "Hola" '**

Si por ejemplo necesitamos obligatoriamente usar comillas dobles o simples, necesitamos usar el carácter conocico como **Carácter de escape (\)** podríamos hacer algo asi:

>>> frase = "Ella me dijo: \"Hola\" "

>>> frase

'Ella me dijo: "Hola" '

Esto le indica a Python que **después de la diagonal invertida (Carácter de escape)** Las comillas las ignore y las tome como parte del texto que tenemos.

El carácter de escape, para indicar texto literal con comillas, ya sean dobles o sencillas se usa asi:

variable = "cadena normal **\"Texto con comillas\"** "

Para indicar el texto entre comillas, siempre se escribe asi:

a = "Texto normal" **\"**Texto con comillas**\"**"

Pero además, el carácter escape se usa para otras cosas, por ejemplo, decirle a Python que queremos hacer algo especial con la letra que viene después, por ejemplo, para hacer un salto de línea.

>>> frase = "Ella me dijo: **\n**\"Hola\""

>>> print (frase)

Ella me dijo:

"Hola"

El salto de línea (\n) se usa para indicar que, **todo lo que esta después de la \n**, se imprima en una nueva línea. Este se muestra **siempre con la instrucción PRINT**.

Tambien podemos usar el símbolo de suma (+), para unir 2 cadenas o 2 variables, a esto se le conoce como **concatenación**.

**>>> nombre = "1gn1z"**

**>>> saludo = "Kiubo! "**

**>>> print (saludo + nombre)**

**Kiubo! 1gn1z**

Aquí ya agregue yo el espacio en la variable saludo, pero se puede agregar el espacio desde el print:

>>> nombre = "1gn1z"

>>> saludo = "Kiubo!"

>>> print(saludo + **" "** + nombre)

Kiubo! 1gn1z

>>>

Simplemente debemos **concatenar un par de comillas con el espacio dentro de ellas**.

Tambien podemos usar el símbolo de multiplicación para multiplicar una cadena, por ejemplo:

>>> print(nombre \* 3)

1gn1z1gn1z1gn1z

Si a la izquierda tiene una cadena, y a la derecha tiene un operador aritmético de suma o multiplicación, lo que hace Python es que va a sumar o multiplicar esa cadena el numero de veces especificado.

**CONDICIONALES**

Las condiciones nos sirven para determinar si un valor de una variable se cumple o no se cumple, puede ser tipo booleana, entero o flotante, incluso cadenas.

Las condiciones en Python se escriben con la palabra reservada **IF** (Si), después lo que queremos testear (en este caso una variable), y después lo comparamos con algún signo de comparación, en este caso igual se escribe con doble signo de igual **==**.

El signo de igual, cuando es uno, sirve para **asignar un valor**. En este caso a la variable **edad** le asignamos el valor de **21**, es decir, el dato 21 se guarda en la variable edad ok.

edad = 21  
  
if edad == 21:  
 print("Puedes entrar a pistiar!")

Como vemos, al terminar la condición if, es necesario cerrarla con dos puntos **:**.

Como vemos, la condición se cumple, asi que se ejecuta el código siguiente.

OJO! El bloque de código que se ejecutara, es el código donde empieza con un **TAB**, después del if, ese código se ejecuta.

edad = 21  
  
if edad == 21:  
 print("Puedes entrar al bar")  
 print("Y tambien puedes beber")

**TODO LO QUE LLEVE TAB ANTES ES LO QUE SE EJECUTA EN LA CONDICIONAL IF.**

edad = 15  
  
if edad == 21:  
 print("Puedes entrar al bar")  
 print("Y tambien puedes beber")

Aquí, la condición NO se cumple, por lo tanto no se imprime nada, pero que pasa si le quitamos el TAB a una línea de código?

edad = 15  
  
if edad == 21:  
 print("Puedes entrar al bar")  
print("Y tambien puedes beber")

**Y tambien puedes beber**

El código SI se ejecuta, ya que **al quitarle el TAB ya NO es parte de la condicional IF**.

Tambien podemos combinar mas de una condición, con la palabra reservada **ELIF** (Else If, Si no). Esto se le llama **ANIDACION.**

NO es recomendable anidar demasiadas condiciones, ya que se hace ilegible el código ok.

if edad < 18:  
 print("No puedes entrar :(")  
elif edad >= 21:  
 print("Puedes entrar al bar")  
 print("Y tambien puedes beber")