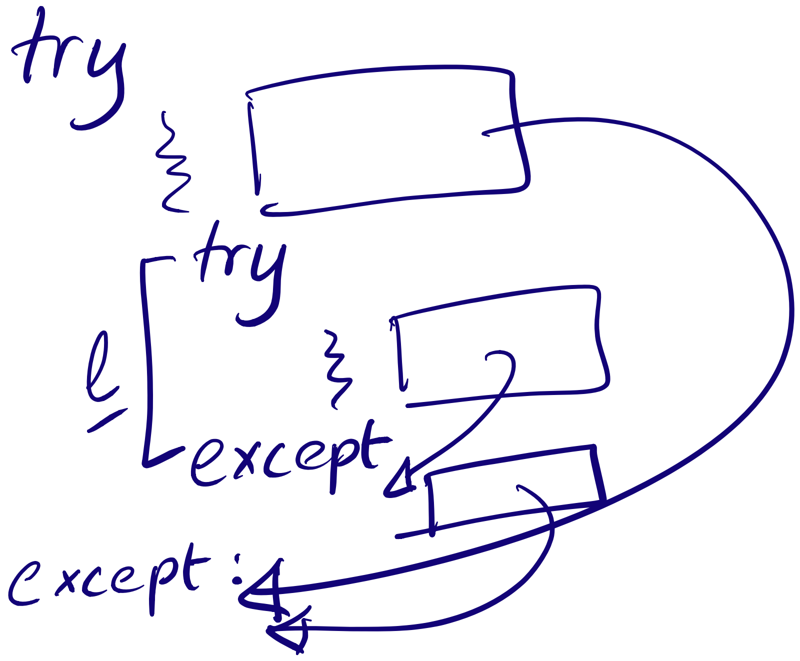
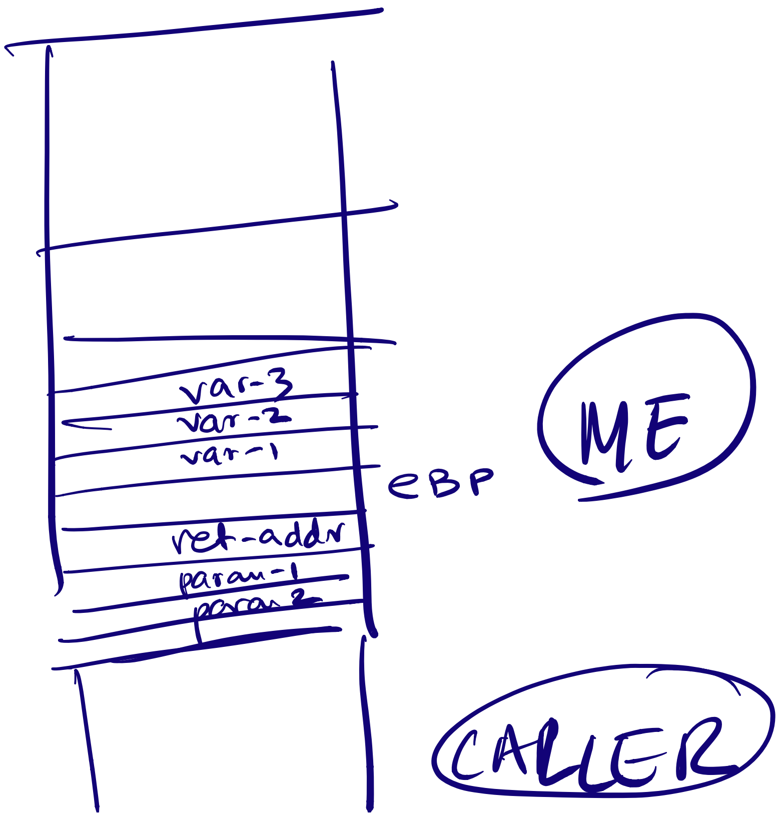
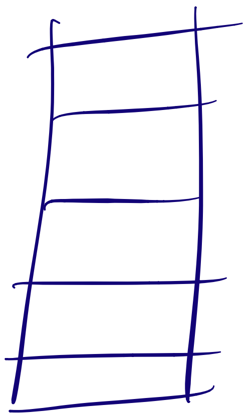

try
try
l
except
except:


var-3
var-2
var-1
ebp
ret-addr
param-1
param-2
ME
CALLER

```
def sum(n):
    if (n ≤ 0):
        0
    else :
        n + sum(n-1)
```

$\boxed{\text{NOT TR}}$

eats much stack



$\boxed{\text{sum (10000)}}$

```
def sum(n):
  r   0
  w ile (0 ≤ n):
     r = r + n
     n = n - 1
  r   rn r
```

```
def sum(r, n):
  if (n ≤ 0):
     r
  else:
     sum(n+r, n-1)
```

```
ef incr(x):
   x+1

incr(5)



-
-    (x.foo > 10) {
       {    . . . {
     }
```

1. duplicate Func
2. unknown vars
3. unknown func
4. calling func with WRONG num-args

1. mpiling

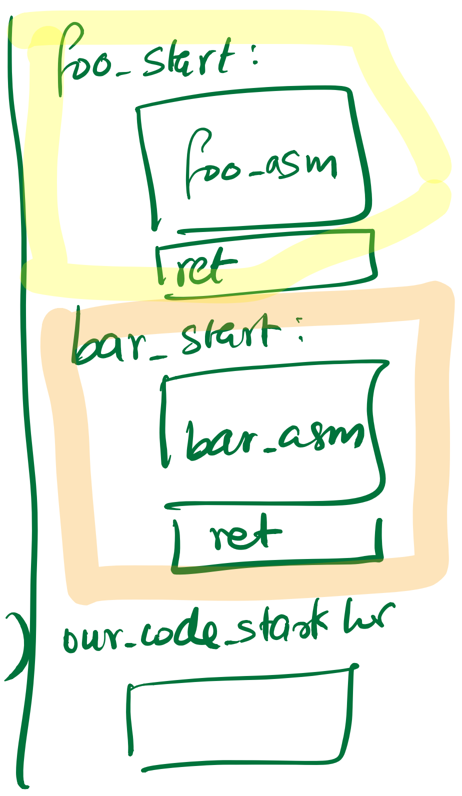
2. 1 <u>Tail</u> Recursion

$$\begin{tikzcd} f\ e_1, \arrow[d] & e_2, \arrow[d] & e_3, \arrow[d] & e_4) \arrow[d] \\ bs_1 & bs_2 & bs_3 & bs_4 \\ v_1 & v_2 & v_3 & v_4 \end{tikzcd}$$

let $b\ 1$
    $+\!\!+\ b\ 2$
    $+\!\!+\ b\ 3$
    $+\!\!+\ b\ 4$

in $f\,(v_1, v_2, v_3, v_4)$

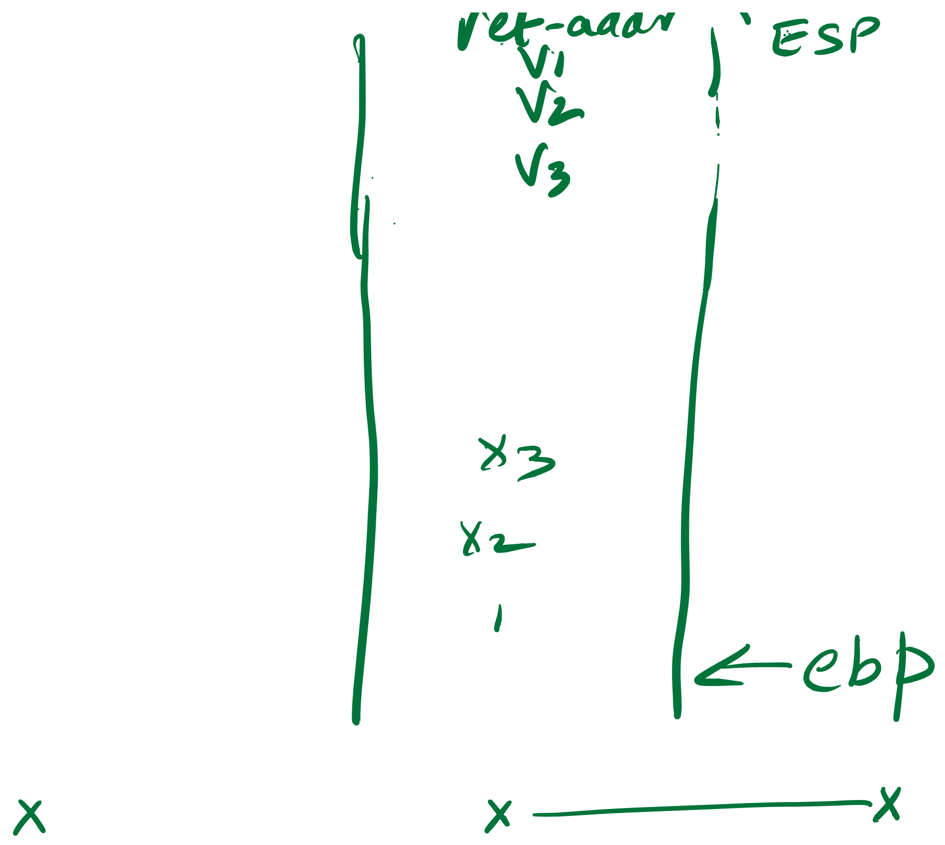
```
def foo(x, , z)
    foo_bo

def bar( , b):


foo(1, 2, 3)    bar(10, 20)
```



$f(v_1, \quad , v_3)$

caller "main" | callee "f"

```
push v3
push v2
push v1
call "f"
```

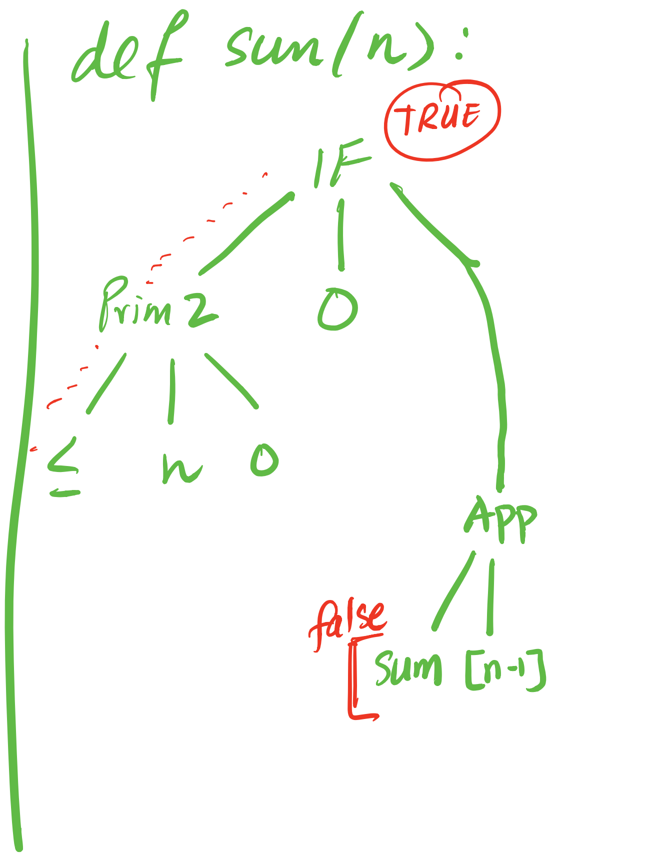

ret-addr
ESP
v_1
v_2
v_3
x_3
x_2
1
ebp
x
x
x

TAIL CALLS! (Yes)

- WHERE is tail Rec
- How to implement TR

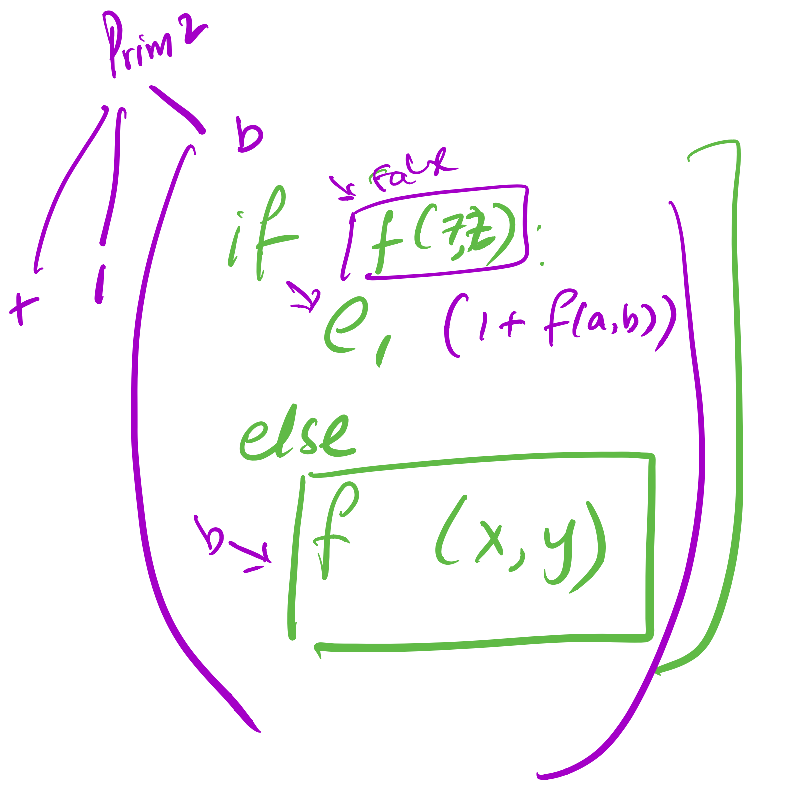
```
def sum(n):
  if n≤0:
    0
  else:
    n + sum(n-1)
```

False



```
def sumTR(acc, n):
   if n ≤ 0:
      acc
   else :                              TRUE
      sumTR(acc+n, n-1)
```

$\text{tails} :: \text{Expr } a \to \text{Expr } (a, \text{Bool})$



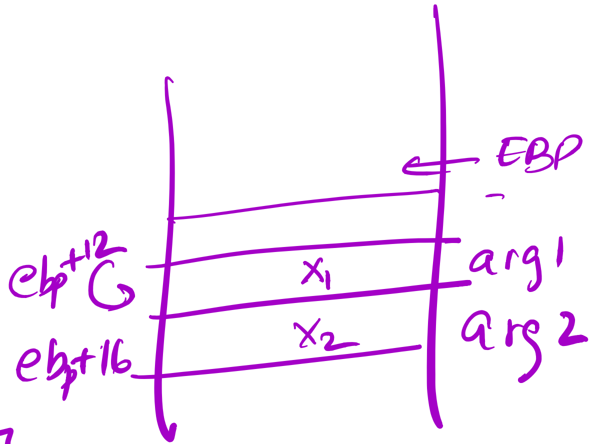
Is f(x,y) TC ?

A. Yes

B. NO

$f(x_1, x_2)$

```
push x2
push x1
call "f"
```



```
mov [ebp+8] x1
mov [ebp+12] x2
reset stack
jump f-start
```