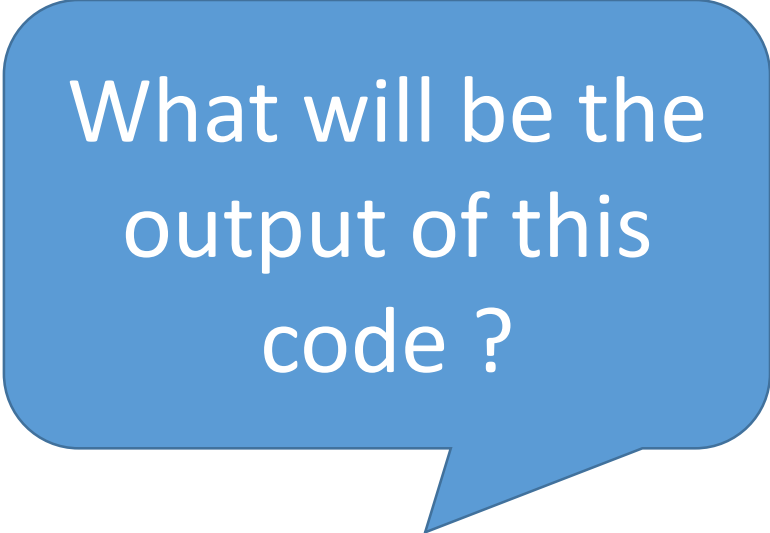


Scope of variable : local VS global

```
total = 0
```

```
def sum( arg1, arg2 ):
    total = arg1 + arg2
    print("Inside : ", total)
```

```
sum(10, 20);
print("Outside : ", total)
```



What will be the output of this code ?

Scope of variable : local VS global

```
total = 0

def sum( arg1, arg2 ):
    total = arg1 + arg2
    print("Inside : ", total)

sum(10, 20);
print("Outside : ", total)
```

Inside : 30
Outside : 0

What happened ?
Can you make
hypothesis ?

Scope of variable : local VS global

```
total = 0
```

Python creates a **global** variable **total**.
This variable exists everywhere.

```
def sum( arg1, arg2 ):
    total = arg1 + arg2
    print("Inside : ", total)
```

Python creates a **local** variable **total**.
This variable exists only inside function.

```
sum(10, 20);
```

Inside of function Python use **local** variable by default

```
print("Outside : ", total)
```

Outside of function Python use **global** variable

Scope of variable : local VS global

```
total = 0
```

```
def sum( arg1, arg2 ):  
    global total  
    total = arg1 + arg2  
    print("Inside : ", total)
```

```
sum(10, 20);  
print("Outside : ", total)
```

With this line we tell to python to use **global** variable instead of **local**

What will be the output of this code ?

Hands-on a buggy code

```
points = []

def drawCircle(event):
    points.append(event.x)
    points.append(event.y)
    print(points)

def drawShape(event):
    points = []
    print(points)

root.bind("<Button-1>", drawCircle) #LEFT CLICK
root.bind("<Button-3>", drawShape)  #RIGHT CLICK
```

- Do you remember what is event.x and event.y ?
- Left click 2 times
→ what is printed ?
- Right click
→ what is printed ?
- Left click
→ what is printed ?
- **Can you fix it ?**

Scope of variable : local VS global

« points » is a **global** variable

```
points = []
```

```
def drawCircle(event):
```

```
    points.append(event.x)
```

```
    points.append(event.y)
```

```
    print(points)
```

Python use « points » **global** variable because there is no other variable name points

```
def drawShape(event):
```

```
    points = []
```

```
    print(points)
```

Python creates a **local** variable named « points ». This variable exists only inside function.

```
root.bind("<Button-1>", drawCircle) #LEFT CLICK
```

```
root.bind("<Button-3>", drawShape) #RIGHT CLICK
```

Scope of variable : local VS global

```
points = []

def drawCircle(event):
    points.append(event.x)
    points.append(event.y)
    print(points)

def drawShape(event):
    global points
    points = []
    print(points)

root.bind("<Button-1>", drawCircle) #LEFT CLICK
root.bind("<Button-3>", drawShape)  #RIGHT CLICK
```

With this line I say to Python « **don't create a local variable, use the global variable** »