CURSO FUNDAMENTOS DE PYTHON

FUNDAMENTOS DE PYTHON - tabla de contenido

[VARIABLES 3](#_Toc129446582)

[TIPOS DE DATOS 4](#_Toc129446583)

[STRINGS 5](#_Toc129446584)

[NUMBERS 7](#_Toc129446585)

[BOOLEANS 8](#_Toc129446586)

[TRANSFORMACION DE TIPOS 9](#_Toc129446587)

[PLAYGROUND: IMPRIME EL FORMATO ADECUADO 11](#_Toc129446588)

[OPERADORES Y CONDICIONALES 12](#_Toc129446589)

[OPERADORES ARITMETICOS 12](#_Toc129446590)

[OPERADORES DE COMPARACION 14](#_Toc129446591)

[COMPARANDO FLOTANTES 16](#_Toc129446592)

[POR LA FORMA DE LOS STRINGS 16](#_Toc129446593)

[PROCESO MATEMÁTICO 17](#_Toc129446594)

[OPERADORES LOGICOS AND & OR 17](#_Toc129446595)

[AND 17](#_Toc129446596)

[OR 18](#_Toc129446597)

[OPERADOR LOGICO NOT 19](#_Toc129446598)

[REPASO DE LO APRENDIDO 20](#_Toc129446599)

[CONDICIONALES 21](#_Toc129446600)

[IF 21](#_Toc129446601)

[ELSE 22](#_Toc129446602)

[ELIF 22](#_Toc129446603)

[PROYECTO CONDICIONALES 23](#_Toc129446604)

[STRING RECARGADO 24](#_Toc129446605)

[INDEXING Y SLICING 27](#_Toc129446606)

[LISTAS 29](#_Toc129446607)

[METODOS DE LISTAS 30](#_Toc129446608)

[TUPLAS 33](#_Toc129446609)

[PROYECTO: TUPLAS 35](#_Toc129446610)

[DICCIONARIOS: DEFINICION Y LECTURA 36](#_Toc129446611)

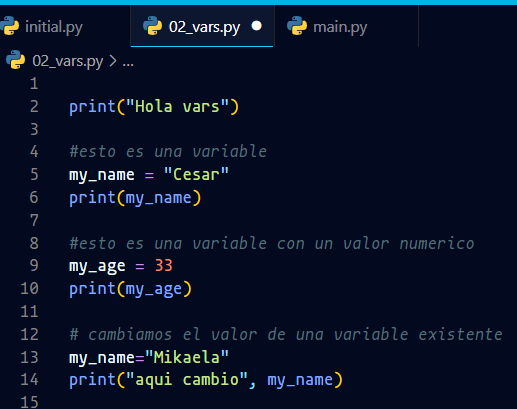
[DICCIONARIOS: INSERCIÓN Y ACTUALIZACION 37](#_Toc129446612)

[LOOPS: WHILE 39](#_Toc129446613)

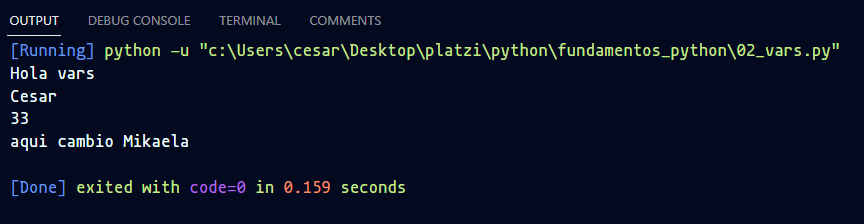
[LOOPS: FOR 41](#_Toc129446614)

[CICLOS ANIDADOS 43](#_Toc129446615)

VARIABLES



Las variables pueden reemplazar sus valores como en my\_name



Utiliza el terminal de windows



TIPOS DE DATOS

Cadenas

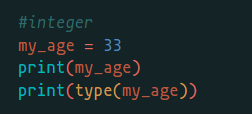






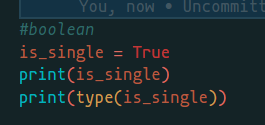
Imprime el tipo de valor de la variable my\_name. El tipo es str (string).





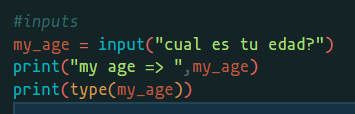


En el caso de los booleanos el valor debe empezar con mayúscula (True o False)



El resultado es True de clase bool



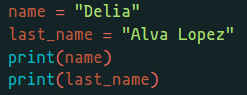


Nota que el valor ingresaste es 33 pero que el tipo devuelve el valor en string. Siempre pasa cuando pasas un numero por un input



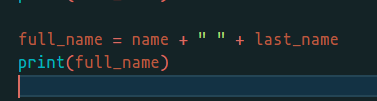
STRINGS

Puedes almacenar texto dentro de una variable, el texto debe estar entre comillas.





Puedes usar + para concatenar cadenas. En este caso se está agregando un espacio en blanco

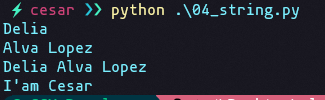




A veces se debe emplear apostrofe sobre todo en la lengua inglesa. Para ello se pide que se utilice las comillas dobles para envolver el texto y dentro el apostrofe







En caso que se necesite usar comillas dobles en la cadena se recomienda envolver el texto con comillas simples





Amazon web service

Cesar.contreras.medina.3600@gmail.com

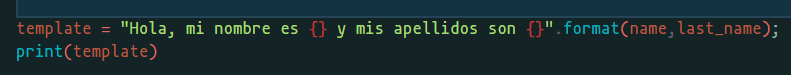
@rasec333CCM

Puede concatenarse cadenas con el signo +



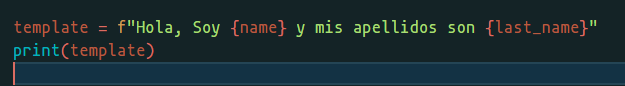


De la misma manera se puede hacer lo siguiente





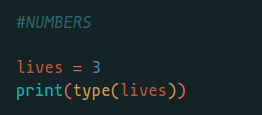
Una tercera forma es colocar la letra f (de formato) y dentro de la cadena ingresar {nombre\_de \_variable}





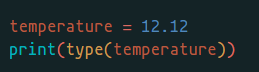
NUMBERS

Los números no llevan comillas



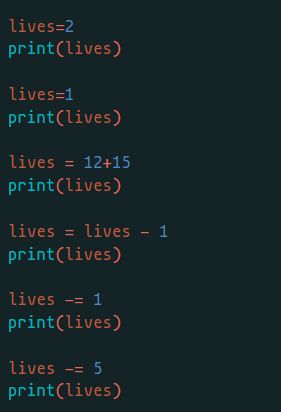


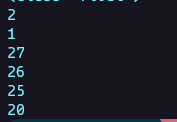
En el caso de agregar decimales el tipo que tendrán en float



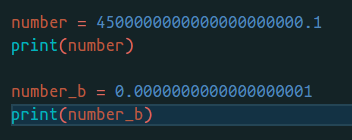


Aquí tenemos varios ejemplos, la mas interesante suma 2 numeros y resta -= 1





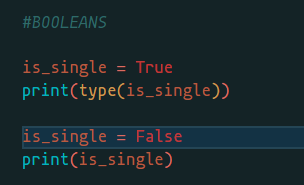
Cuando los números son muy grandes o pequeños Python los va a expresar de la sgte manera.





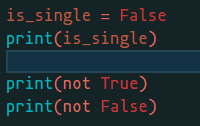
BOOLEANS

Los booleanos siempre tiene que empezar con la T o F como mayúscula: “True” o “False”



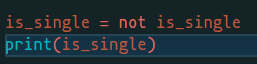


Se puede cambiar el valor de la variable booleana usando el operador NOT





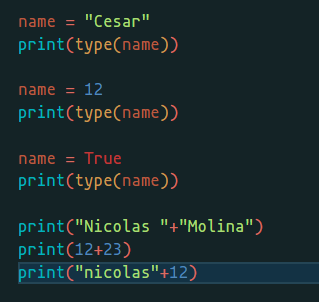
Otro ejemplo



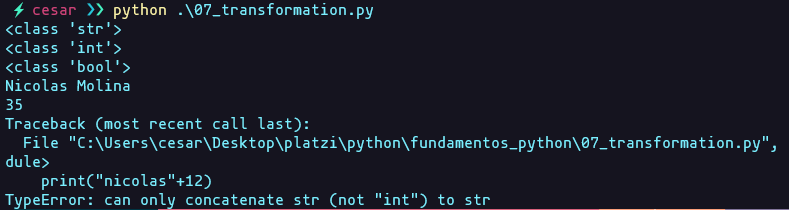


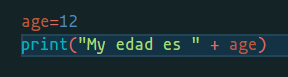
TRANSFORMACION DE TIPOS

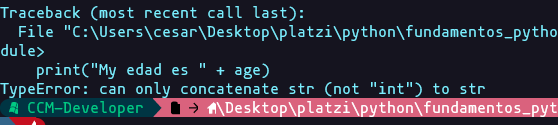
Todos los ejemplos son comunes menos el ultimo. Allí habrá un error debido a que Python no sabrá si transformarlo en un numero o en una cadena.



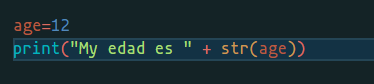
El error dice que solo se puede concatenar strings (no números enteros) con strings.







Como dijimos antes la solución es tener los elementos en un mismo tipo. En este caso transformaremos age a cadena.





De esta forma la función format sabe qué hacer con age

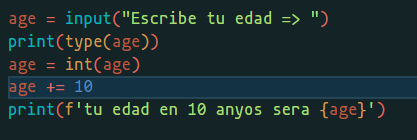




En este ejemplo pedimos la edad mediante un input, se imprime el tipo que es string

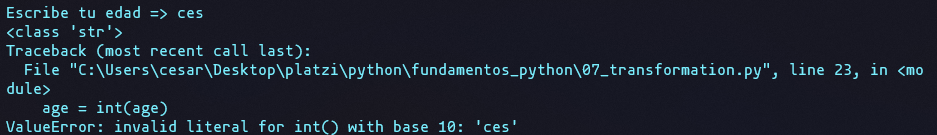
Luego transformamos age a entero y le sumamos 10

Imprimimos mediante el método format

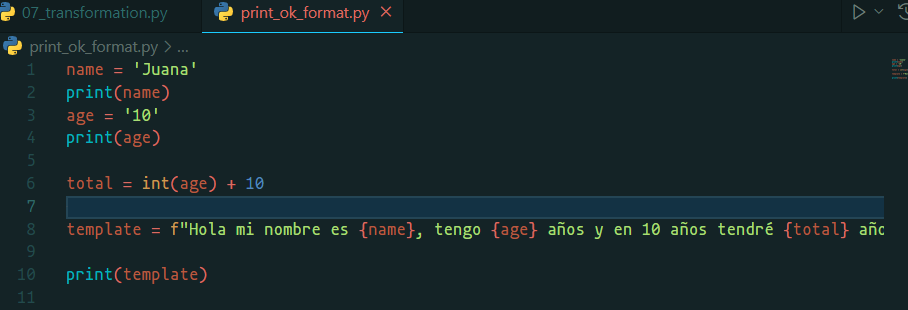




El problema ocurre cuando el usuario no ingrese un numero como una edad sino de otro tipo como un string o un booleano.



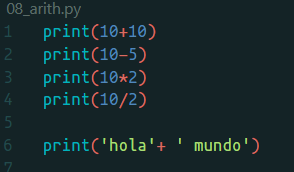
PLAYGROUND: IMPRIME EL FORMATO ADECUADO

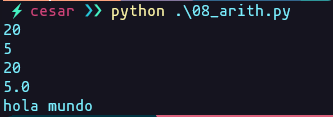


OPERADORES Y CONDICIONALES

OPERADORES ARITMETICOS

Permiten ejecutar cualquier calculo

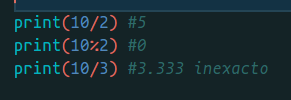




Repite según la cantidad con la que lo multipliques





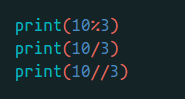




La operación con signo **%** devuelve el residuo de la división

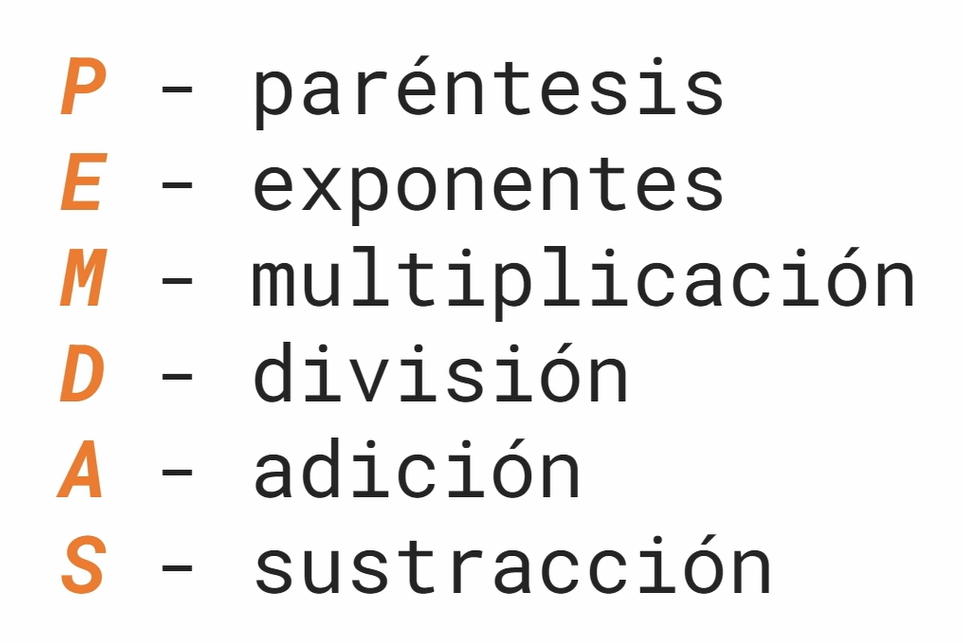
**/** devuelve la división así sea flotante o entera

**//** devuelve el entero del resultado de la división, obviando los decimales si es que existieran





Este es el orden de ejecución de las operaciones. Primero se ejecuta los paréntesis hasta llegar a la sustracción.



La sgte operación tiene como resultado



El resultado es:



Primero ejecuta => 2\*\*3 = 8

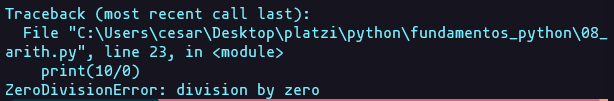
7/1//4 => 1

Al final se operara de la Sgte. forma:

8 +3 -1 = 10

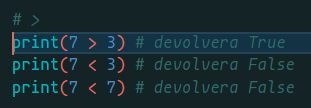
Dividir entre cero traerá error:





OPERADORES DE COMPARACION

Mayor que



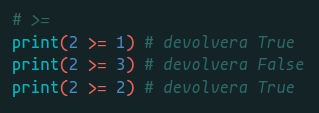


Menor que



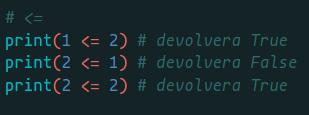


Mayor o igual que



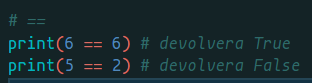


Menor o igual que



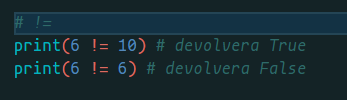


igual



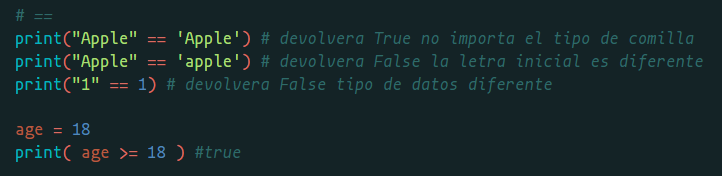


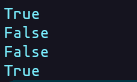
diferente



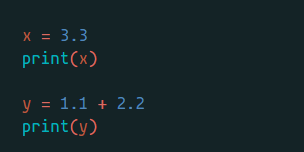


Igualdad tomando en cuenta el tipo de datos





COMPARANDO FLOTANTES



Son diferentes. Los dos tiene una precisión decimal diferente.



Al compararlos el resultado es false





Hay dos maneras de igualar:

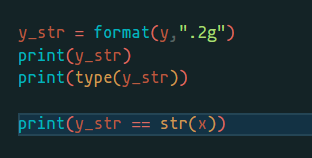
POR LA FORMA DE LOS STRINGS

Seria cortarle los string que sobran para que coincidan

Le damos formato a y con el método format y .2g

Visualizamos el valor y el tipo de “y\_str”

Luego transformamos x a string para hacer la comparación que devolvera True



Básicamente estamos cortando el valor resultante de Y que es un decimal largo.

Con el método format, este método nos devolverá un string, luego transformamos el valor de x a string también y hacemos la comparación e imprimimos el resultado.

Este proceso no es matemático

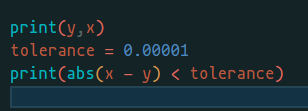


PROCESO MATEMÁTICO

Primero imprimimos los valores y, x

Luego creamos una variable llamada tolerancia el cual valdrá 0.00001

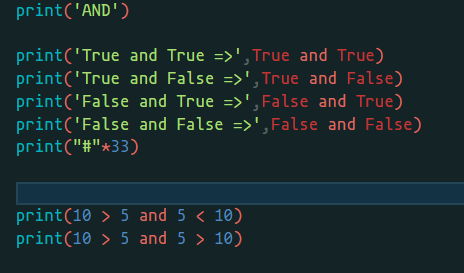
O sea que si la resta entre x, y es menor al valor de la tolerancia entonces es True

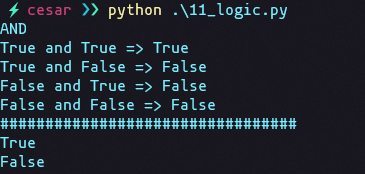




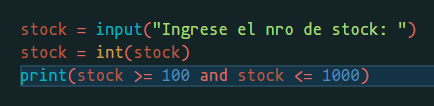
OPERADORES LOGICOS AND & OR

AND





En el ejemplo capturamos el nro de stock, luego se transforma en un entero y dependiendo que cumpla la regla de negocio devuelve True o False

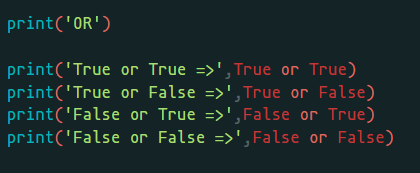


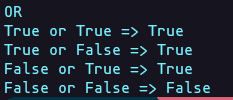
En el ejemplo ingrese 400 y cumple las reglas de negocio por lo tanto devuelve True.

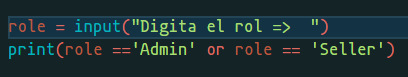


OR

Evalúa si alguno de es verdadero

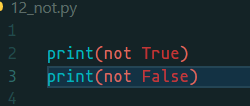




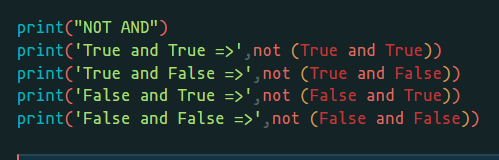


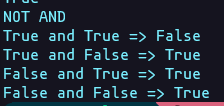


OPERADOR LOGICO NOT



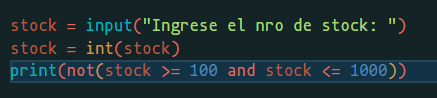






En este ejemplo ingresamos un numero como stock y evaluamos

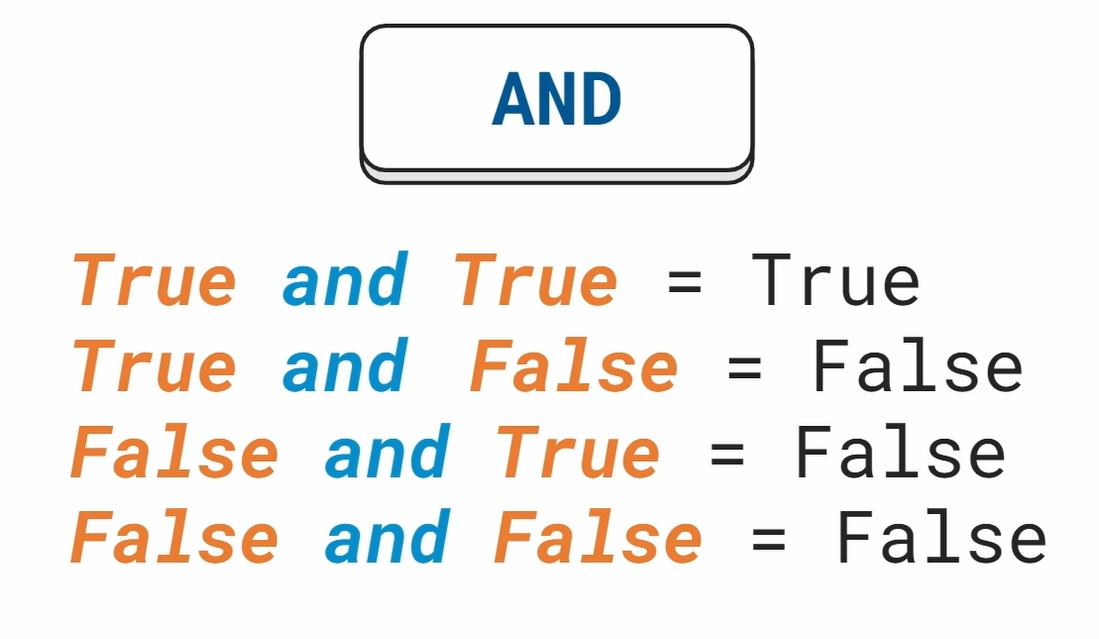
Si NO se encuentra en el rango de mayor o igual a 100 y menor o igual a 1000 entonces devuelve **True**. En caso contrario devuelve **False**

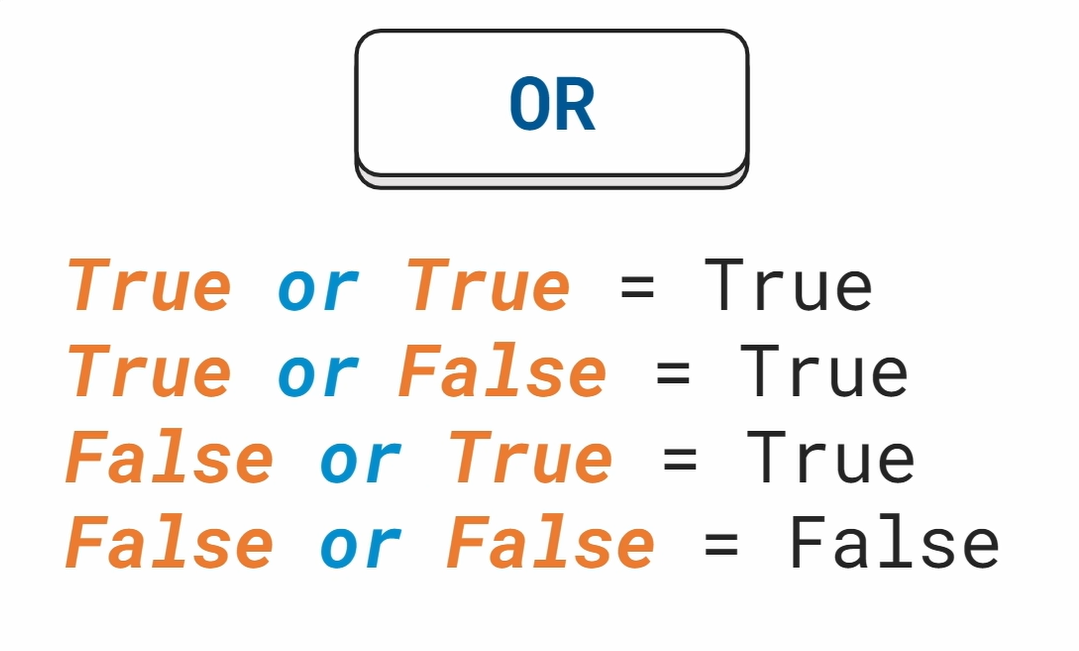


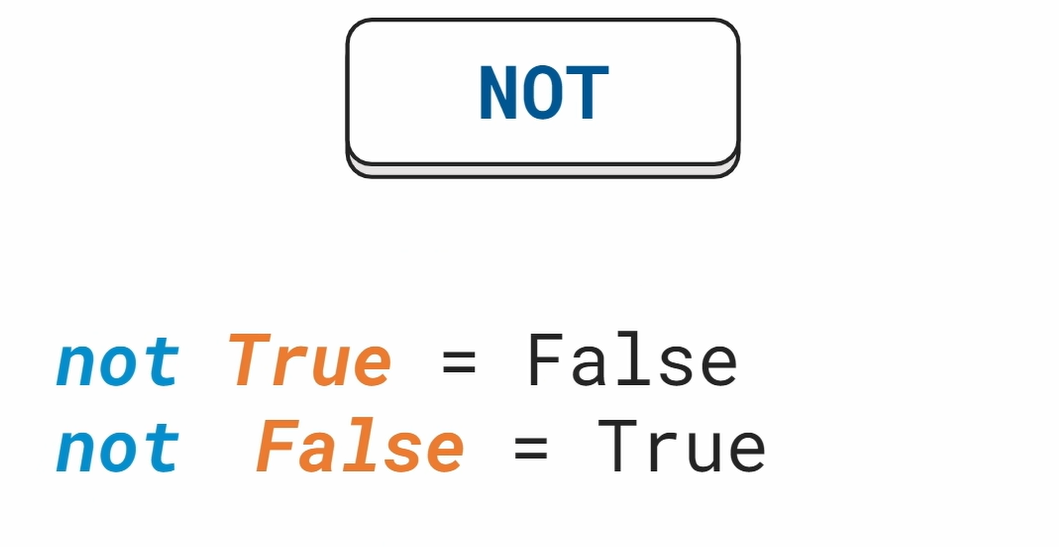




REPASO DE LO APRENDIDO



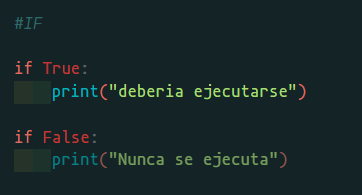




CONDICIONALES

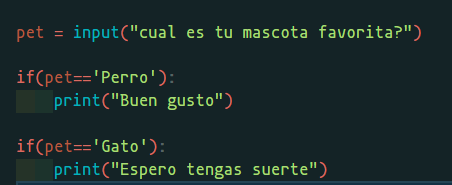
IF

Las condicionales van a evaluar si es True



Nunca muestra la segunda línea porque es False





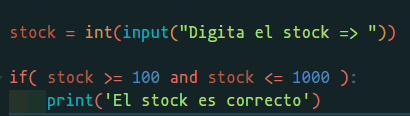






Ingrese stock y se transforma en un entero

Si el stock está dentro del rango muestra el mensaje

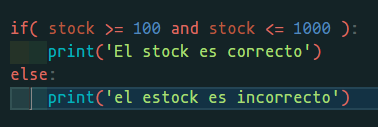


El stock está dentro del rango, entonces muestra el mensaje en caso contrario no mostrara nada.



ELSE

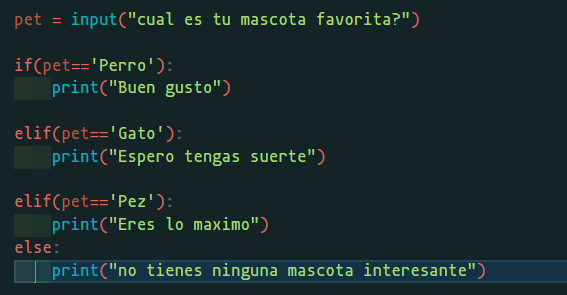
Se puede agregar un else





ELIF

También se puede agregar elif (else if en otros lenguajes)

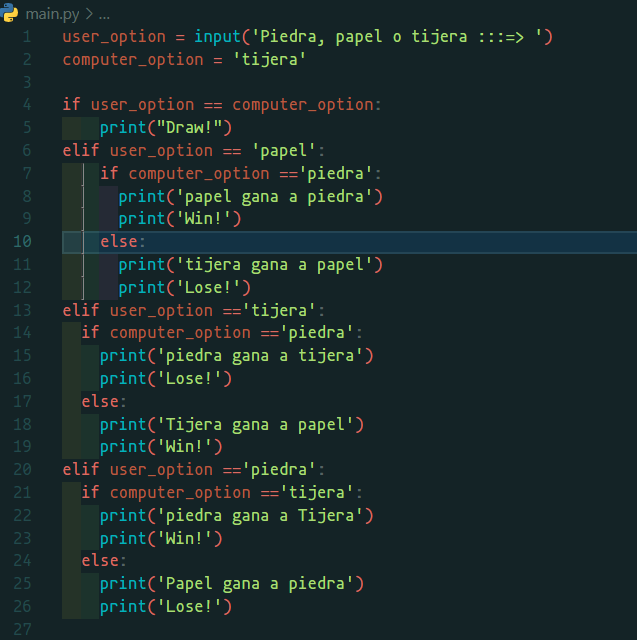


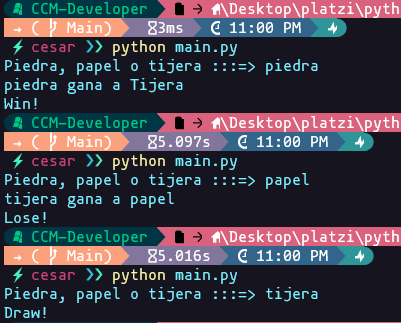


PROYECTO CONDICIONALES

Se ingresa una opción y se compara contra la opción del computador

Según la comparación devuelve un resultado.

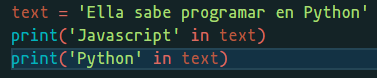




STRING RECARGADO

Python puede verificar la existencia de un subtexto.

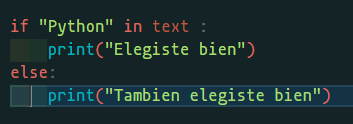
En esas cosas se esta preguntando a Python si la palabra Javascript se encuentra en el texto “Ella sabe programar en Python”.



En el primer caso es falso y en el segundo es verdadero.



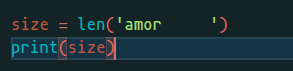
Podemos utilizarlo en una condicional





Si se colocara otra cosa en vez de “Python” escribiría el segundo print

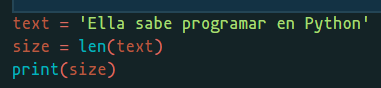
También se pueden contar la cantidad de caracteres.



El resultado será 9 por que también cuenta espacios en blanco.

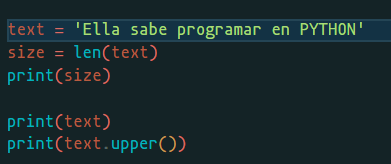


También puede contar el valor de una variable que contiene una cadena



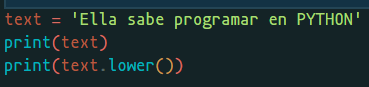


Podemos poner en mayúscula un texto con el método upper()



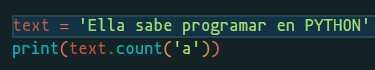


También podemos ponerlos en minúscula con lower()



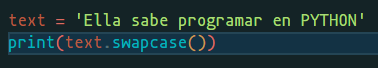


Nro de apariciones que tiene una letra





Swapcase transforma de mayúscula minúscula y viceversa





Pregunta si un texto empieza con alguna palabra en específico. Devuelve un booleano.

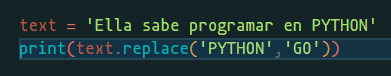




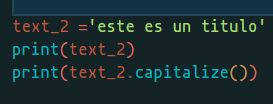




Reemplaza los textos









Coloca en mayúscula todas las primeras letras de las palabras





Isdigit() Devuelve un booleano si es que el texto evaluado es un numero

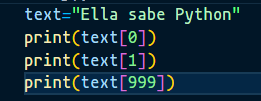


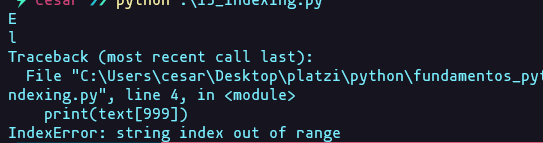


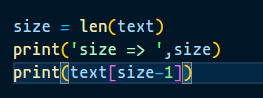




INDEXING Y SLICING





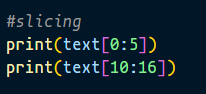




Con -1 va directamente a la última posición

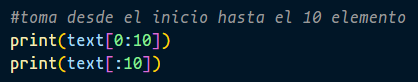








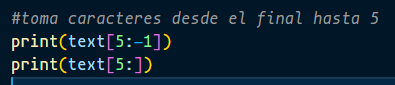
Si quieres tomar un elemento desde el inicio hasta el 10 puedes escribirlo de estas 2 formas:





Si quieres tomar desde el final hasta 5 caracteres atrás entonces puedes colocar [ 5 : -1 ] sin embargo esto tiene un error y es que no tomara en cuenta el ultimo carácter.

La segunda solución [ 5 : ] si tomara desde el final hasta cinco caracteres atrás.





Esto imprimirá todo

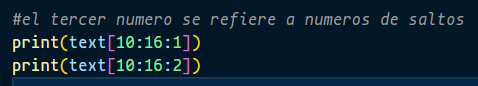




Saltara desde la posición 10 y tomara caracteres hasta la posición 16.

El tercer valor dirá que tomará de uno en uno.

En el segundo ejemplo saltara de 2 en 2.



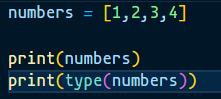


También se puede imprimir el texto de 2 en 2 tomando en cuenta todo el texto

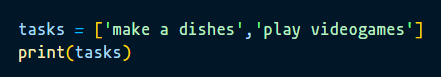




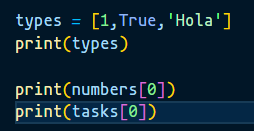
LISTAS





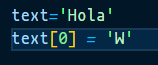


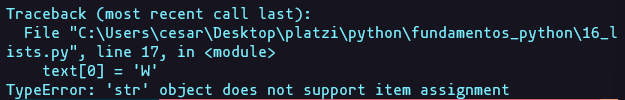


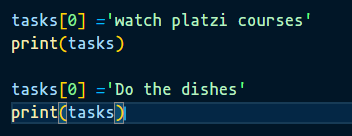




Las cadenas no son mutables

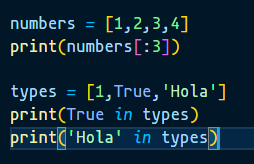


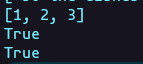




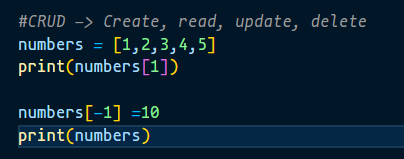


Puedes buscar si un elemento se encuentra dentro de una lista utilizando “in”





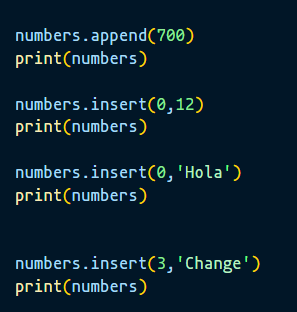
METODOS DE LISTAS

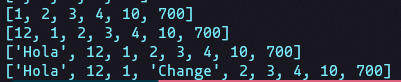




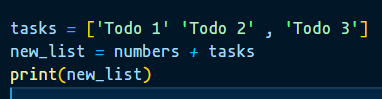
Append agrega el numero 700 al final de la lista numbers

Mientras que insert agrega algún elemento desde la posición que le indica, ojo no reemplaza.



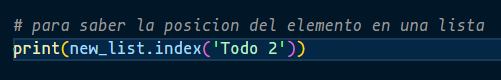


También puede unir listas con el operador +



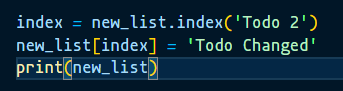


Si necesitas saber la posición de un elemento puedes usar index.



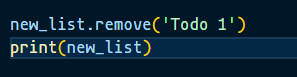


Reemplaza Todo 2 extrayendo su posición y reemplazándolo con TodoChanged



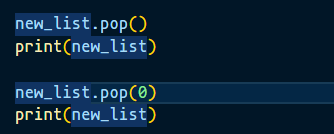


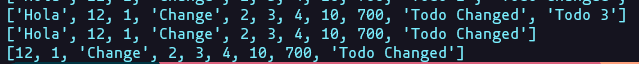
Se puede remover un elemento con remove()





Con pop también se puede remover. Si no se indica la posición borra el ultimo en caso contrario remueve el elemento según el index que se indique



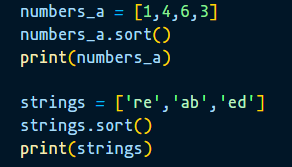


Pone al revés el contenido de una lista





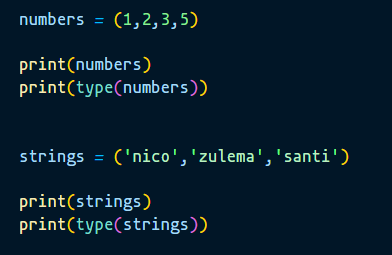
Ordena una lista según su contenido. No puede ordenar cuando tiene elementos de tipos distintos.

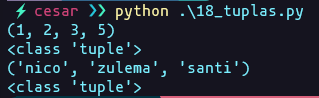


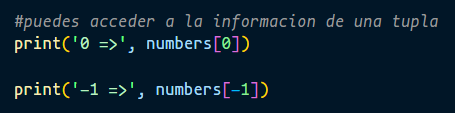


TUPLAS

Las tuplas son inmutables por lo tanto no puede agregarse o quitarse sus elementos.

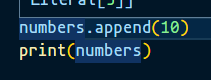


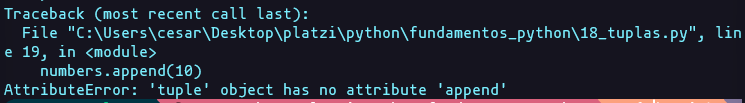


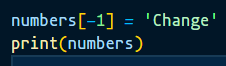


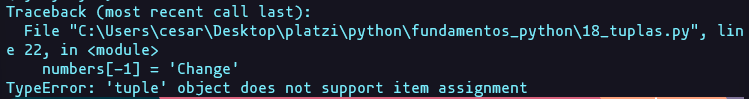


Las tuplas no pueden modificarse

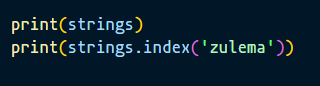






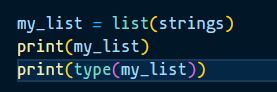


Puedes acceder a la posición de un elemento de una tupla



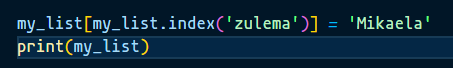


Se puede cambiar de tupla a lista con el método list()



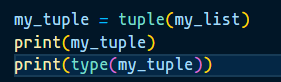


Al ser una lista podemos acceder al index de Zulema y reemplazarlo por mikaela





Luego podemos pasar la lista a una tupla con el método tuple()



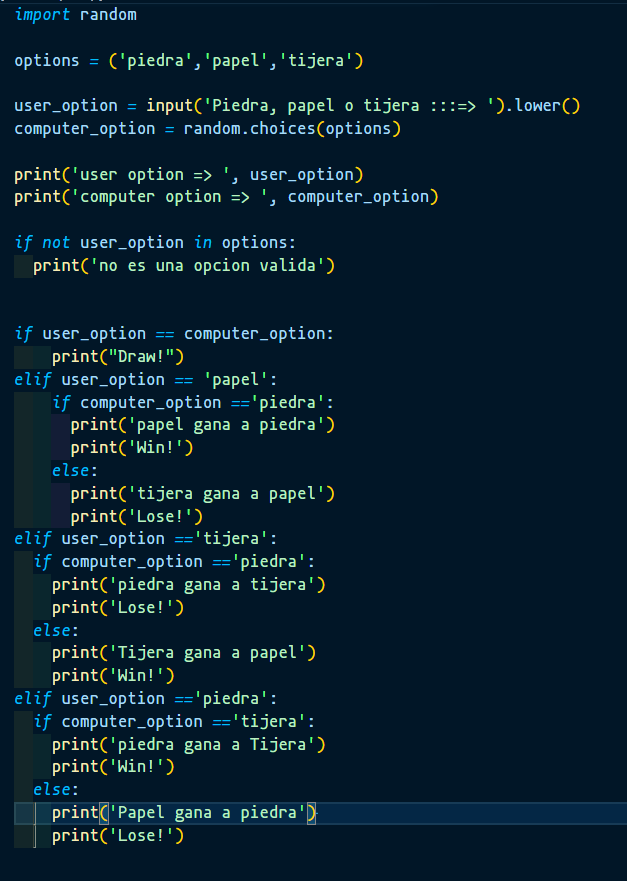


PROYECTO: TUPLAS

Se importa random

Random.choice escoge al azar dentro de la tupla options y lo guarda dentro de la variable computer\_options

Si no existe el valor ingresado del usuario dentro de la tupla entonces muestra un mensaje de error



DICCIONARIOS: DEFINICION Y LECTURA

Se comportan como diccionarios. Lo puedes comparar como un objeto de javascript.



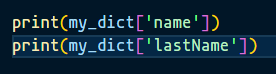


Puedes mostrar la cantidad de elementos de un diccionario con len()



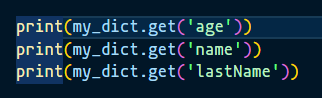


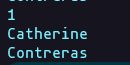
También puedes acceder a los datos utilizando el nombre de la key para extraer su valor asociado.



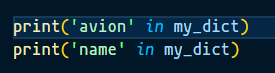


También se puede acceder a los datos usando get. Si no existe un valor devolverá none a diferencia que en la option anterior te mandara error.





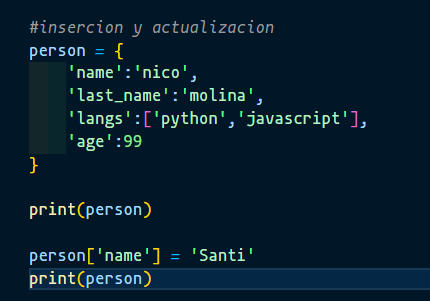
También se puede buscar dentro de un diccionario. Esto devolverá un Booleano

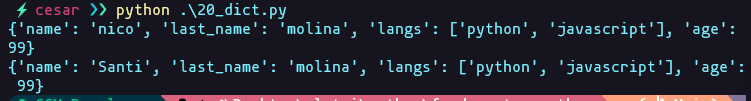


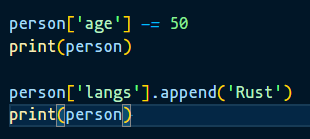


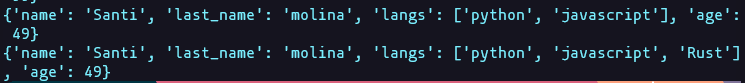
DICCIONARIOS: INSERCIÓN Y ACTUALIZACION

Aquí estamos actualizando el valor de name “nico” por “santi”.



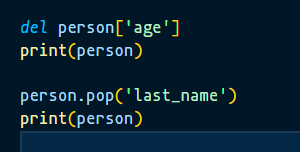






Existen 2 maneras de borrar un elemento del diccionario,

Mediante el uso de “del” y mediante el método “pop”



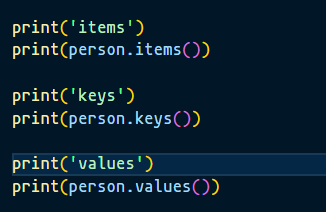


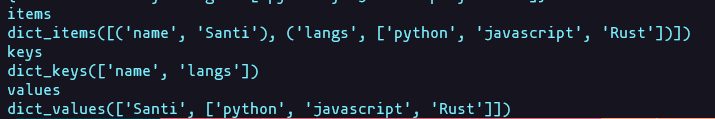
Vamos a listar los

Ítems devolverá en pares de tuplas

Keys devuelve un arreglo con las keys

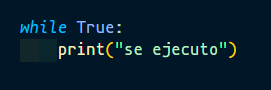
Values devuelve un array con los valores de las keys

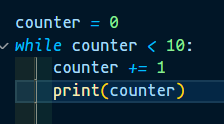


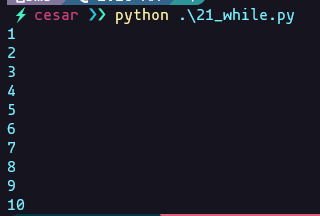


LOOPS: WHILE

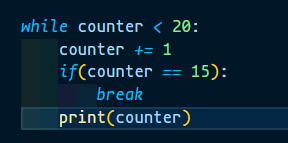
**While** significa “mientras”. mientras sea True entonces imprime “se ejecutó”. Creando un ciclo infinito.

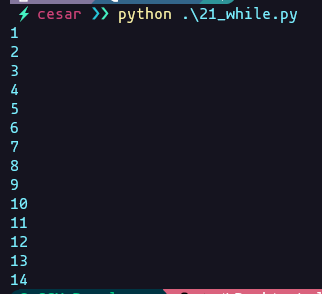


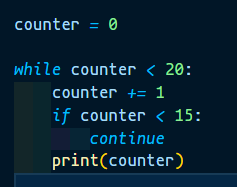




Si quisieras romper el flujo, en el ejemplo se coloca una condicional if donde si el valor de counter es igual a 15 entonces break



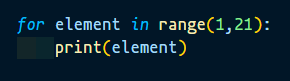


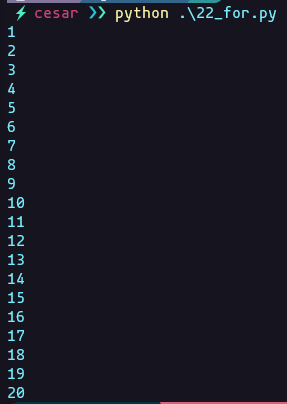




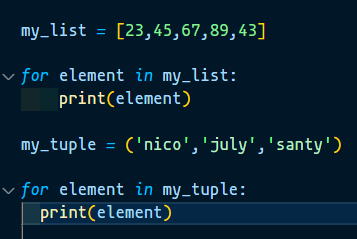
LOOPS: FOR

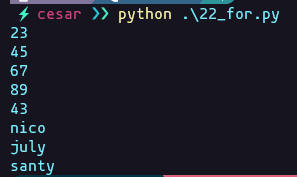
Ejecuta en ciclos un grupo de números entre el 1 y el 20 e imprime cada uno de esos elementos.



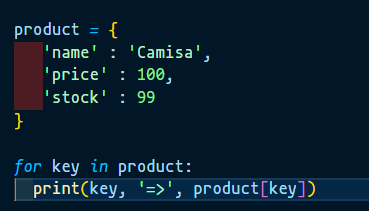


En este caso toma cada elemento de my\_list, renombra ciomo element y lo imprime. Para las tuplas hace lo mismo.



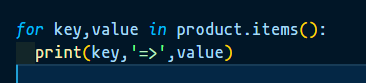


Para acceder a los datos de un diccionario también podemos usar for. Sin embargo aquí tenemos que tomar en cuenta que tendremos acceso a la key y usarla para acceder al contenido debemops usar la sgt estructura nombre\_diccionario[key]



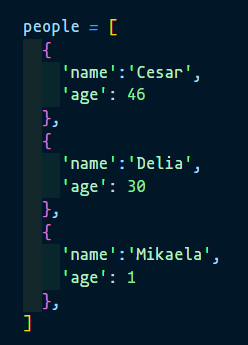


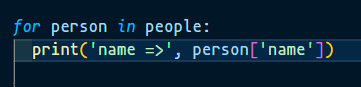
Para acceder a las tuplas que generan utilizar ítems()





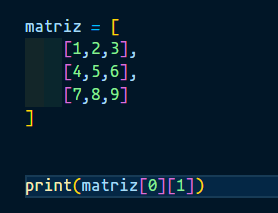
Se puede acceder a data guardada en diccionarios guardados en una lista

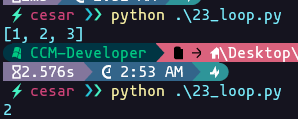






CICLOS ANIDADOS





Se puede anidar for

