Universidad de Valparaíso
CHILE

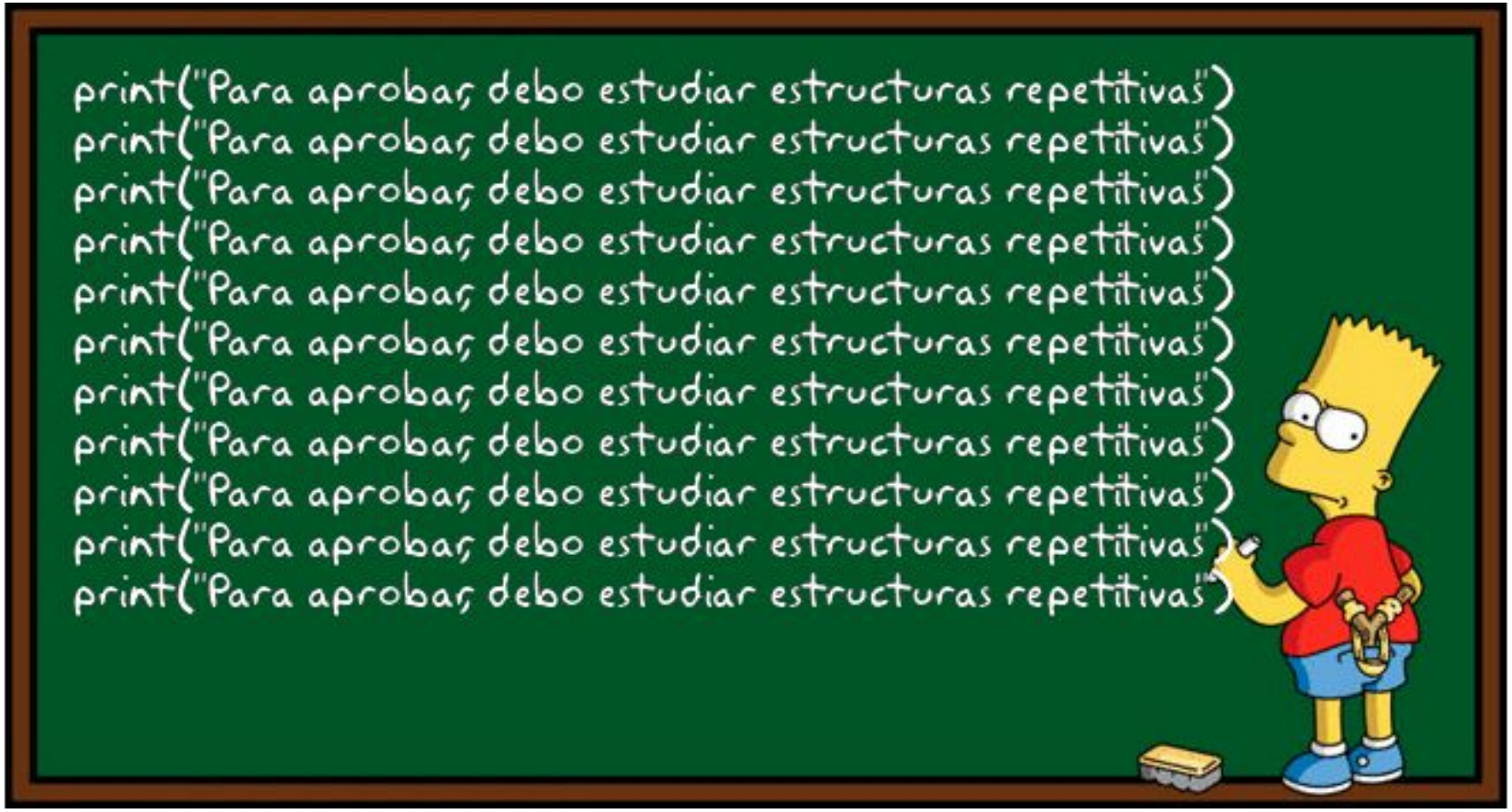
Fundamentos de Programación

Prof. Dr. Roberto Muñoz S. (Sec 1, Lab 502)
Prof. Dr. Rodrigo Olivares O. (Sec 2, Lab 401)
Prof. Víctor Ríos (Sec 3, Lab 303)
Prof. Pablo Olivares (Sec 4, Lab 301)

Escuela de Ingeniería en Informática
Facultad de Ingeniería
Universidad de Valparaíso

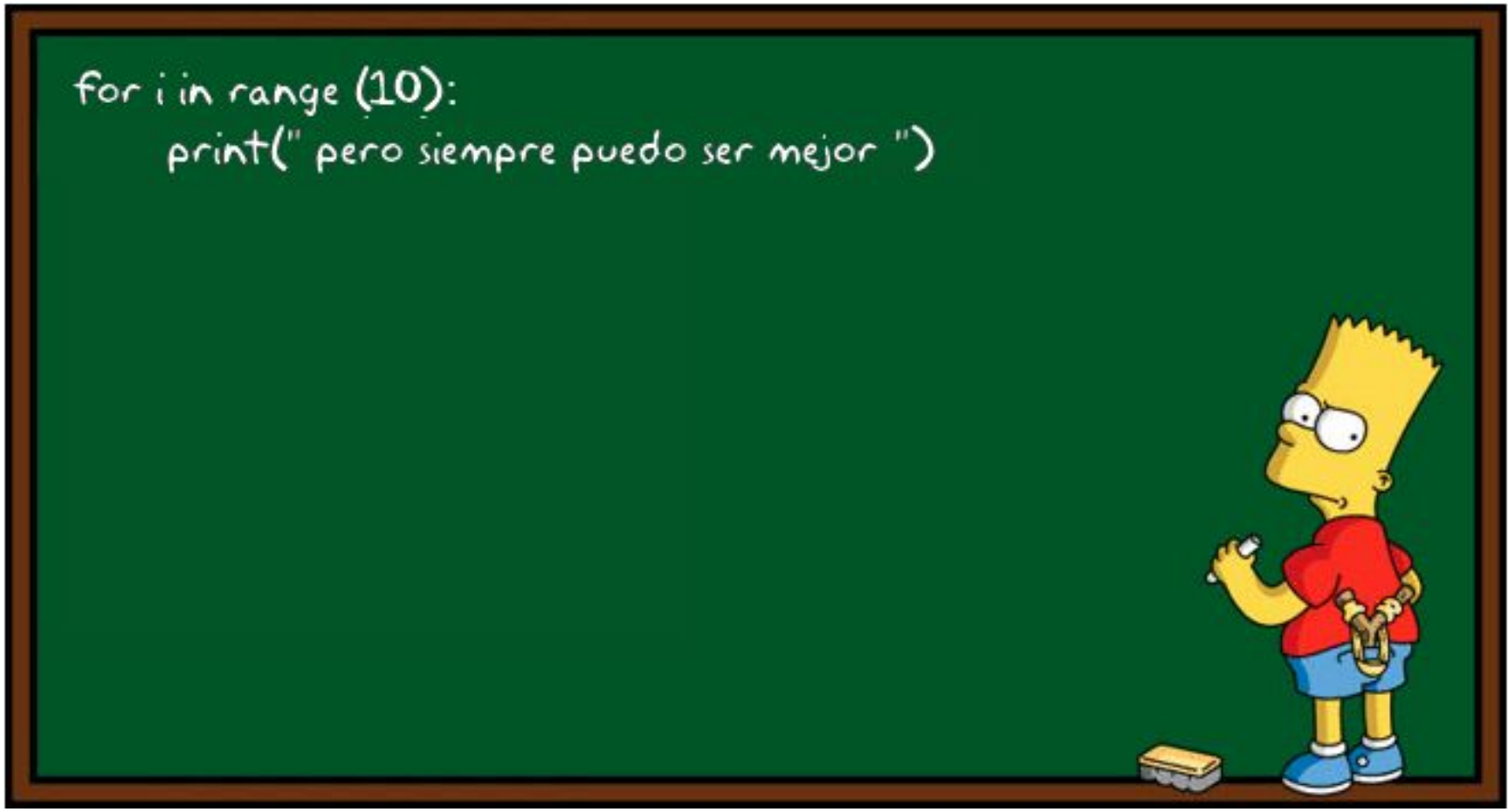
Estructuras repetitivas

Estructuras Repetitivas

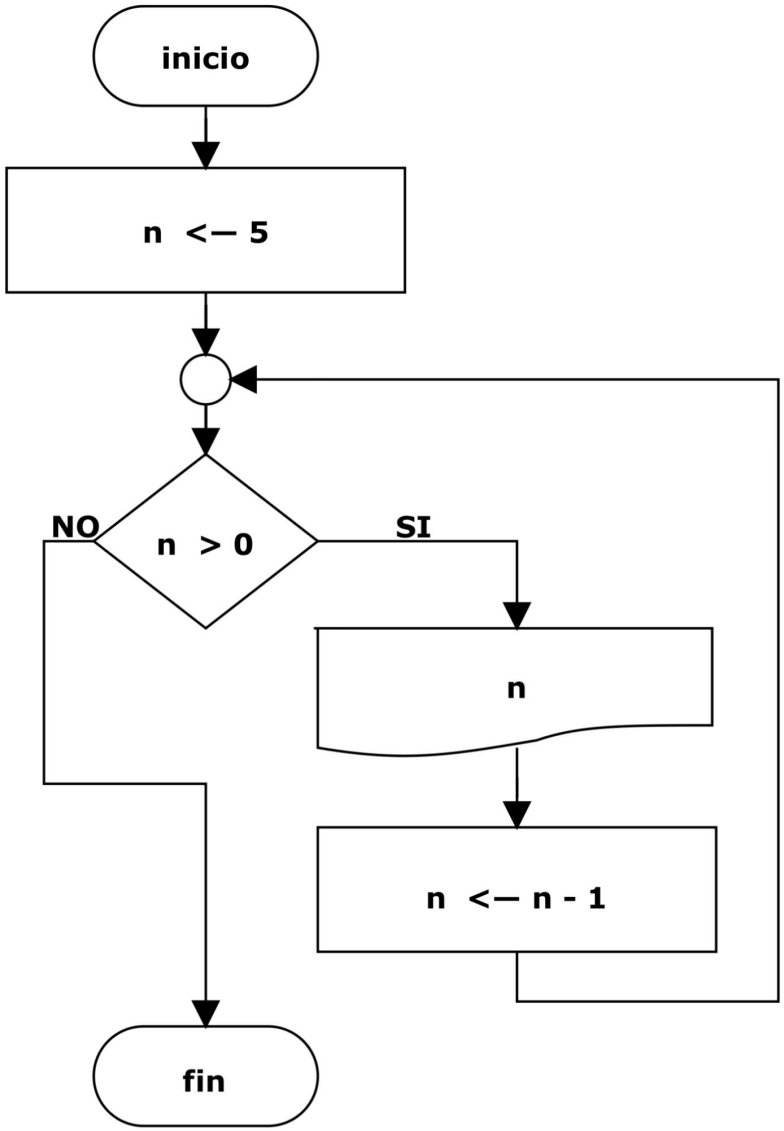


Estructuras Repetitivas

```
i = 0
while i < 10:
    print(" Ya soy experto en estructuras repetitivas ")
    i = i +1
```


for i in range (10):
print(" pero siempre puedo ser mejor ")

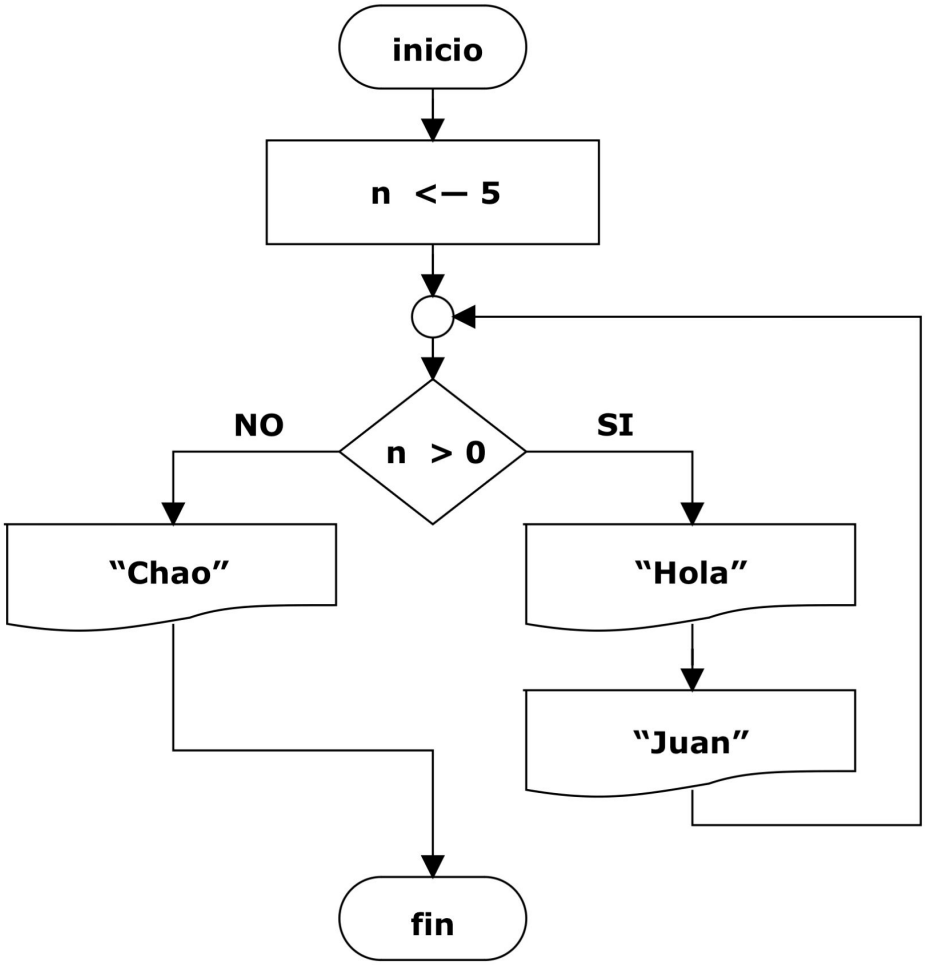
Estructuras repetitivas
While (Mientras)



Salida:

5
4
3
2
1
Fin

Los **bucles** tienen **variables de control** que cambian cada vez que se recorre el bucle. A menudo esas **variables de control** recorren una secuencia de números.



Un bucle Infinito (loop)

```
n = 5
while n > 0:
   print("Hola")
   print("Juan")
print("Chao")
```

¿Qué está mal en este bucle?

Bucles indefinidos

Los bucles ***while*** son llamados "***bucles indefinidos***" porque continúan hasta que una condición lógica se vuelve falsa (False)

En los ciclos que hemos visto hasta ahora es posible determinar si terminan o se vuelven "**bucles infinitos**"

En ocasiones es más **difícil** determinar si un bucle terminará.

Bucles definidos

A veces tenemos una lista de ítems (conjunto de cosas o de elementos en un archivo) o bien queremos que algo se ejecute desde un número determinado de veces con un claro inicio y fin.

Podemos escribir un bucle que recorra el ciclo una vez por cada uno de los ítems de la lista (o bien que se ejecute desde un inicio a un fin) utilizando la estructura ***for.***

Estos bucles son llamados "**bucles definidos**" porque se ejecutan un **número exacto** de veces.

Decimos que "los bucles definidos iteran a través de los miembros de un conjunto".

Ejercicio

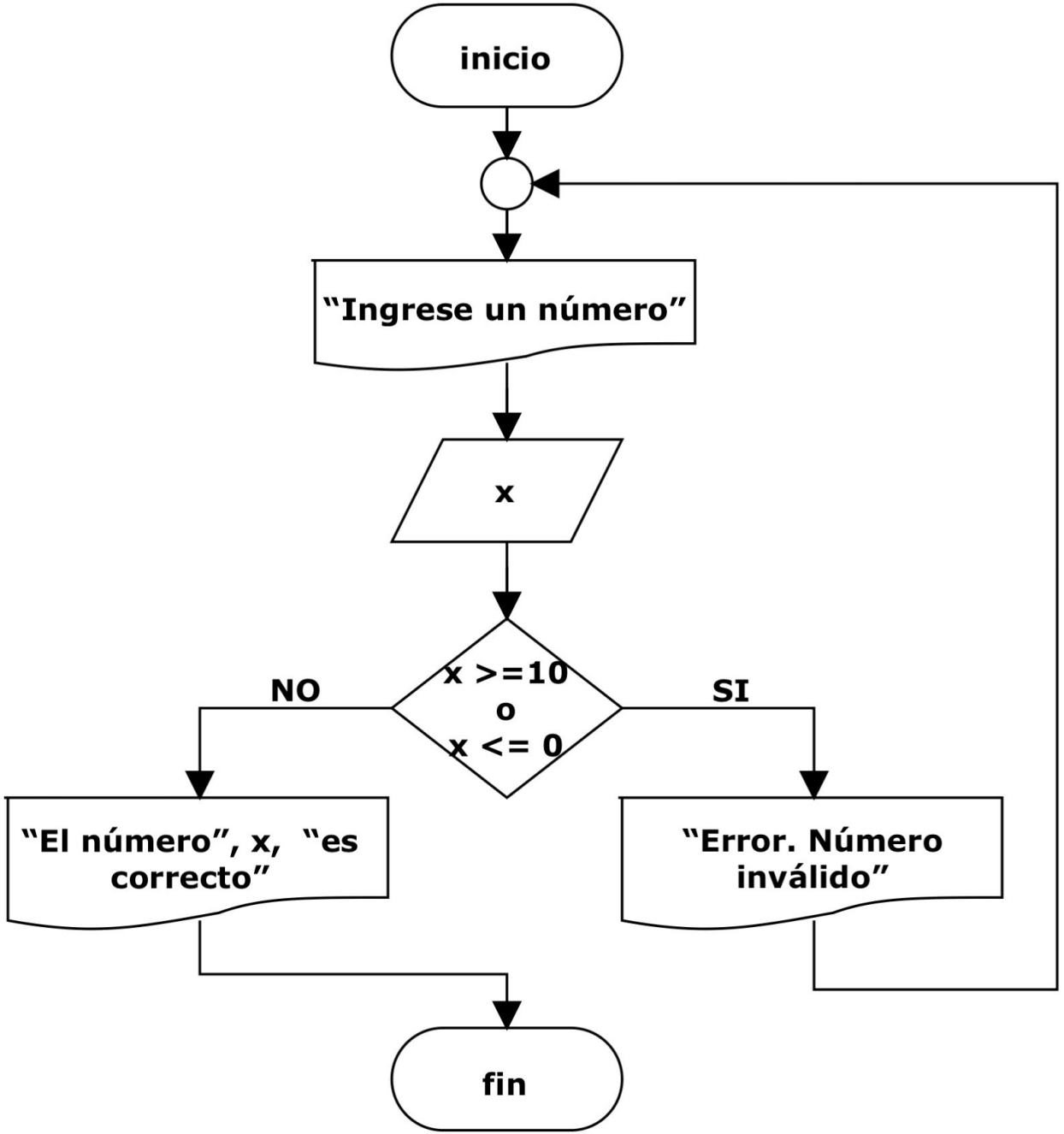
Universidad de Valparaíso
CHILE

Validación de un número

Desarrolle un Diagrama de Flujo y luego implemente en Python, de un programa que reciba un número y sólo termina cuando sea mayor a 0 y menor a 10.

Mientras el usuario ingrese un valor diferente, el programa debe indicar que es un error y debe volver a solicitarlo.

Posible solución



Posible solución en Python 3

¿Cuál es el código en Python 3?
(incluye try/except)

Validar SIEMPRE un número entre 1 y 9


```
while True:
    try:
        print("Ingrese un número:")
        x = int(input())
        while x >= 10 or x <= 0:
            print("Error! Número inválido.")
            print("Ingrese un número:")
            x = int(input())
        break
    except:
        print("Error! No es un número.")
print("El número", x, "es correcto")
```

Ejercicio

Universidad de Valparaíso
CHILE

Ejercicio

Invertir un número

Implemente un programa en Python que **reciba un número mayor a cero**, **extraiga sus dígitos** y lo **invierta**. Si se ingresa un número distinto o se ingresa algo que no sea un número, **se debe indicar error** y se **debe volver a solicitar**. Algunas salidas:

```
Ingrese un número: 2023
3202
```

```
Ingrese un número: 12345678
87654321
```

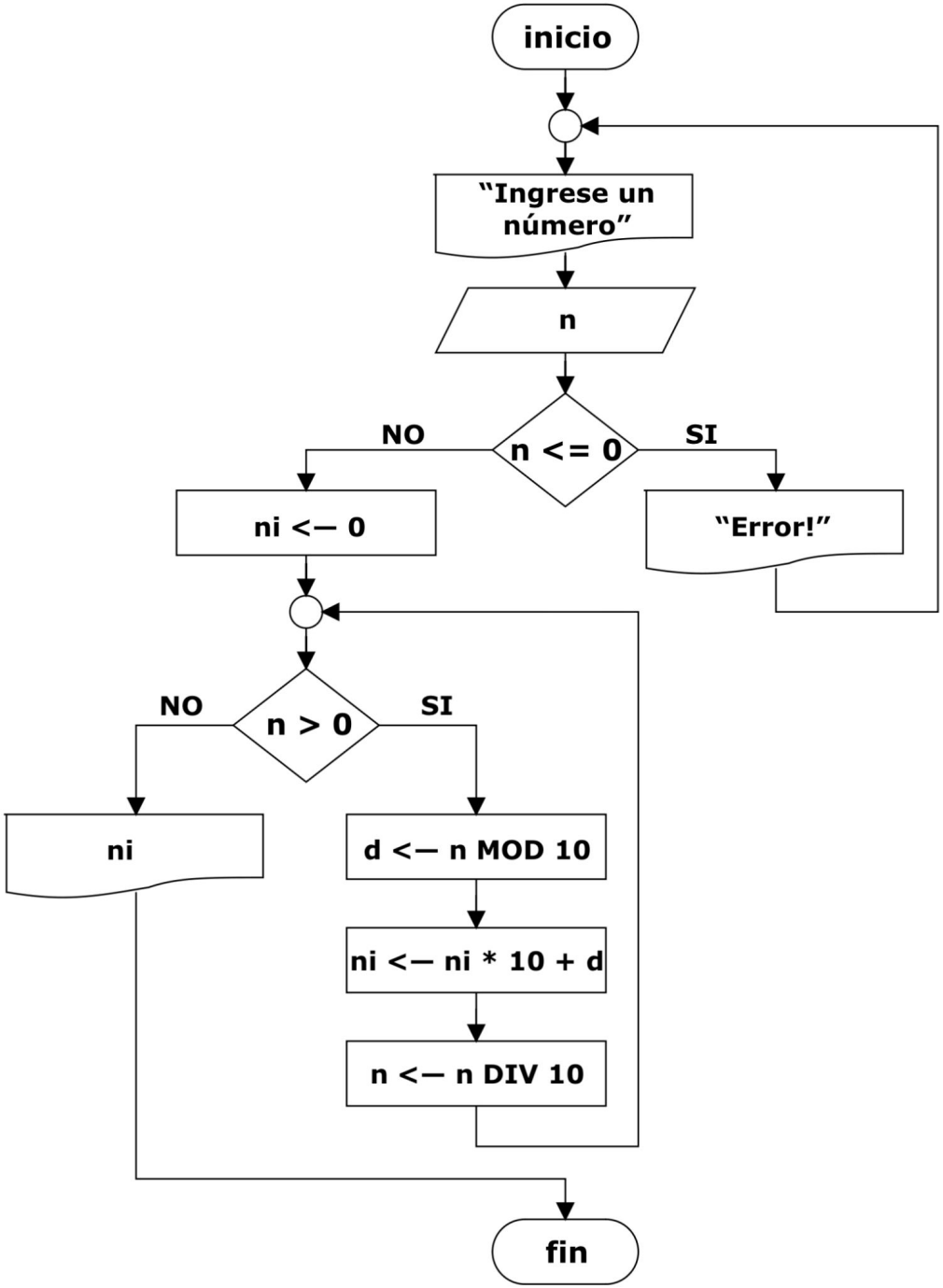
¡No está permitido usar reverse!

```
Ingrese un número: 2o2e
Error.
Ingrese un número: hg$#
Error.
Ingrese un número: 01
1
```

¿Cuál es el diagrama de flujo?

Posible solución

Posible solución



Invertir un número (solución)

```
while True:
    try:
        print("Ingrese un número:")
        num = int(input())
        while num <= 0:
            print("Error! Número inválido.")
            print("Ingrese un número:")
            num = int(input())
        break
    except:
        print("Error! No es un número.")

num_inv = 0
while num > 0:
    digito = num % 10
    num_inv = num_inv * 10 + digito
    num = num // 10
print(num_inv)
```

Ejercicio

Universidad de Valparaíso
CHILE

Descubre el dígito

Un dígito escondido es un elemento que se descubre de un número, mayor que cero, ingresado por el usuario. El dígito se calcula de la siguiente manera:

- Contar la cantidad de ceros que tiene el número. X
- Sumar los dígitos. X
- A la suma anterior, sumar la cantidad de ceros.
- De esa suma final, obtener el dígito. Asuma que el resultado de la suma final tiene siempre 2 dígitos.

Ejemplo:

- **1230456099 -> contador de ceros = 2**
- **12345699 -> 1+2+3+4+5+6+9+9 = 39**
- **39 + 2 -> 41**
- **4 + 1 -> 5**

Ejercicio

¿Cuál es el diagrama de flujo?

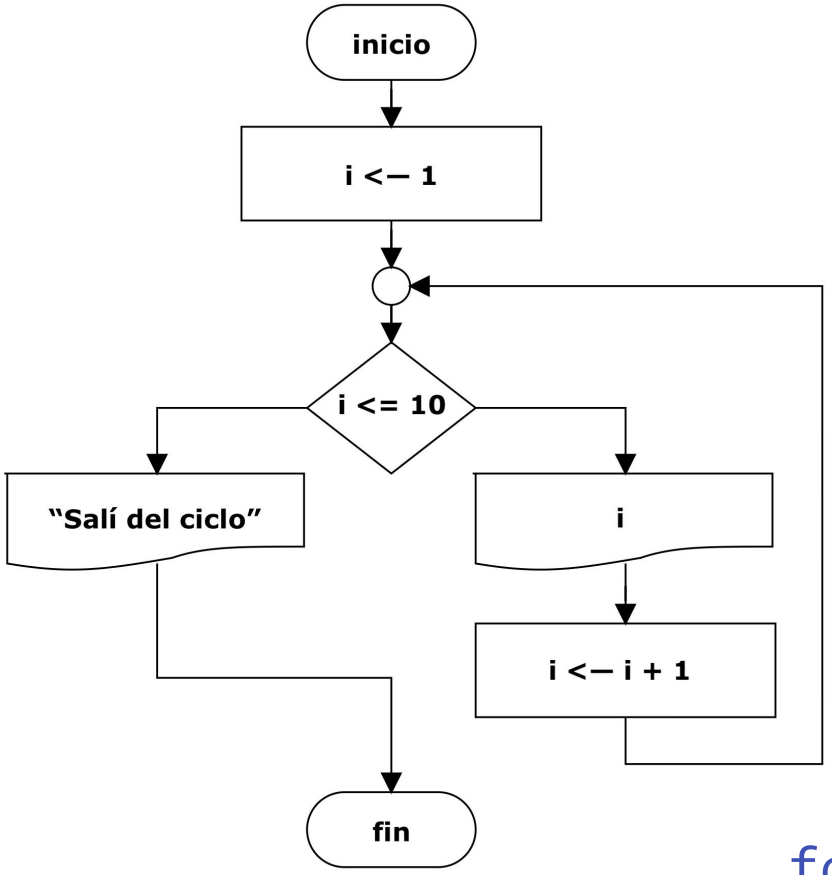
Posible solución

Posible solución

```
while True:
    try:
        num = int(input("Ingrese numero: "))
        while num <= 0:
            print("Error!")
            num = int(input("Ingrese numero: "))
        break
    except:
        print("Error!")

con_ceros = 0
suma = 0
while num > 0:
    digito = num % 10
    num = num // 10
    suma = suma + digito
    if digito == 0:
        con_ceros = con_ceros + 1

suma = suma + con_ceros

unidad = suma % 10
decena = suma // 10
dig_final = decena + unidad
```

Estructuras repetitivas
For (Para)

Bucle **definido**



Los bucles definidos (**bucles *for***) tienen una variable de iteración explícita que cambia cada vez que se recorre el ciclo. Esta variable de iteración se mueve recorriendo la secuencia o conjunto.

```
for i in range(1,11):
    print(i)
print("Salí del ciclo!")
```

Bucle **definido** con rango

```
for i in range(10):
    print(i)

for i in range(0,10):
    print(i)

for i in range(0,10,1):
    print(i)

for i in range(9,-1,-1):
    print(i)
```

Bucles definidos simple

```
for i in range(5,0,-1):
    print (i)
print("Salí del ciclo!")
```

Salida:
5
4
3
2
1
Sali del ciclo!

Una mirada a *in*...

```
for i in [5,4,3,2,1]:
    print (i)
print("Salí del ciclo!")
```

La variable de iteración “itera” a través de la secuencia (conjunto ordenado)

El bloque de código (cuerpo) es ejecutado una vez por cada valor en (*in*) la secuencia

La variable de iteración se mueve a través de todos los valores en (*in*) la secuencia

Ejercicio

Universidad de Valparaíso
CHILE

Ejercicio

- Escriba un programa que pida al **usuario las horas** trabajadas de sus empleados (número de empleados ingresados también por el usuario, mayor a 0) y el **pago por hora** de cada una de ellos, para así calcular sus **pagos brutos** y **líquidos** (**80% del bruto**).
- **¿Cuál es el pago total y promedio de la empresa?.**

```python
while True:
    try:
        num_emp = int(input("Ingrese la cantidad de empleados: "))
        while num_emp <= 0:
            print("Error!")
            num_emp = int(input("Ingrese la cantidad de empleados: "))
        break
    except:
        print("Error!")

valor_hora = 1600
suma_b = 0
suma_l = 0
for i in range(1, num_emp + 1):
    while True:
        try:
            cant_horas = int(input("Ingrese las horas del empleado " + str(i) + " : "))
            while cant_horas <= 0:
                print("Error!")
                cant_horas = int(input("Ingrese las horas del empleado " + str(i) + " : "))
            break
        except:
            print("Error!")
    pago_b = valor_hora * cant_horas
    pago_l = int(pago_b * 0.8)
    print("El pago bruto del empleado", i, "es", pago_b)
    print("El pago líquido del empleado", i, "es", pago_l)
    suma_b = suma_b + pago_b
    suma_l = suma_l + pago_l

print("La empresa pago:", suma_b)
prom_emp = suma_l // num_emp
print("El pago promedio es:", prom_emp)
```

Ejercicio

Universidad de Valparaíso
CHILE

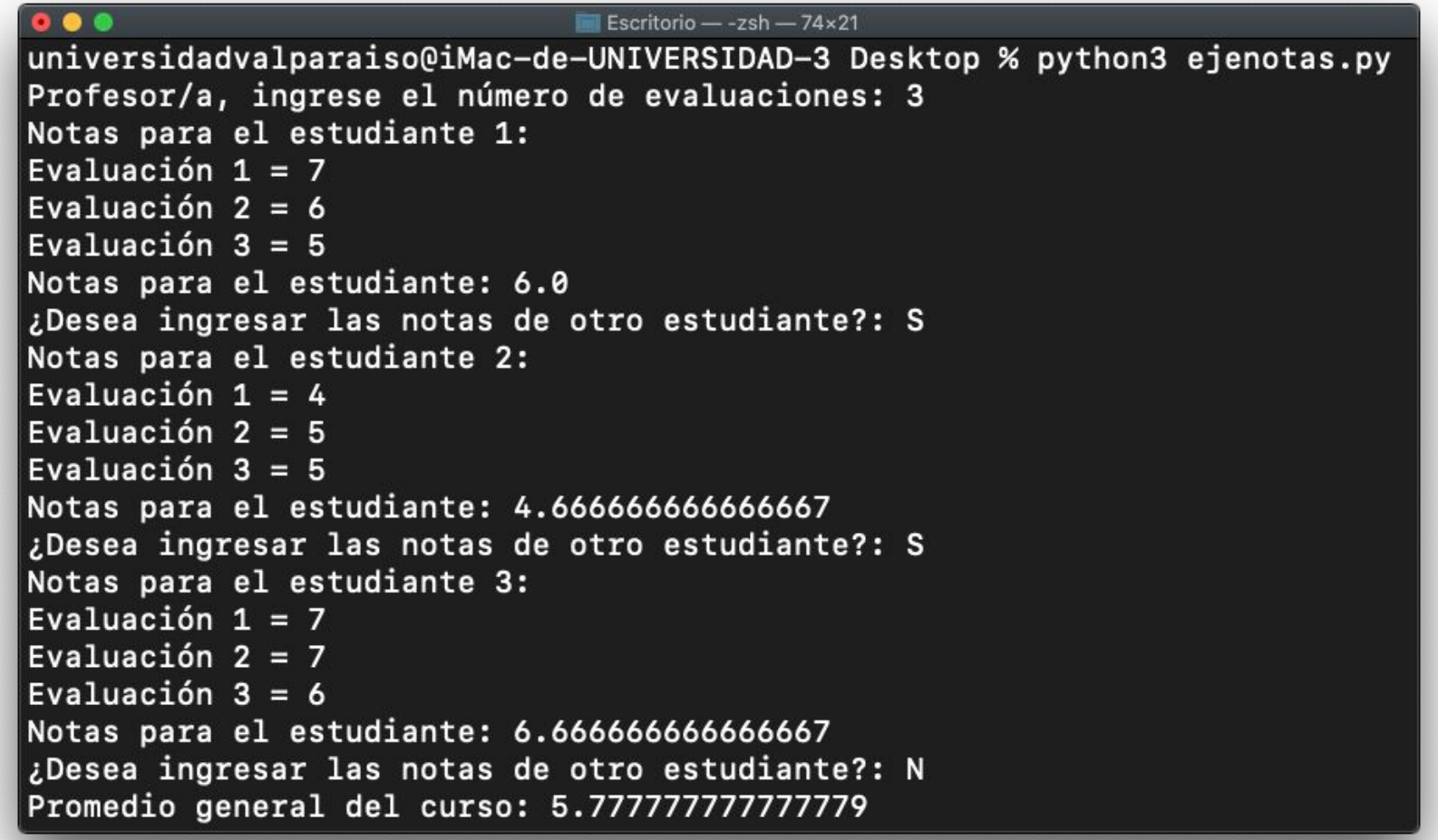
Ejercicio

- Se le ha solicitado que escriba un programa que ayude a su profesor/a a calcular las notas de sus estudiantes y además generar el promedio general del curso.

- La cantidad de evaluaciones de la asignatura es un valor entero mayor a cero, ingresado por el usuario.

- El número de estudiantes es desconocido y se ingresarán mientras el usuario así lo estime. **Recomendación: Use un ciclo que consulte al usuario y capture la opción, por ejemplo, si ingresa *S* se siguen ingresando notas de estudiante.**

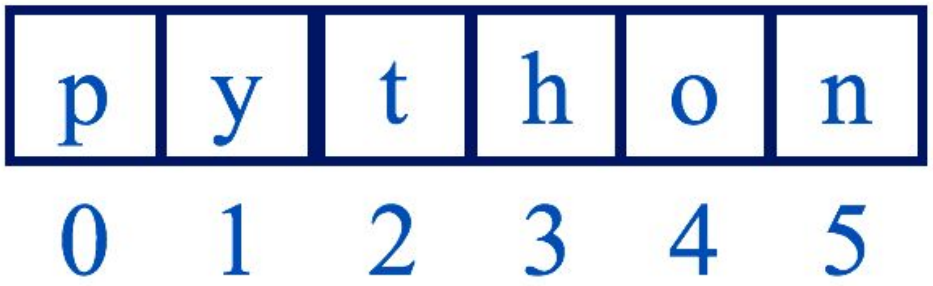
Uso del programa:



Ejercicio

- Se puede acceder a cualquier carácter de una cadena usando un índice especificado entre **corchetes**
- El valor del **índice** debe ser un **número entero** y empezar por **cero**.
- El valor del índice puede ser una **expresión** que da como resultado un **entero**.

‘python’ ⟺




```
lenguaje = 'python'
letra = lenguaje[1]
print(letra)
```

```
> y
```

```
i = 3
palabra = lenguaje[i-1]
print(palabra)
```

```
> t
```

- Si se intenta **indexar** (posicionar) más allá del último carácter de una secuencia, se obtendrá un error.

- Por tanto, hay que tener cuidado al construir los valores del índice y sus partes

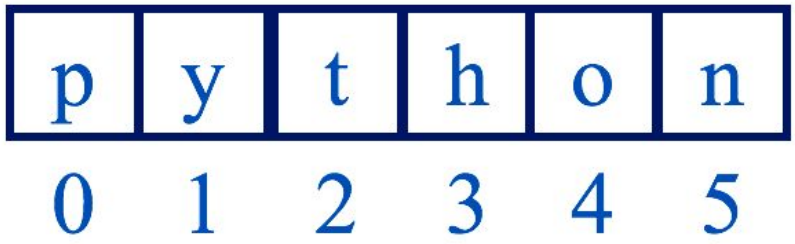
```
lenguaje = 'python'
letra = lenguaje[10]
print(letra)
```

```
> IndexError: string index out of range
```

- La función **len** del lenguaje retorna la longitud de una cadena

```
lenguaje = 'python'
print(len(lenguaje))
```

```
> 6
```



Usando la estructura **while** y una **variable de iteración** más la función **len**, se genera un bucle que permite ver cada una de las letras de una secuencia por separado.

```
print("Mostrar letra y posición:")
lenguaje = 'python'
i = 0
while (i < len(lenguaje)):
    letra = lenguaje[i]
    print("Letra:", letra, "Posición:", i)
    i = i + 1
```

```
> Mostrar letra y posición
Letra: p Posición: 0
Letra: y Posición: 1
Letra: t Posición: 2
Letra: h Posición: 3
Letra: o Posición: 4
Letra: n Posición: 5
```

Usando la estructura **for** y una **variable de iteración** con la función **range** que incluye la función **len**, se genera el mismo resultado anterior.

```
print("Mostrar letra y posición")
lenguaje = 'python'
for i in range(len(lenguaje)):
    print("Letra:", lenguaje[i], "Posición:", i)
```

```
> Mostrar letra y posición
Letra: p Posición: 0
Letra: y Posición: 1
Letra: t Posición: 2
Letra: h Posición: 3
Letra: o Posición: 4
Letra: n Posición: 5
```

- Sin embargo, utilizar la instrucción **for** en conjunto con el método **in**, el código queda más fino ☺️

```
lenguaje = 'python'
for letra in lenguaje:
    print("Letra:", letra)
```

```
> Letra: p
Letra: y
Letra: t
Letra: h
Letra: o
Letra: n
```

in como operador

- La expresión ***in*** también sirve para comprobar si una cadena está «***en***» otra cadena

- *in* es una expresión lógica que devuelve True (verdadero) o False (falso) y que puede usarse en una instrucción if

```
if 'Py' in s:
    print("Está!")
```

Buscar…

- *str.capitalize()*
- *str.center(width[, fillchar])*
- *str.endswith(suffix[, start[, end]])*
- *str.find(sub[, start[, end]])*
- *str.lstrip([chars])*
- *str.replace(old, new[, count])*
- *str.lower()*
- *str.rstrip([chars])*
- *str.strip([chars])*
- *str.upper()*

- .upper()
- .lower()
- .find(str)
- .replace(str,str)
- .rstrip(), lstrip()
- .startswith(str)

Ejercicio

Universidad de Valparaíso
CHILE

Descomponer una cadena de caracteres

Implemente un programa en Python que reciba una cadena de caracteres y muestra el conteo de:

- Vocales. Revise la función **isalpha().**
- Consonantes. Revise la función **isalpha().**
- Dígitos. Revise la función **isdigit().**

Salida:

```
Ingrese una cadena: abcdefdg12300o
Cantidad de vocales: 3
Cantidad de consonantes: 6
Cantidad de dígitos: 5
```

Ejercicio

Ejercicio - Solución

```
cadena = input("Ingrese una cadena: ")
vocales = "aeiou"
cont_d = cont_v = cont_c = 0
for c in cadena:
    if c.isdigit():
        cont_d = cont_d + 1
    elif c.isalpha():
        if c in vocales:
            cont_v = cont_v + 1
        else:
            cont_c = cont_c + 1
print("Cantidad de vocales:", cont_v)
print("Cantidad de consonantes:", cont_c)
print("Cantidad de dígitos:", cont_d)
```

Ejercicio

Universidad de Valparaíso
CHILE

Leet speak

Leetspeak, o simplemente leet (**13375p34k** o **1337** en la escritura **leet**) es un tipo de escritura compuesta **sólo de caracteres alfanuméricos.** Los códigos leets de **nivel 1** son : {A=4, B=8, E=3, G=9, I=1, L=1, O=0, S=5, T=7, Z=2}. Ciertas comunidades y usuarios de diferentes medios de Internet utilizan esta escritura para formar textos incomprensibles que no pueden ser descifrados por *script kiddies*.

Desarrolle un programa en Python que reciba una cadena de caracteres y muestre el texto en formato **13375p34k**.

Salida:

```
Ingrese una cadena: texto de prueba
Texto de salida leet: 73x70 d3 pru384
```

Ejercicio

Ejercicio

```
cadena = input("Ingrese una cadena:").upper()
leet = ""
for c in cadena:
    if c == '0':
        leet = leet + "0"
    elif c == 'I':
        leet = leet + "1"
    elif c == 'Z':
        leet = leet + "2"
    elif c == 'E':
        leet = leet + "3"
    elif c == 'A':
        leet = leet + "4"
    elif c == 'S':
        leet = leet + "5"
    elif c == 'T':
        leet = leet + "7"
    elif c == 'B':
        leet = leet + "8"
    elif c == 'G':
        leet = leet + "9"
    else:
        leet = leet + c

print("Texto de salida leet", leet)
```

Ejercicio

Universidad de Valparaíso
CHILE

Universidad de Valparaíso CHILE

Sars-Cov 2 - ARNm (mensajero)

Revisar el ejercicio en Aula Virtual

Datos de prueba:

Válido
27733106808318069615188831091273366975186091

Inválidos
1091233675
109123xf3675

Ejercicio

Sars-Cov 2 - ARNm (mensajero) : Solución 1

```
while True:
    cadena = input("Ingrese la secuencia ARNm:")
    while not cadena.isnumeric() or len(cadena) % 2 != 0:
        print("Error!")
        cadena = input("Ingrese una nueva secuencia ARNm:")
    break

num = int(cadena)
texto = ""
while num > 0:
    n = num % 100
    if n >= 65 and n <= 90:
        texto = texto + chr(n)
    num = num // 100

if "S" in texto and "P" in texto and "I" in texto and "K" in texto and "E" in texto:
    print("Se produce la proteína SPIKE!")
else:
    print("No es un ARNm válido!")
```

Sars-Cov 2 - ARNm (mensajero) : Solución 1

```
while True:
    cadena = input("Ingrese la secuencia ARNm:")
    while not cadena.isnumeric() or len(cadena) % 2 != 0:
        print("Error!")
        cadena = input("Ingrese una nueva secuencia ARNm:")
    break

texto = ""
for i in range(len(cadena) - 1, -1, -2):
    codigo = int(cadena[i-1] + cadena[i])
    if codigo >= 65 and codigo <= 90:
        texto = texto + chr(codigo)

if "S" in texto and "P" in texto and "I" in texto and "K" in texto and "E" in texto:
    print("Se produce la proteína SPIKE!")
else:
    print("No es un ARNm válido!")
```

Universidad de Valparaíso
CHILE

Fundamentos de Programación

Prof. Dr. Roberto Muñoz S. (Sec 1, Lab 502)
Prof. Dr. Rodrigo Olivares O. (Sec 2, Lab 401)
Prof. Víctor Ríos (Sec 3, Lab 303)
Prof. Pablo Olivares (Sec 4, Lab 301)

Escuela de Ingeniería en Informática
Facultad de Ingeniería
Universidad de Valparaíso