Strings 🎉

Clase #13

IIC1103 – Introducción a la Programación

El plan de hoy es...

- + plazo set hasta el miércoles a las 7pm

- Recursión
- print("hola") #Strings

Problema #0

- Escribe un programa que lea una lista de números positivos (hasta que el usuario ingrese -1) y luego escriba el número más grande
- `numero? 134`
- `numero? 98`
- `numero? 4`
- `numero? 256`
- `numero? -1`
- `El mayor numero es: 256`

Solución

```
mayor = -1
sigue = True
while sigue:
    numero= int(input("numero?"))
    if numero== -1:
        sigue = False
    elif numero>mayor:
        mayor=numero
print("El mayor numero es "+str(mayor))
```

Problema #1

- Escribe un programa que lea una lista de palabras (hasta que el usuario ingrese “fin”) y luego escriba la palabra más larga
- `palabra? casa`
- `palabra? felicidad`
- `palabra? arbol`
- `palabra? dedo`
- `palabra? fin`
- `La palabra más larga es: felicidad`

Solución

```
mayor = ""
sigue = True
while sigue:
    palabra = input("palabra?")
    if palabra == "fin":
        sigue = False
    elif len(palabra) > len(mayor):
        mayor=palabra
print("La palabra más larga es "+mayor)
```

Str: len

- Entrega nº de caracteres

- `s="hola"`
- `largo = len(s)`
- `s="hola como estas"`
- `largo = len(s)`

Strings (texto)

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

- Un String representa una cadena alfanúmerica.
- En general, trabajaremos con Strings mediante métodos (`x.met()`)/funciones (como `func(x)`).
- En comillas simples (‘hola’), dobles(“hola”) o triples(“’’hola’’’) (¿Por qué?)
  - Maneras de escribir **`El nombre del restaurant es “McDonald’s”`**

Operador + en strings

- ¿Qué hace?

- Recordar también: +=, *

Problema #2

- Escribe una función que determine si una palabra es palíndrome o no

Solución

```
def palindromo(x):
    i=0
    while i<len(x)//2:
        if x[i]!=x[len(x)-i-1]:
            return False
        i+=1
    return True
```

Str: s[i]

- Entrega caracter ubicado en el índice i (desde 0)

- s=”abracadabra"
- c = s[1]
- c = s[4]

Cuidado con salir del rango del string!

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

- c = s[11]

Advertencia

- Asignación de variables:

- c = s[1] #ok
- **s[1] = “a” #NO**

Solución recursiva

```
def palindrome(s):
    if len(s)<=1:
        return True
    if s[0] != s[len(s)-1]:
        return False
    return palindrome(s[1:len(s)-1])
```

Str: s[i:j] (string slices)

- Entrega caracteres desde índice i hasta j-1

- `s=”abracadabra"`
- `c = s[1:3]`
- `c = s[0:11]`

`c = s[2:6]`

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

- Variaciones:
  - `s[:j]`
  - `s[i:]`

Solución

```
def palindrome(s):
    if s==s[::-1]:
        return True
    return False

def palindrome(s):
 return s==s[::-1]
```

Str: s[i:j:k] (string slices, con 3 argumentos)

- Entrega caracteres desde índice i hasta j-1, entregando los caracteres cada k caracteres.
- `s="abracadabra"`
- `c = s[0:len(s):2]`
- `c = s[0:len(s):3]`
- `c = s[::-1]`
- `c = s[1:9:2]`

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

Problema #3

- Escribe una función **recursiva** para calcular el largo de un string, sin usar `len`.

Resumen del día

- **Hoy vimos**:

x →

a	b	r	a	c	a	d	a	b	r	a
0	1	2	3	4	5	6	7	8	9	10

- x[4]

a	b	r	a	**c**	a	d	a	b	r	a
0	1	2	3	**4**	5	6	7	8	9	10

- x[2:6]

a	b	**r**	**a**	**c**	**a**	d	a	b	r	a
0	1	**2**	**3**	**4**	**5**	6	7	8	9	10

- x[1:9:2]

a	**b**	r	**a**	c	**a**	d	**a**	b	r	a
0	**1**	2	**3**	4	**5**	6	**7**	8	9	10