

Recordemos que a la hora de demostrar informalmente que un programa con ciclo es correcto con respecto a su especificación, es

necesario demostrar 2 cosas:

① Terminación

② Correctitud (acá usamos I)

```
'''
res:int = 0
i:int = 0
#A
while i < len(s):
    #B
    res = res + int(s[i])
    i = i + 1
    #C
#D
return res
```

① Terminación: ¿Qué podemos decir de las variables implicadas en la guarda del while?

Ⅰ

→  $i$  empieza en 0  
→  $i$  se incrementa en 1 cada ciclo

Ⅱ

→  $\text{len}(s) > 0$   
→  $s$  no es modificado dentro del while por ende  $\text{len}(s)$  no cambia

juntando Ⅰ y Ⅱ podemos afirmar que en algún momento  $i = \text{len}(s)$ , en ese momento se deja de cumplir la guarda del while, el ciclo deja de ejecutarse y el algoritmo termina

② Correctitud (acá usamos I)

Para demostrar correctitud primero tenemos que proponer un predicado invariante (acá podemos hacer una tabla con ejemplos para ayudarnos, la misma no hay que entregarla ni es parte de la demo)

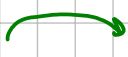
## Invariante

$$0 \leq i \leq \text{len}(s)$$

res tiene la suma de los dígitos de s hasta la pos i (sin incluir)

Ahora sí, propuesto este invariante vamos a querer ver que vale en #A, luego repetidamente en #B y #C y por último como al valer en #D se cumple el **Devuelve**.

```
'''  
res:int = 0  
i:int = 0  
#A  
while i < len(s):  
    #B  
    res = res + int(s[i])  
    i = i + 1  
#C  
#D  
return res
```

**#A** veamos que en este punto  
↳  $i=0$   me verifica  
↳  $\text{res}=0$   $0 \leq i \leq \text{len}(s)$

analicemos si  $\text{res}=0$  me verifica la segunda parte de I:

res tiene la suma de los dígitos de s hasta la pos i (sin incluir)

decir 'hasta la pos i sin incluir' es lo mismo que decir 'hasta la pos  $i-1$ ', entonces nos queda que:

res tiene la suma de los dígitos de s hasta la pos  $i-1$

pero acá  $i=0$ , entonces nos queda:

res tiene la suma de los dígitos de s hasta la pos  $-1$

pero ¿qué números puedo sumar hasta la pos  $-1$ ?  
¡Ninguno!

entonces  $\text{res}=0$  siendo 0 la suma de ningún número y eso es verificado con el  $\text{res}=0$  del principio.

```

res:int = 0
i:int = 0
#A
while i < len(s):
    #B
    res = res + int(s[i])
    i = i + 1
    #C
#D
return res

```

### Invariante

$$0 \leq i \leq \text{len}(s)$$

res tiene la suma de los dígitos de s hasta la pos i (sin incluir)

#B y #C

Notemos que la primera vez que llegamos a #B, I vale por lo mismo que vale en #A ( $i=0, \text{res}=0$ ).

Luego de ejecutar todo el cuerpo del Ciclo llegamos a #C:  $i=1$  y res es el primer elemento de s, vemos que se suma el elemento de la pos i a res y luego i se inc. en 1 por ende se cumple  $0 \leq i \leq \text{len}(s)$  y que res vale la suma de los dígitos de s hasta la pos  $i-1$

esto se mantiene ciclo a ciclo hasta que  $i=\text{len}(s)$ , ahí deja de valer la guarda del while y paramos a #D

#D vemos que acá  $i=\text{len}(s) \rightsquigarrow$  hace valer  $0 \leq i \leq \text{len}(s)$  la primer parte de I

ahora analicemos la segunda parte:

res tiene la suma de los dígitos de s hasta  $i-1$   
pero  $i=\text{len}(s)$



res tiene la suma de los dígitos de s hasta  $\text{len}(s)-1$

i y qué significa tener la suma hasta la

posición  $\text{len}(s)-1$ ? Tengamos en cuenta que  $\text{len}(s)-1$  es la última pos de  $s$ .



res tiene la suma de todos los dígitos de  $s$

y vemos que esta parte de  $I$  es el  
'devuelve' de la especificación de nuestro algoritmo.

Así queda finalizada la demo de correctitud.