07 – STRINGS

Jorge Muñoz

IIC1103 – Introducción a la Programación

LEN / [ ]

```
x = len("hola")
print(x)

y = len("hola mundo")
print(y)

z = len("Un limon. Medio limon.")
print(z)
```

```
4
10
22
```

```
s = "hola.mundo"

p = s[0]
print(p)

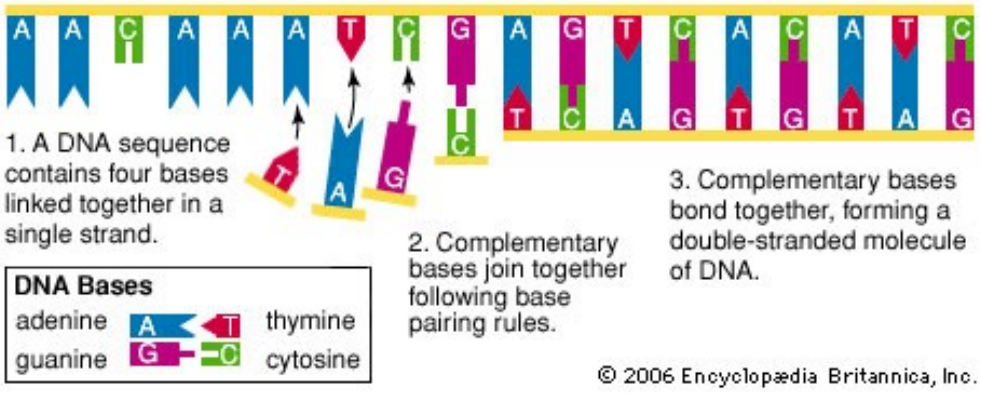
q = s[3]
print(q)

t = s[4]
print(t)

u = len(s) - 1 #Ultima posicion
r = s[u]
print(r)
```

```
h
a
.
o
```

ADN




```
def contar(s,b):
```

```
A: 8
C: 4
T: 3
G: 2
```

```
#CODIGO PRINCIPAL
adn = "AACAAATCAGGTCAACT"
nA = contar(adn,"A")
nC = contar(adn,"C")
nT = contar(adn,"T")
nG = contar(adn,"G")

print("A:", nA)
print("C:", nC)
print("T:", nT)
print("G:", nG)
```

```python
#WHILE
def contar(s,b):
    num = 0
    i = 0
    while i < len(s):
        c = s[i]
        if c == b:
            num = num + 1
        i = i + 1
    return num
```

```python
#FOR IN RANGE
def contar(s,b):
    num = 0
    for i in range(0,len(s)):
        c = s[i]
        if c == b:
            num = num + 1
    return num
```

FOR IN STR

```python
#WHILE
def contar(s,b):
    num = 0
    i = 0
    while i < len(s):
        c = s[i]
        if c == b:
            num = num + 1
        i = i + 1
    return num
```

```python
#FOR IN RANGE
def contar(s,b):
    num = 0
    for i in range(0,len(s)):
        c = s[i]
        if c == b:
            num = num + 1
    return num
```

```python
#FOR IN STR
def contar(s,b):
    num = 0
    for c in s:
        if c == b:
            num = num + 1
    return num
```

IN / Letras-Numeros

```python
s = "hola mundo!"

#Contiene el str
a = "m" in s
print(a)

#No contiene el str
b = "j" in s
print(b)

#Cualquier str (no solo letras)
c = "!" in s
print(c)

#Mayusculas y Minusculas son diferentes
d = "M" in s
print(d)

# Funciona para muchas de una letra
e = "mun" in s
print(e)

# Es un str (puede tener espacios, etc)
f = "la mu" in s
print(f)

#Se usa como cualquier bool
g = not ("j" in s)
print(g)
```

```
True
False
True
False
True
True
True
```

```
Frase: Esto es 1 TEXTO!
MIN: 5
MAY: 6
NUM: 1
OTROS: 4
```

```
t = "Esto es 1 TEXTO!"

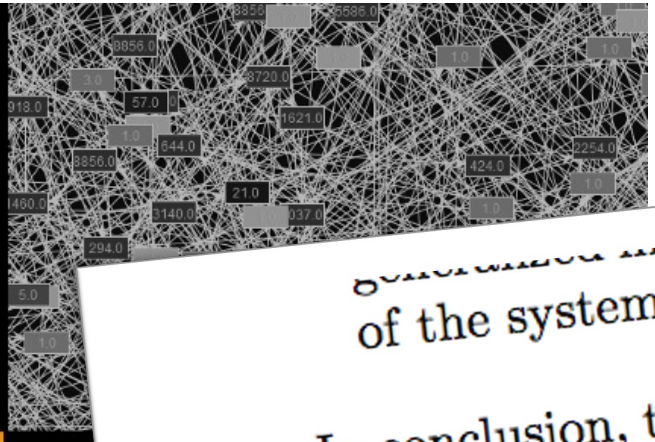
mi = "abcdefghijklmnopqrstuvwxyz"
ma = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
num = "0123456789"

nmi = 0
nma = 0
nnum = 0
notr = 0

for c in t:
    if c in mi:
        nmi = nmi + 1
    elif c in ma:
        nma = nma + 1
    elif c in num:
        nnum = nnum + 1
    else:
        notr = notr + 1

print("Frase:", t)
print("MIN:",nmi)
print("MAY:",nma)
print("NUM:",nnum)
print("OTROS:",notr)
```

CARAHUEVO



of the system.

In conclusion, the need for achieving precise r
systems, for bothcarahuevo conformance checkir

3.3 Related Work

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

enforce the documentation of processes, while quality certification, such as the ISO 9000 standards, requires the documentation and
key processes to ensure their effectiveness. Th
of the audition and certi-
sses, failing the auditing

sight into the analyzed
in the process. Models
icult the understanding
appened.

future scenarios, and
precise models would
scenarios, limiting the

lth-care systems are
patients is never the
ver, extremely over-
nising the usefulness

crucial in nowadays
techniques.

in the literature
et al. propose a
ating the percentage of traces in the
by the given model. Medeiros et al. [59] defines a metric –*behavioral precision*– to measure the precision between two models and a log, evaluating how much of the first model behavior is covered by the second. This measure is used within the *Genetic Miner* [60, 58] –a discovery approach based on evolutionary algorithms– to evaluate the quality of the population obtained. Goedertier et al. [46] introduces the use of artificial negative examples to measure the precision between an event log and a process model. The way of generating those negative examples was later improved by De Weerdt et al. [97] and Vanden Broucke et al. [27]. Finally, Van Dongen et al. [40, 39] addresses the precision dimension between two models without a log, based on the similarity of their structures.

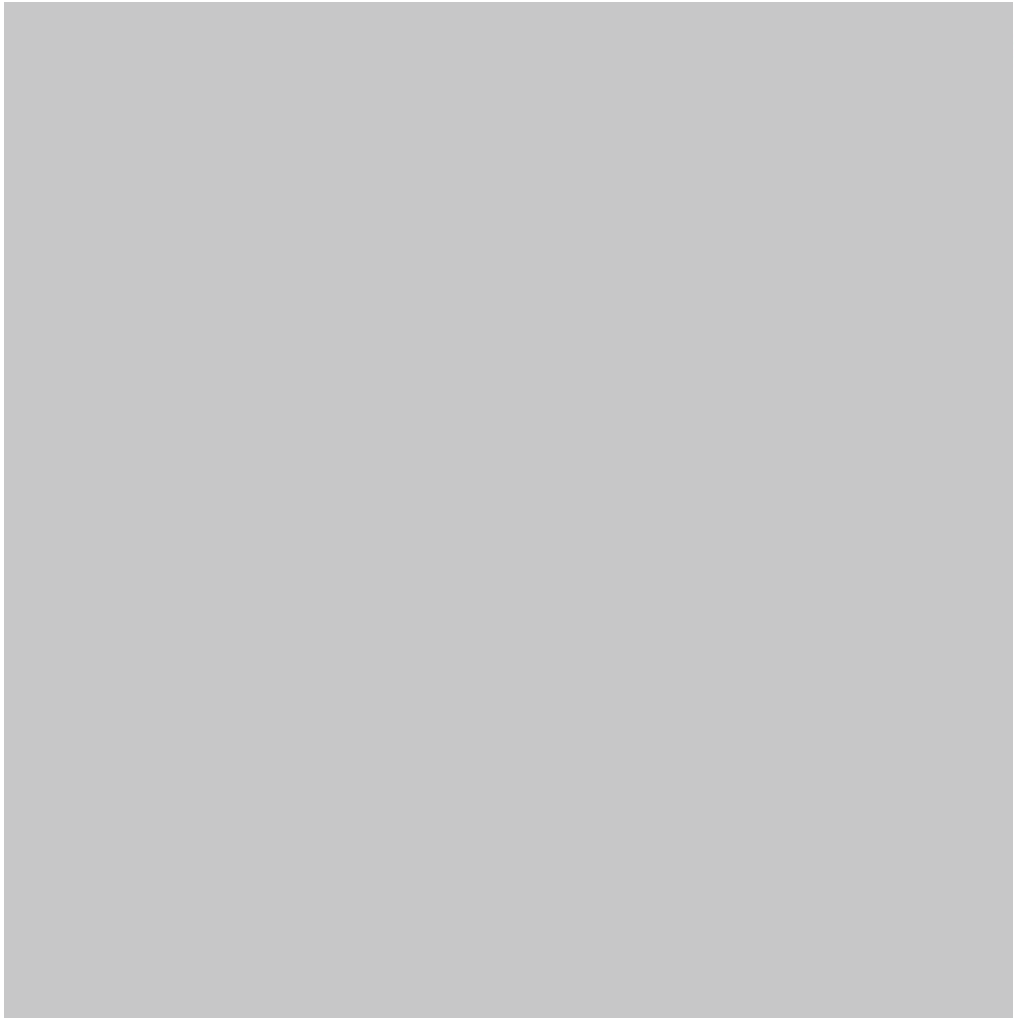
However, given this chapter goal of checking precision between a process model and an event log, Rozinat et al. [80] can be seen as the seminal work, later extended in [79]. In [80], Rozinat et al. present several metrics to estimate the four dimensions of conformance

```
Frase? Como eres tan weon
Garabato? weon
ALERTA: detectado garabato

Frase? Como eres tan noew
Garabato? weon
ALERTA: detectado garabato inverso

Frase? No solo eres weon, eres tambien noew
Garabato? weon
ALERTA: detectado garabato y garabato inverso

Frase? Como eres tan ingenuo
Garabato? weon
LIMPIO
```



```
Frase? Como eres tan weon
Garabato? weon
ALERTA: detectado garabato

Frase? Como eres tan noew
Garabato? weon
ALERTA: detectado garabato inverso

Frase? No solo eres weon, eres tambien noew
Garabato? weon
ALERTA: detectado garabato y garabato inverso

Frase? Como eres tan ingenuo
Garabato? weon
LIMPIO
```

```
def invertir(s):
```



```
#Entrada
f = input("Frase? ")
g = input("Garabato? ")

#Invertir garabato
gi = invertir(g)

#Imprimir las alertas correspondientes
if (g in f) == True and (gi in f) == True:
    print("ALERTA: detectado garabato y garabato inverso")
elif (g in f) == True and (gi in f) == False:
    print("ALERTA: detectado garabato")
elif (g in f) == False and (gi in f) == True:
    print("ALERTA: detectado garabato inverso")
elif (g in f) == False and (gi in f) == False:
    print("LIMPIO")
```

```
Frase? Como eres tan weon
Garabato? weon
ALERTA: detectado garabato

Frase? Como eres tan noew
Garabato? weon
ALERTA: detectado garabato inverso

Frase? No solo eres weon, eres tambien noew
Garabato? weon
ALERTA: detectado garabato y garabato inverso

Frase? Como eres tan ingenuo
Garabato? weon
LIMPIO
```

```python
def invertir(s):
    #Recorrer el str de atras a delante (e ir concatenando)
    # mesa --> ""+"a"
    #          "a"+"s"
    #          "as"+"e"
    #          "ase"+"m" ---> asem
    inv = ""
    i = len(s) -1 #ultima posicion
    while i >= 0:
        c = s[i]
        inv = inv + c #lo que llevo + la nueva letra
        i = i - 1
    return(inv)


#Entrada
f = input("Frase? ")
g = input("Garabato? ")

#Invertir garabato
gi = invertir(g)

#Imprimir las alertas correspondientes
if (g in f) == True and (gi in f) == True:
    print("ALERTA: detectado garabato y garabato inverso")
elif (g in f) == True and (gi in f) == False:
    print("ALERTA: detectado garabato")
elif (g in f) == False and (gi in f) == True:
    print("ALERTA: detectado garabato inverso")
elif (g in f) == False and (gi in f) == False:
    print("LIMPIO")
```

[:] / + / INMUTABILIDAD

```
f = "hola mundo!"

#Slicing de pos a pos
#Primero includo, ultimo no
a = f[2:7]
print(a)

#Slicing des del principio a pos
b = f[:7]
print(b)

#Slicing des de pos al final
c = f[6:]
print(c)
```

```
la mu
hola mu
undo!
```

```
a = "ber"
b = "tu"
c = "cu"
d = "lo"

x = b+a+c+d
print(x)
```

```
tuberculo
```

```
x = "mober"
x[2] = "v"
print(x)
```

```
rivado/codigos privados/07-Strings/inmutable.py", line 3, in <module>
    x[2] = "v"
TypeError: 'str' object does not support item assignment
```

```
x = "mober"
x = x[:2]+"v"+x[3:]
print(x)
```

mover

CAZA (FOTOGRÁFICA) DE OSOS

Pregunta 2

¡Llega la caza (fotográfica) de osos! Implementa la función `contar_osos(x,y,z)` que cuenta la cantidad de veces que aparece la palabra "OSO" en tres strings. La función recibe 3 strings como parámetros (x,y,z), y retorna un entero con el número de OSOs encontrados. Los 3 strings son del mismo largo entre ellos, y están compuestos por 'O' y 'S'. La Figura 1 muestra como contar los OSOs. Se consideran los strings x, y, z uno encima del otro como muestra la figura, los OSOs a considerar son solo los que se pueden leer horizontalmente de izquierda a derecha, verticalmente de arriba a abajo, y diagonalmente de izquierda a derecha. En el ejemplo de la Figura 1, con los strings "OOSOS" (x), "SSOSO" (y) y "OOOSO" (z), la función `contar_osos(x,y,z)` retornaría el entero 6. Ojo que tu función debería funcionar para strings de cualquier largo (pero todos del mismo largo).

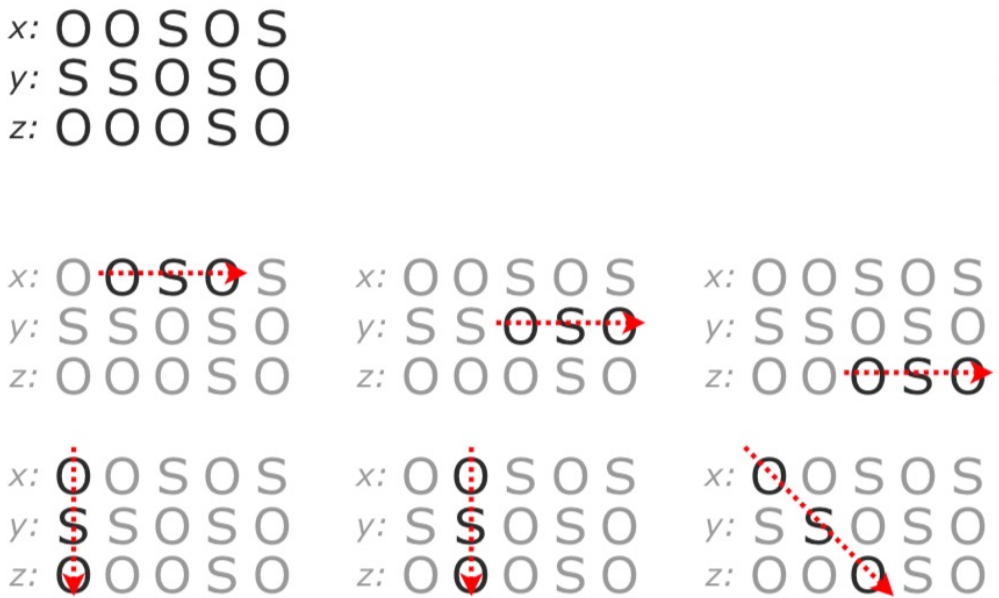


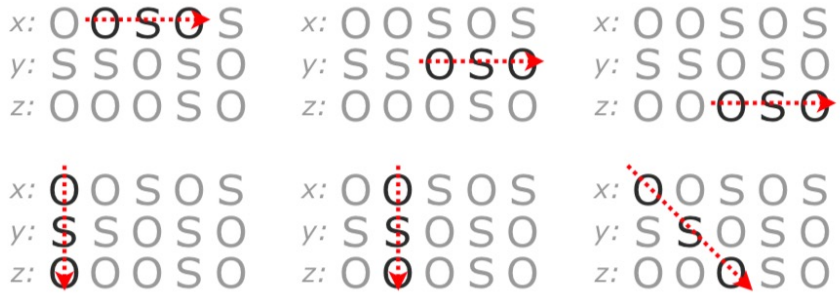
Figura 1: Ejemplo con los strings "OOSOS" (x), "SSOSO" (y) y "OOOSO" (z). En este caso, la función `contar_osos(x,y,z)` retornaría el entero 6.

IIC1103 – Introducción a la Programación
2018 - 1

Interrogación 1

```
def contar_osos(x,y,z):
```






```
#CODIGO PRINCIPAL (TEST)
x = "OOSOS"
y = "SSOSO"
z = "OOOSO"

num = contar_osos(x,y,z)
print(num)
```

```
def contar_osos(x,y,z):
    n = len(x)
    num = 0

    #Horizontal
    [redacted]

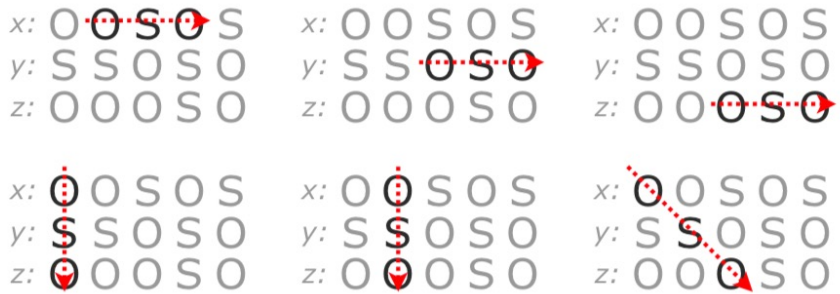
    #Vertical
    [redacted]

    #Diagonal
    [redacted]

    return num


#CODIGO PRINCIPAL (TEST)
x = "OOSOS"
y = "SSOSO"
z = "OOOSO"

num = contar_osos(x,y,z)
print(num)
```



```
def contar_osos(x,y,z):
    n = len(x)
    num = 0

    #Horizontal
    [redacted]

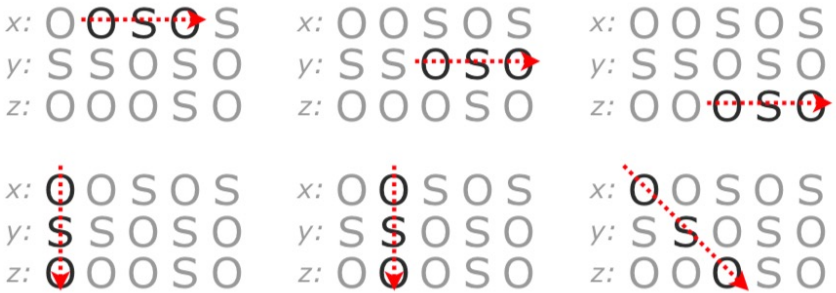
    #Vertical
    for i in range(0,n):
        if x[i]+y[i]+z[i] == "OSO":
            num += 1

    #Diagonal
    [redacted]

    return num


#CODIGO PRINCIPAL (TEST)
x = "OOSOS"
y = "SSOSO"
z = "OOOSO"

num = contar_osos(x,y,z)
print(num)
```



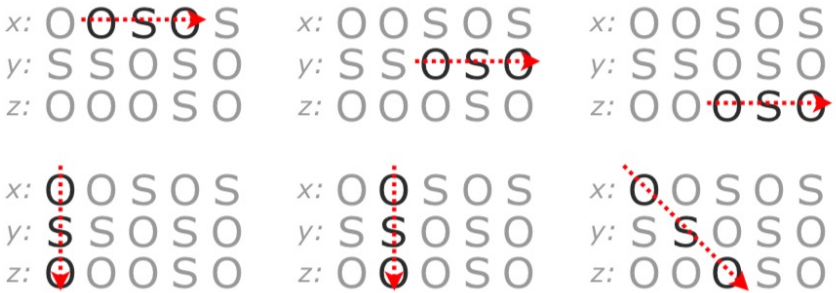
```
def contar_osos(x,y,z):
    n = len(x)
    num = 0

    #Horizontal
    for i in range(0,n-2):
        if x[i]+x[i+1]+x[i+2] == "OSO":
            num += 1
        if y[i]+y[i+1]+y[i+2] == "OSO":
            num += 1
        if z[i]+z[i+1]+z[i+2] == "OSO":
            num += 1

    #Vertical
    for i in range(0,n):
        if x[i]+y[i]+z[i] == "OSO":
            num += 1

    #Diagonal
```

```
    return num
```




```
#CODIGO PRINCIPAL (TEST)
x = "OOSOS"
y = "SSOSO"
z = "OOOSO"

num = contar_osos(x,y,z)
print(num)
```

```python
def contar_osos(x,y,z):
    n = len(x)
    num = 0

    #Horizontal
    for i in range(0,n-2):
        if x[i]+x[i+1]+x[i+2] == "OSO":
            num += 1
        if y[i]+y[i+1]+y[i+2] == "OSO":
            num += 1
        if z[i]+z[i+1]+z[i+2] == "OSO":
            num += 1

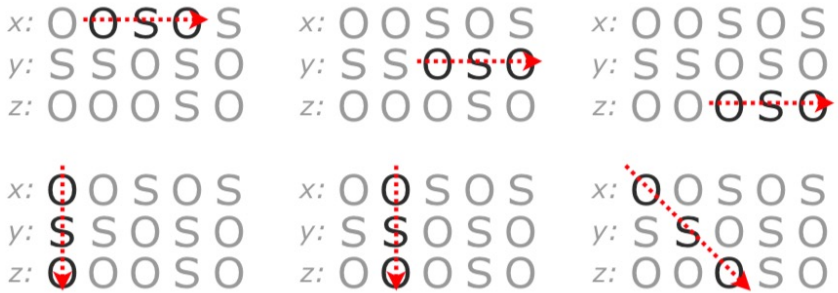
    #Vertical
    for i in range(0,n):
        if x[i]+y[i]+z[i] == "OSO":
            num += 1

    #Diagonal
    for i in range(0,n-2):
        if x[i]+y[i+1]+z[i+2] == "OSO":
            num += 1

    return num


#CODIGO PRINCIPAL (TEST)
x = "OOSOS"
y = "SSOSO"
z = "OOOSO"

num = contar_osos(x,y,z)
print(num)
```



MÉTODOS

```
t = "Rubi"
a = t.islower()
print(a)

p = "terrassa"
b = p.islower()
print(b)

y = "Castellar"
z = y.lower()
print(z)

m = "SABADELL"
n = m.isupper()
print(n)

f = "Sant Llorenç Savall"
g = f.upper()
print(g)
```

```
False
True
castellar
True
SANT LLORENÇ SAVALL
```

- Métodos
  - Funciones especiales que se llaman '.metodo(...)' después de la variable
  - Se explicarán en detalle en el futuro (OO)
  - Strings tiene MUCHOS métodos
    - **TEMARIO SOLO ESTOS 4**

GMAIL (+ y .)

```
Email: example@gmail.com
example@gmail.com


Email: ExAmPle@gmail.com
example@gmail.com


Email: mi.nombre@gmail.com
minombre@gmail.com


Email: Nombre.Apellido1.Apellido2@gmail.com
nombreapellido1apellido2@gmail.com

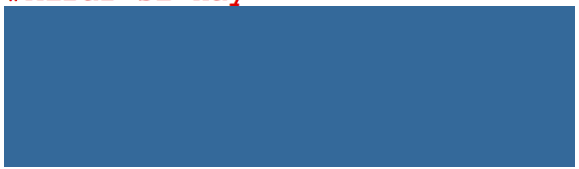
Email: example+iic1103@gmail.com
example@gmail.com
```

```
largo = input("Email: ")

#Buscar la arroba (siempre hay)


#Quedarse con el user


#Quitar puntos


#Mirar si hay +


#Quitar despues de +


#Minuscula


#Agregar el gmail

```

```
largo = input("Email: ")

#Buscar la arroba (siempre hay)
for i in range(0,len(largo)):
    if largo[i] == "@":
        a = i

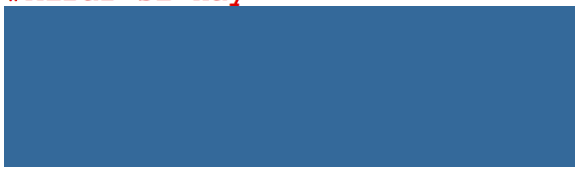
#Quedarse con el user
```



```
#Quitar puntos
```



```
#Mirar si hay +
```



```
#Quitar despues de +
```



```
#Minuscula
```



```
#Agregar el gmail
```



```
largo = input("Email: ")

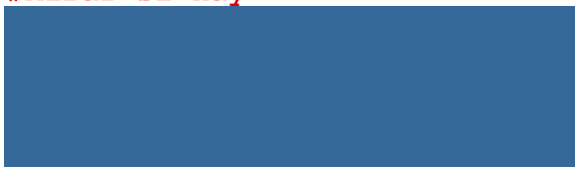
#Buscar la arroba (siempre hay)
for i in range(0,len(largo)):
    if largo[i] == "@":
        a = i

#Quedarse con el user
user = largo[:a]

#Quitar puntos
```



```
#Mirar si hay +
```



```
#Quitar despues de +
```



```
#Minuscula
```



```
#Agregar el gmail
```



```
largo = input("Email: ")

#Buscar la arroba (siempre hay)
for i in range(0,len(largo)):
    if largo[i] == "@":
        a = i

#Quedarse con el user
user = largo[:a]

#Quitar puntos
sp = ""
for c in user:
    if c != ".":
        sp = sp + c

#Mirar si hay +
```



```
#Quitar despues de +
```



```
#Minuscula
```



```
#Agregar el gmail
```



```
largo = input("Email: ")

#Buscar la arroba (siempre hay)
for i in range(0,len(largo)):
    if largo[i] == "@":
        a = i

#Quedarse con el user
user = largo[:a]

#Quitar puntos
sp = ""
for c in user:
    if c != ".":
        sp = sp + c

#Mirar si hay +
#(pos si hay, o -1 si no hay)
m = -1
for i in range(0,len(sp)):
    if sp[i] == "+":
        m = i

#Quitar despues de +
```



```
#Minuscula
```



```
#Agregar el gmail
```



```
largo = input("Email: ")

#Buscar la arroba (siempre hay)
for i in range(0,len(largo)):
    if largo[i] == "@":
        a = i

#Quedarse con el user
user = largo[:a]

#Quitar puntos
sp = ""
for c in user:
    if c != ".":
        sp = sp + c

#Mirar si hay +
#(pos si hay, o -1 si no hay)
m = -1
for i in range(0,len(sp)):
    if sp[i] == "+":
        m = i

#Quitar despues de +
if m != -1:
    limpio = sp[:m]
else:
    limpio = sp

#Minuscula
[redacted]

#Agregar el gmail
[redacted]
```

```
largo = input("Email: ")

#Buscar la arroba (siempre hay)
for i in range(0,len(largo)):
    if largo[i] == "@":
        a = i

#Quedarse con el user
user = largo[:a]

#Quitar puntos
sp = ""
for c in user:
    if c != ".":
        sp = sp + c

#Mirar si hay +
#(pos si hay, o -1 si no hay)
m = -1
for i in range(0,len(sp)):
    if sp[i] == "+":
        m = i

#Quitar despues de +
if m != -1:
    limpio = sp[:m]
else:
    limpio = sp

#Minuscula
limpiomin = limpio.lower()

#Agregar el gmail
```

```python
largo = input("Email: ")

#Buscar la arroba (siempre hay)
for i in range(0,len(largo)):
    if largo[i] == "@":
        a = i

#Quedarse con el user
user = largo[:a]

#Quitar puntos
sp = ""
for c in user:
    if c != ".":
        sp = sp + c

#Mirar si hay +
#(pos si hay, o -1 si no hay)
m = -1
for i in range(0,len(sp)):
    if sp[i] == "+":
        m = i

#Quitar despues de +
if m != -1:
    limpio = sp[:m]
else:
    limpio = sp

#Minuscula
limpiomin = limpio.lower()

#Agregar el gmail
final = limpiomin+"@gmail.com"
print(final)
```

APNEA

IIC1103 – Introducción a la Programación
2018 - 2

Interrogación 1

Pregunta 1

Estás realizando una investigación de pregrado en HumaLab UC, un grupo que investiga en computación centrada en las personas. El grupo ha desarrollado un dispositivo que monitoriza a los adultos mayores mientras duermen para detectar periodos en donde estos no respiran, llamados **apneas**.

Este dispositivo entrega un caracter por cada segundo que pasa. Si el caracter es una "`X`", significa que el adulto mayor no está respirando. Si este es un "`.`", significa que éste sí está respirando. Un ejemplo de un periodo de medición es el siguiente:

```
...XXX...X.X....XXXX......X.XXX.........XXX....XX..X.
```

En el `string` anterior, vemos que el adulto estuvo respirando por 3 segundos, luego tuvo una apnea de 3 segundos, luego volvió a respirar por 3 segundos, luego tuvo una apnea de 1 segundo, y así hasta que finaliza el periodo de medición con una respiración de un segundo. En esta secuencia hubo 9 apneas. Como el grupo sabe que estás cursando el curso Introducción a la Programación, te han pedido que programes las siguientes funciones:

Puedes suponer que la secuencia siempre partirá y terminará con el caracter "`.`".

a) (20 pts) `apneas(secuencia)`: recibe un `string` con una secuencia, y retorna un `int` que representa la cantidad de periodos de apnea que hubo durante el periodo de medición.

Secuencia	`apneas`
...XXX...X.X....XXXX......X.XXX.........XXX....XX..X.	9
...XXX....X...XXXX...XXXX....XXX....XX..XXXXX.	7
.....	0
.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.	1
.X..............XX..............XX............X.....	4

```
def apneas(s):
    num = 0
    for i in range(0,len(s)-1):
        if s[i:i+2] == ".X":
            num += 1
    return num
```

Secuencia	apneas
...XXX...X.X....XXXX......X.XXX.........XXX....XX..X.	9
...XXX....X...XXXX...XXXX....XXX....XX..XXXXX.	7
.....	0
.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.	1
.X..............XX..............XX............X.....	4

b) (20 pts) `max_apnea(secuencia)`: recibe un `string` con una secuencia, y retorna un `int` que representa la duración de la apnea más larga que hubo durante el periodo de medición.

Secuencia	`apneas`	`max_apnea`
...XXX...X.X....XXXX......X.XXX.........XXX....XX..X.	9	4
...XXX....X...XXXX...XXXX....XXX....XX..XXXXX.	7	5
.....	0	0
.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.	1	35
.X..............XX..............XX............X.....	4	2

```
def max_apnea(s):
    maxi = 0
    l = 0
    for i in range(0,len(s)):
        if s[i] == "X":
            l += 1
        elif s[i] == ".":
            if l > maxi:
                maxi = l
            l = 0
    return maxi
```

Secuencia	apneas	max_apnea
...XXX...X.X....XXXX......X.XXX.........XXX....XX..X.	9	4
...XXX....X...XXXX...XXXX....XXX....XX..XXXXX.	7	5
.....	0	0
.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.	1	35
.X.............XX.............XX.............X.....	4	2

c) (20 pts) `kventana(secuencia, k)`: recibe un `string` con la secuencia y un `int` `k`. Retorna `True` si la `secuencia` tiene algún período de `k` segundos consecutivos durante el cual el adulto mayor estuvo más tiempo sin respirar que respirando, o `False` en caso contrario.

Secuencia	apneas	max_apnea	kventana (k=5)	kventana (k=8)
...XXX...X.X....XXXX......X.XXX.........XXX....XX..X.	9	4	True	False
...XXX....X...XXXX...XXXX....XXX....XX..XXXXX.	7	5	True	True
.....	0	0	False	False
.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.	1	35	True	True
.X..............XX..............XX............X.....	4	2	False	False

```
def kventana(s,k):
    res = False
    for i in range(0,len(s)-k+1):
        v = s[i:i+k]
        num = 0
        for c in v:
            if c == "X":
                num += 1
        if num > len(v) / 2:
            res = True
    return res
```

Secuencia	apneas	max_apnea	kventana (k=5)	kventana (k=8)
...XXX...X.X....XXXX......X.XXX.........XXX....XX..X.	9	4	True	False
...XXX....X...XXXX...XXXX....XXX....XX..XXXXX.	7	5	True	True
.....	0	0	False	False
.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.	1	35	True	True
.X.............XX.............XX.............X.....	4	2	False	False