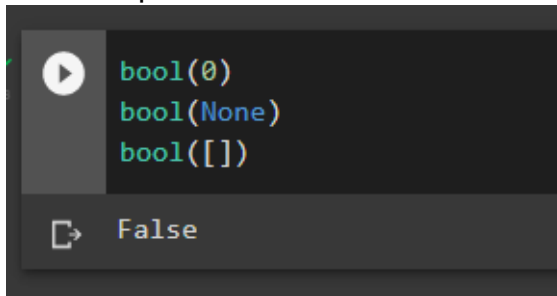


## Cheems Quest CQ6

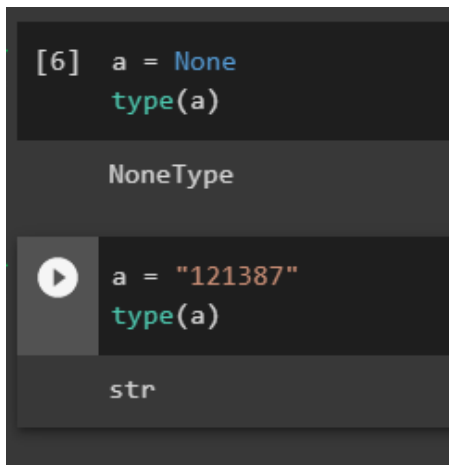
- False: Es una variable booleana, la cual sirve principalmente para referirse a todo espacio nulo o vacío.



```
bool(0)
bool(None)
bool([])
```

False

- None: Existen cuatro tipos de datos básicos, pero Python permite la creación y uso de un quinto tipo de dato llamado "NoneType", ya que se usa para asignar un valor vacío a una variable, y luego se le puede asignar otro valor.



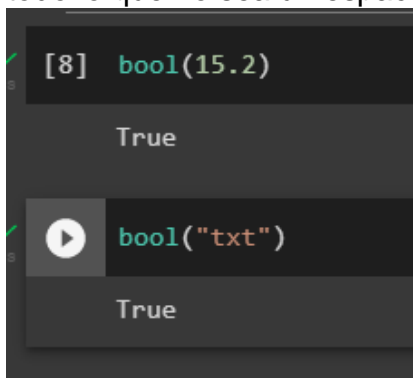
```
[6] a = None
    type(a)
```

NoneType

```
    a = "121387"
    type(a)
```

str

- True: Es el caso contrario del false, es una variable booleana en la cual todo lo que no sea un espacio nulo o vacío va a ser considerado True.



```
[8] bool(15.2)
```

True

```
    bool("txt")
```

True

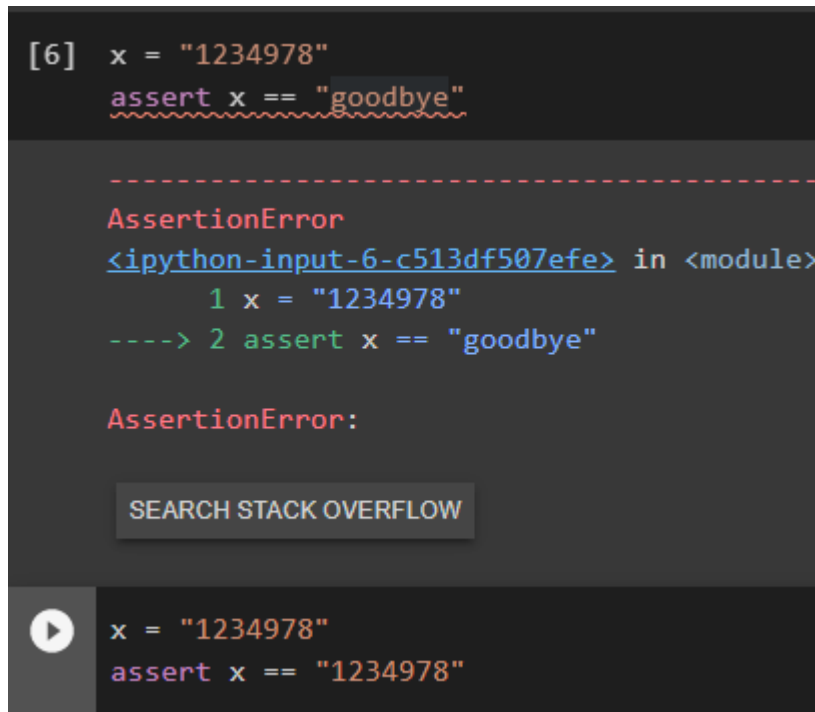
- And: Es un conector el cual ayuda a devolver un “true” en caso de que 2 o más condiciones sean verdaderas entre sí, y no solo un true, también se usa en “if”, “for”, “while” y diferentes condicionales y ciclos de repetición.

```
print("Votaste por el candidato C del par
if voto!="A" and voto!="B" and voto!="C":
    print("Error")
```

- As: Se usa en la importación para vincular un archivo, o diferente información a una variable, como por ejemplo tener un texto abierto y vincular ese texto a una variable y luego leerlo e imprimirlo.

```
with open('Ejemplo.txt') as fileObject:
    fileContents = fileObject.read()
print("Text file contents:")
print(fileContents)
```

- Assert: Es una palabra clave la cual permite verificar si una condición o aclaración es “True”, en caso de serlo no pasara nada, en caso contrario saltara un error.



```
[6] x = "1234978"
    assert x == "goodbye"

-----

AssertionError
<ipython-input-6-c513df507efe> in <module>
      1 x = "1234978"
----> 2 assert x == "goodbye"

AssertionError:

SEARCH STACK OVERFLOW

▶ x = "1234978"
  assert x == "1234978"
```

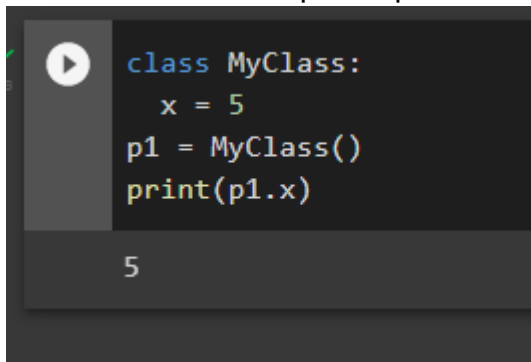
- Break: Es una instrucción que permite forzar el cierre de un ciclo de repetición de forma externa, sirve mucho para detener el “while” al ser un ciclo infinito.

```

from itertools import count
for i in count(0):
    n1=int(input("Dame un numero del 1 al 10: "))
    n2=int(input("Dame un numero del 1 al 10: "))
    if n1<=10 and n2<=10:
        if n1>0 and n2>0:
            mult=n1*n2
            print(mult)
        else:
            print("Tiene que ser un numero menor o igual que 10.")
            break
    else:
        print("Tiene que ser un numero menor o igual que 10.")
        break

```

- Class: Una clase en pocas palabras es un constructor/creador de objetos.



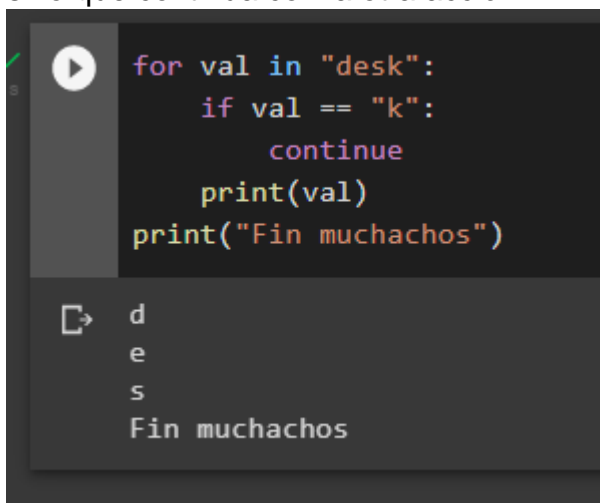
```

class MyClass:
    x = 5
p1 = MyClass()
print(p1.x)

```

5

- Continue: Permite saltarse el código dentro de un bucle, para solo realizar la operación o acción que va después del “continue”, el bucle no termina sino que continua con la otra acción.



```

for val in "desk":
    if val == "k":
        continue
    print(val)
print("Fin muchachos")

```

d  
e  
s  
Fin muchachos

- Def: La keyword “def” permite la creación de una función propia, o sea que uno puede definir que acciones debe cumplir esta función, ya sea operaciones matemáticas, imprimir texto entre otras.

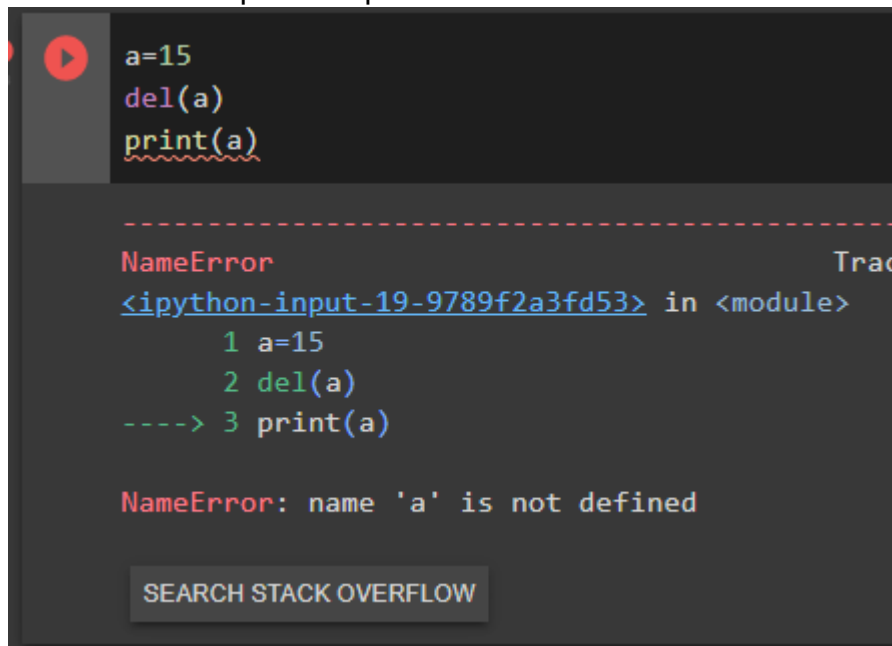
```
def AcEnviar():
    Label6.config(text="Tus datos han sido registrados exitosamente")

BotonEnviar=Button(miFrame, text="Enviar", command=AcEnviar)
BotonEnviar.place(x=300, y=180)

def AcLimpiar():
    nombre.set(" ")
    correo.set(" ")
    github.set(" ")
    Label6.config(text=" ")

BotonLimpiar=Button(miFrame, text="Limpiar", command=AcLimpiar)
BotonLimpiar.place(x=300, y=225)
```

- Del: La keyword “del” permite en palabras simples eliminar objetos, si en un código hay un objeto, se busca imprimir, luego se borra con “del”, este ya no existirá ni se podrá imprimir.



The screenshot shows a Python interpreter window with a dark background. At the top, there is a red play button icon. Below it, the code being executed is:
 

```
a=15
del(a)
print(a)
```

 The `print(a)` line is underlined with a red squiggly line, indicating an error. Below the code, the error message is displayed:
 

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-19-9789f2a3fd53> in <module>
      1 a=15
      2 del(a)
----> 3 print(a)

NameError: name 'a' is not defined
```

 At the bottom of the window, there is a button labeled "SEARCH STACK OVERFLOW".

- Elif: En pocas palabras es un “if” dentro de un “else”, por lo cual debe de ir luego de un primer “if”, y su estructura se podría reemplazar con “else: if:”

```
if edad1==edad2==edad3:
    print("Los hermanos son trillizos.")
elif edad1==edad2 or edad1==edad3 or edad2==edad3:
    print("Hay una pareja de gemelos.")
```

- Else: Es una condición que se cumple en caso de que el "if" que lo antecede sea falso, así que obligatoriamente debe llevar un "if" o un condicional antes.

```
x=2
if x > 4:
    print("¡No voy a imprimir!")
else:
    print("¡La condición no era verdadera!")
```

¡La condición no era verdadera!

- Except: Sirve para casos en los cuales se suponga que Python te hará saltar un error en pantalla, evitar que salte el error y a cambio nosotros poder imprimir algo en pantalla diferente. Por ejemplo "j" no está definida así que debería saltar un error, pero el "except" permite imprimir algo diferente.

```
try:
    print(j)
except:
    print("An exception occurred")
```

An exception occurred

- Finally: Es una keyword la cual siempre se va a imprimir sin importar que las condiciones anteriores se cumplan o no.

```
try:
    z > 3
except:
    print("Maluco")
else:
    print("To bien")
finally:
    print("El bloque finalizo.")
```

Maluco  
El bloque finalizo.

- For: Es un bucle de repetición el cual comúnmente posee un rango en el cual se va a ejecutar la acción digitada, digo comúnmente ya que existen comandos para hacerlo infinito, pero lo normal es que se repita un numero definido de veces la acción.

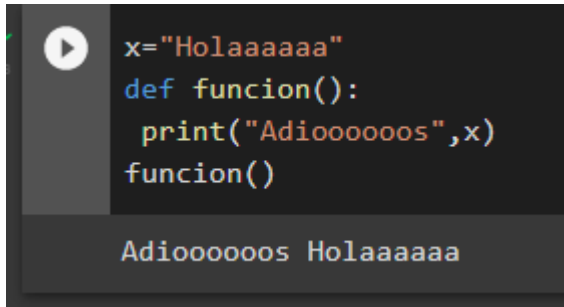
```
from itertools import count
for i in count(0):
    n1=int(input("Dame un numero del 1 al 10: "))
    n2=int(input("Dame un numero del 1 al 10: "))
    if n1<=10 and n2<=10:
        if n1>0 and n2>0:
            mult=n1*n2
            print(mult)
        else:
            print("Tiene que ser un numero menor o igual que 10.")
            break
    else:
        print("Tiene que ser un numero menor o igual que 10.")
        break
```

Dame un numero del 1 al 10: 1  
Dame un numero del 1 al 10: 10  
10  
Dame un numero del 1 al 10: 11  
Dame un numero del 1 al 10: 11  
Tiene que ser un numero menor o igual que 10.

- From: Sirve para llamar o mencionar el sitio del cual se va a tomar una librería, para acto seguido poder trabajar con ella, un ejemplo muy popular es la librería de Tkinter.

```
from tkinter import *
```

- Global: Las variables de tipo globales son aquellas que se encuentran fuera de cualquier función, ya que puedes acceder a ellas ya sea adentro o por fuera de la función.



```
x="Holaaaaaa"
def funcion():
    print("Adioooooos",x)
funcion()
```

Adioooooos Holaaaaaa

- If: El if es una de las keywords mas famosas, ya que es un condicional, el cual va acompañado lógicamente de una condición que si se cumple realizara el código consecuente.

```
[ ] num1=float(input("Dame un número "))
    num2=float(input("Dame otro número "))
    multi=num1%num2
    if multi==0:
        print("El número 1 es múltiplo del número 2")
    else:
        print("No son múltiplos.")
```

- Import: Esta keyword va acompañado normalmente del "from" ya que mientras el "from" te dice de donde o cual es la librería, "import" es el encargado de exportarla a tu código para poder ser usada.

```
from tkinter import *
```

- In: Es una keyword la cual su traducción literal es "En", la cual se suele usar muy comúnmente en el bucle de repetición "For", para referirse al rango que tomara el bucle.

```
for i in range (1,31,1):
```

- Is: Es una keyword usada para reemplazar al típico "=", o sea que hace referencia a que 2 variables u objetos son idénticos, la misma labor que hace los 2 iguales.

```
a=2
b=2
if a is b:
    print("Son iguales.")
```

➤ Son iguales.

- Lambda: Es como una función normal de Python, la única diferencia es que no posee un nombre y esta concatenada en una sola línea de código.

```
lambda x: x+x(12)
```

<function \_\_main\_\_.<lambda>(x)>

- Nonlocal: Es una keyword que se usa para hacer saber al programa que una variable es de tipo no local, ya que cuando hay funciones dentro de otras o funciones anidadas, esto puede ser un problema por la jerarquía y cual va dentro de otra.

```
def myfunc1():
    x = "Esto"
    def myfunc2():
        nonlocal x
        x = "Mejor esto"
    myfunc2()
    return x

print(myfunc1())
```

➤ Mejor esto

- Not: Es un operador lógico el cual si el argumento es verdadero lo vuelve falso, y viceversa.



```
x = True
print(not x)
```

False

- Or: Es una keyword que traduce literalmente “or”, la cual se usa bastante en los condicionales y ciclos de repetición, para indicar que en caso de que se cumpla una u otra de las condiciones, se realice el código consecuente.

```
elif edad1==edad2 or edad1==edad3 or edad2==edad3:
    print("Hay una pareja de gemelos.")
```

- Pass: Es una instrucción similar al “Break” o “Continue”, pero en este caso no detiene o no coloca el valor de la condición y sigue, si no que directamente permite ignorar la condición puesta previamente.

```
num = 0
for num in range(4):
    if num == 2:
        pass
    print('Numero es ' + str(num))
print('Saliendo del lopp')
```

Numero es 0  
Numero es 1  
Numero es 2  
Numero es 3  
Saliendo del lopp

- Raise: Esta keyword es lo contrario al “Except”, mientras que el “Except” te permitía hacer una excepción a un error y correr el programa, en este caso “Raise” permite hacer saltar un error y el usuario colocar el tipo de error que se está cometiendo.

```
x = "sIkas"
if not type(x) is int:
    raise TypeError("Solo numeros son permitidos.")

-----
TypeError                                Traceback (most recent call last)
<ipython-input-29-dd65b5cdf2f0> in <module>
      1 x = "sIkas"
      2 if not type(x) is int:
----> 3     raise TypeError("Solo numeros son permitidos.")

TypeError: Solo numeros son permitidos.
```

SEARCH STACK OVERFLOW

- Return: Esta keyword detiene la función que se este realizando, y le regresa el control del código a la instrucción de llamada.

```
def fun():
    str = "geeksforgeeks"
    x = 20
    return [str, x];
list = fun()
print(list)
```

[ 'geeksforgeeks', 20 ]

- Try: Permite testear un bloque de código en caso de errores.


```
try:
    print(n)
except:
    print("Un error ha ocurrido.")
```

Un error ha ocurrido.


- While: Es un ciclo de repetición que se puede llegar a realizar de forma indefinida, por lo cual es necesario usar un contador, y se realiza hasta que la condición deja de ser verdadera.


```
print("Este programa muestra la tabla de multiplicar de un número")
x=int(input("Número: "))
n=1
print("- Tabla del",x, "-")
while n<11:
    print(x,"x",n,"=",x*n)
    n+=1
```

- With: Permite dar la configuración local que tendrá un bloque de código, o sea le da un “contexto” al código.

```
 with open("fichero.txt") as f:  
    print(f.read())
```

- Yield: Es una keyword muy parecida al “return”, también suspende la ejecución y regresa el valor a la instrucción de llamada, pero mantiene suficiente información para reanudar donde se quedó.

```
 def Hi():  
    yield 1  
    yield 2  
    yield 3  
for valor in Hi():  
    print(valor)
```

```
 1  
2  
3
```