Programación Y Robotica 4º ESO 24_25

Clase 10- Python: Ahorcado, explica-1: listas y cadenas de caracteres

Enero 2025

Colegio Santo Domingo FESD - Madrid

Profesor voluntario :

Jose Carlos Santamaria Poza : jcspoza@gmail.com

4ESO-CL10 – Objetivos / Índice

- Punto de situación
- Para Entender Ahorado necesitamos conocer :
  - Listas
  - Cadenas caracteres
- Estudiar Ahorcado 1ª
- Módulos (recordar)

1. Punto situación dentro del Mapa de conceptos de programación
2. HACER : uso de Cadenas de caracteres + f-strings
3. HACER : uso de Listas
4. RECORDAR el diseño del ‘Ahorcado’: bloques en “pseudocodigo”
5. ESTUDIAR: “AhorcadoSimple” 1ª
6. Reto CL10: Usar listas y cadenas

Punto situación dentro del Mapa de conceptos de programación

Estructura de Flujo operaciones

- ✓ **Secuencia**
- ✓ **Selección**
- ✓ **Bucles** While/ for
- ✓ **Errores/ Eventos** / Contextos

Tipos de Datos

Variables

- ✓ **Tipos Simples** Int, float
- ✓ **Cadenas caracteres** Rebanado, métodos,..
- **Com-puestos** Listas, .

Nuevos objetos (OOP)

Módulos: módulos gráficos GUI´, tratamiento datos para ciencia o IA

Objetivo de programa …

✓ **Programación Funcional: parámetros**

'Name space'

Funciones sobre funciones : Decoradores

Objetivo de programa para 3 y 4 ESO

Programación Orientada a Objetos: Herencia, métodos, ……

Inter-process communication: consumo API´s HTTP, modulo requests, JSON, …

Multiproceso : Programación asíncrona, corutinas, ..

RECORDAR : Cadenas profundizar: Rebanar cadenas y trocear
Referencia detallada en https://ellibrodepython.com/cadenas-python

5'

Rebanar: si dentro de [] usamos 1,2 o 3, que corresponde a [inicio: fin : paso] números separados por ':', conseguiremos 'rebanar' un trozo de la cadena.

Si no se indica inicio **[:n]** o fin **[n:]** , se entiende principio o fin de la cadena

Trocear : **split**(sep=None): devuelve una **lista** de las palabras de la cadena separadas acorde el parámetro sep. Si este parámetro no se especifica, la cadena se separa teniendo en cuenta los espacios en blanco

Juntar : El método **join**() devuelve la primera cadena unida a cada uno de los elementos de la lista que se le pasa como parámetro.

Caracteres :	P	y	t	h	o	n
Índice :	0	1	2	3	4	5
Índice inverso :	-6	-5	-4	-3	-2	-1

```
>>> cadena = 'Python'
>>> cadena[1:4] # indices 1, 2 y 3
'yth'
>>> cadena[1:] # indice 1 a final
'ython'
>>> cadena[1:-1] #de indice 1 a penultimo
'ytho'
```

```
>>> cadena = 'el tio pepe de madrid'
>>> s = cadena.split()
>>> s
['el', 'tio', 'pepe', 'de', 'madrid']
>>> cjuntar = "".join(s)
>>> cjuntar
'eltiopepedemadrid'
```

HACER : uso + de Cadenas de caracteres + f-strings

Vamos a practicar el uso de listas con el IDE online-Python https://www.online-python.com/

Para ampliar podeis mirar en **https://ellibrodepython.com/cadenas-python**

Cadenas de caracteres: + funciones:

len(cadena) -> numero de carácter de la cadena / str(int o float) -> convierte a cadena un numero

s = "Hola"; print(s*3) —> "HolaHolaHola" / ss = 'Hola' + 'Pepe' ; print(ss) —> 'HolaPepe'

print(**ord**('$')) -> 36 / print(**chr**(36)) —> '$' : convertir carácter y valor numérico LIOSO si caracteres raros

Métodos de la clase string: capitalize(), lower(), upper(), strip([<chars>]) .....

Uso de los métodos : '### hola ##'.strip('#') —> ' hola ' :

F-strings: es la forma mas moderna de **formatear cadenas de caracteres e incluir valores de variables o funciones**

Mira el ejemplo del programa P_ahorcado_Simple.py, en función muestraPanel

print(f'Letras Falladas {len(letrasFall)} de {MAXFALLOS} fallos: {letrasFall}')

HACER : uso de Listas

https://www.online-python.com/

Vamos a practicar el uso de listas con el IDE online-Python

Vamos a hacer algunos casos de https://ellibrodepython.com/listas-en-python

Listas: objetos **compuesto** que permiten almacenar un c**onjunto** de datos, que pueden ser de **distinto tipo**, se pueden **modificar** los valores y **añadir** nuevos elementos. Se pueden anidar = tener listas de istas

Son muy versátiles y por eso se usan en muchísimos programas de Python.
Algunos ejemplos de uso

Crear : l = [1, 2, 3, 4] / o con algunos objetos usando l = list(range(0,5)) , l-> [0,1,2,3,4]

Aceder : com índices como los strings a = [90, "Python", 3.87], a[1] -> "Python",

Añadir al final con el método append() / borrar el ultimo con .pop()

……..

RECORDAR : Diseño del 'Ahorcado': bloques en "pseudocodigo"

Constantes: lista palabra secretas, dibujos ahorcado , alfabeto, máximo de fallos, texto de instrucciones, en MÓDULO aparte

1-Inicio

Carga las **CONSTANTES** **en módulo a parte** y **define las 4 funciones**

Presenta-ción

Instruccio-nes

Inicializa variables dinámicas

+

Elige palabra a Adivinar

palabraSecreta

2-Bucle de Juego

Mientras No ***JuegoAcabado***

Mostrar panel

Llama a FUNCION muestraPanel()

Elige Letra nueva

Llama a FUNCION adivinaLetra()

Dinámica de Juego:

Si letra nueva ESTA en palabra secreta

Actualiza ***letrasCorrectas***

CheckGano()

Si letra nueva NO ESTA

Actualiza ***letrasFalladas***

CheckPerdio()

Actualiza **JuegoAcabado** **todasLetrasEncontradas**

3-Fin

Si **TodasLetrasEncontradas**

Mensaje Gano

Si No

Mensaje Perdio

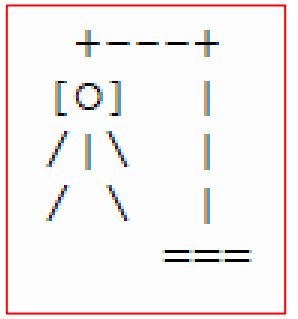
Variables dinámicas : ***letrasFalladas, letrasCorrectas, todasLetrasEncontrada, JuegoAcabado***

PROGRMMING ROBOOTIICS

ESTUDIAR y PROBAR: "AhorcadoSimple"

https://replit.com/~

Link al repositorio en REPLIT



https://replit.com/@jsantamariadom/CL9ahorcado?v=1

Con el seudocódigo definido, ahora podemos entender el programa (mira el seudocódigo en los comentarios) . Empezaremos por la versión más simple

P_ahorcado_Simple.py —> cambiamos el nombre a 'main.py' para ejecutarlo

(si da tiempo) Las versiones con la mejora 1 y la versión con la mejora 1y2, solo cambian la función muestraPanel()

Reto CL10 : Usar listas y cadenas

3 opciones para el reto

a. Recorrer una lista simple con for

Hacer un programa que defina una **lista** con varios elementos y que los muestre 1 a 1 con print dentro de un bucle **for**

b. Recorrer una cadena con for

Hacer un programa que defina una **cadena de caract**eres y que los muestre 1 a 1 con print dentro de un bucle **for**

c. (mas difícil) Recorrer una lista anidada con 2 for anidados

Usa el programa con la **lista anidada** ya definida llamado P_ListadobleRETO10c_input.py

Continúa el programa con un doble bucle **for** que muestre la lista

SABER (recordar) : Módulos: ¿Por qué? Ventajas

Los módulos en Python son una de sus características más **poderosas y versátiles,** ***son simplemente ficheros separados que se ejecutan junto a otros***

- **Organización:** Los módulos permiten organizar el código en unidades más pequeñas, lo que mejora la legibilidad y facilita la colaboración entre desarrolladores.
- **Reusabilidad:** Permiten agrupar funciones, clases o bloques de código relacionados en el mismo archivo. En lugar de copiar sus definiciones en varios programas, podemos definir nuestras funciones más utilizadas en un módulo e importarlo.
- **Mantenibilidad:** Al separar el código en módulos, es más sencillo realizar cambios, mejoras o correcciones en áreas específicas sin afectar el resto del programa.

DISEÑAR : Ahorcado: bloques de seudocodigo

Constantes: lista palabra secretas, dibujos ahorcado , alfabeto, máximo de fallos, texto de instrucciones, en MÓDULO aparte

1-Inicio

Carga las **CONSTANTES** **en módulo a parte** y **define las 4 funciones**

Presenta-ción

Instruccio-nes

Inicializa variables dinámicas

+

Elige palabra a Adivinar

palabraSecreta

2-Bucle de Juego

Mientras No ***JuegoAcabado***

Mostrar panel

Llama a FUNCION muestraPanel()

Elige Letra nueva

Llama a FUNCION adivinaLetra()

Dinámica de Juego:

Si letra nueva ESTA en palabra secreta

Actualiza ***letrasCorrectas***

CheckGano()

Si letra nueva NO ESTA

Actualiza ***letrasFalladas***

CheckPerdio()

Actualiza **JuegoAcabado** **todasLetrasEncontradas**

3-Fin

Si **TodasLetrasEncontradas**

Mensaje Gano

Si No

Mensaje Perdio