Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación

Clase 10: Listas

Rodrigo Toro Icarte (rntoro@uc.cl)

IIC1103 Introducción a la Programación - Sección 5

21 de Abril, 2015

Listas

El mundo está lleno de listas.

Listas

El mundo está lleno de listas.

Apellido Paterno	Apellido Materno	Nombre
ÁLVAREZ	JOHNSON	SOFÍA PAZ
ANDRADE	POBLETE	ISIDORA CAROLINA
AYLWIN	REYES	MANUEL EDUARDO
BALBONTÍN	GUMUCIO	NICOLÁS FELIPE
BRAHM	ESCALONA	THOMAS IGNACIO
BRAVO	DE LA CRUZ	JUAN PABLO
BUSTAMANTE	ALONZO	YAEL FERNANDA
CABROLIER	OSSES	BIANCA MACARENA
CÁDIZ	BOSCH	CRISTIAN RICARDO
CANALES	CÓRDOVA	FRANCISCA MACARENA
CANIO	FERNÁNDEZ	NELSON RUBÉN
CARRASCO	SALINAS	RICARDO ANDRÉS
CARVAJAL	MEJIA	NICOLAS IGNACIO
CASTILLO	FUENTES	PABLO FRANCISCO

Listas

El mundo está lleno de listas.

Listas

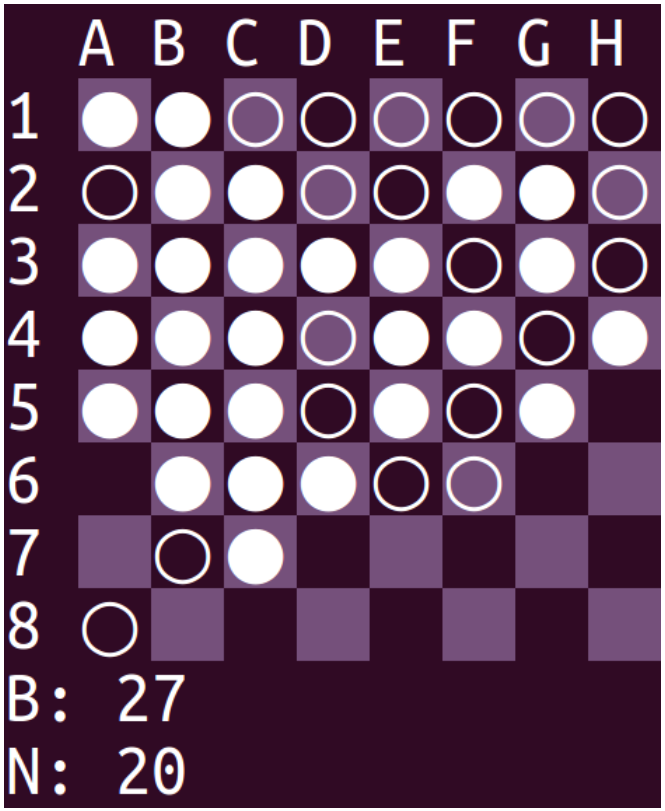
El mundo está lleno de listas.

Listas

El mundo está lleno de listas.

Listas

El mundo está lleno de listas.



Listas

Definición: Una lista es una serie *mutable* e *indexable* de elementos.

Listas

Definición: Una lista es una serie *mutable* e *indexable* de elementos.

Sintaxis

lista = [elemento_1, elemento_2, elemento_3, ...]

Listas

Definición: Una lista es una serie *mutable* e *indexable* de elementos.

Sintaxis

lista = [elemento_1, elemento_2, elemento_3, ...]

```
1 l1 = []                                  # lista vacía
2 l2 = [3,4,2,4,9,6]                       # lista de números
3 l3 = ["a","b","c","d"]                   # lista de strings
4 l4 = ["a",3.4,True]                      # lista mixta
5 l5 = [[1,2,3],[4,5,6],[7,8,9]]           # lista de listas
```

Listas

Indexable: Sus elementos se obtienen indicando su índice.

Listas

Indexable: Sus elementos se obtienen indicando su índice.

```
1 l = [3,4,2,4,9,6]   # lista de números
2 print(l[3])         # >>> 4
3 print(l[-2])        # >>> 9
```

Listas

Indexable: Sus elementos se obtienen indicando su índice.

```
1 l = [3,4,2,4,9,6]   # lista de números
2 print(l[3])         # >>> 4
3 print(l[-2])        # >>> 9
```

Mutable: Sus elementos pueden ser modificados.

Listas

Indexable: Sus elementos se obtienen indicando su índice.

```
1 l = [3,4,2,4,9,6]   # lista de números
2 print(l[3])         # >>> 4
3 print(l[-2])        # >>> 9
```

Mutable: Sus elementos pueden ser modificados.

```
1 l = [3,4,2,4,9,6]   # lista de números
2 l[0] = 6
3 print(l)            # >>> [6,4,2,4,9,6]
```

Listas

4 operadores básicos

Operador	Operación
$l_1 + l_2$	Concatena l_1 y l_2
n * l	Concatena n veces l
i **in** l	es True ssi $i \in l$
i **not in** l	es True ssi $i \notin l$

Listas

4 operadores básicos

Operador	Operación
$l_1 + l_2$	Concatena l_1 y l_2
n * l	Concatena n veces l
i **in** l	es True ssi $i \in l$
i **not in** l	es True ssi $i \notin l$

```
1 l1 = [1,2,3]
2 l2 = [4,5]
3
4 print(l1+l2)          # >>> [1,2,3,4,5]
5 print(3*l2)           # >>> [4,5,4,5,4,5]
6 print(2 in l1)        # >>> True
7 print(2 not in l2)    # >>> True
```

Listas: For

For: permite recorrer una lista (o string) y ejecutar código para cada elemento en ella.

Listas: For

For: permite recorrer una lista (o string) y ejecutar código para cada elemento en ella.

Sintaxis

for variable **in** lista (o string)**:**
    bloque_de_código_for

Listas: For

For: permite recorrer una lista (o string) y ejecutar código para cada elemento en ella.

Sintaxis

for variable **in** lista (o string)**:**
    bloque_de_código_for

```
1 l = ['fruta', 'carne', 'arroz', 'café']
2
3 # Recorre cada elemento de "l"
4 for e in l:
5   # Operamos con el "e" actual
6   print("No olvides comprar",e)
```

Listas: For

Range: Permite *simular* una listas creciente de números.

Listas: For

Range: Permite *simular* una listas creciente de números.

Sintaxis

lista = **range**(num_final)
lista = **range**(num_inicio, num_final)

Listas: For

Range: Permite *simular* una listas creciente de números.

Sintaxis

lista = **range**(num_final)
lista = **range**(num_inicio, num_final)

```
1 l1 = range(6)          # Equivale a [0,1,2,3,4,5]
2 l2 = range(4,6)        # Equivale a [4,5]
3 l3 = range(7,5)        # Equivale a []
4
5 print(l1)              # >>> range(0, 6)
6 print(l1[3])           # >>> 3
7 l1[3] = 5              # >>> Error!
```

Listas: For

¿Cuál es su gracia?

Listas: For

¿Cuál es su gracia?

Permite realizar loops con incremento automático.

Listas: For

¿Cuál es su gracia?

Permite realizar loops con incremento automático.

Ejemplo: Loop simple

```
1 # loop simple con i = 0..n-1
2 n = int(input("Ingrese número: "))
3 for i in range(6):
4   print(i)
```

Listas: For

¿Cuál es su gracia?

Permite realizar loops con incremento automático.

Ejemplo: Loop simple

```
# loop simple con i = 0..n-1
n = int(input("Ingrese número: "))
for i in range(6):
  print(i)
```

Ejemplo: Loops anidados

```
# loop doble
for i in range(1,11):
  print("-----------")
  for j in range(1,11):
    print(i,j)
```

Listas: For

Comandos **break** y **continue** también funcionan con *for*.

Listas: For

Comandos **break** y **continue** también funcionan con *for*.

```
1 for i in range(1000000):
2   if(i == 10):
3     break
4   else:
5     continue
6   print(i)
```

¿Qué muestra este código?

Listas: For (usos típicos)

Contador: Cuente los múltiplos de 7 entre 1 y 1000

Listas: For (usos típicos)

Contador: Cuente los múltiplos de 7 entre 1 y 1000

```
1 total = 0
2 for num in range(1,1001):
3   if(num % 7 == 0):
4     total += 1
5 print("Múltiplos de 7 entre 1 y 1000:", total)
```

Listas: For (usos típicos)

Acumulador: Sumar impares del 1 y 1000

Listas: For (usos típicos)

Acumulador: Sumar impares del 1 y 1000

```
1 suma = 0
2 for num in range(1,1001):
3   if(num % 2 == 1):
4     suma += num
5 print("La suma es", suma)
```

Listas: For (usos típicos)

Loops anidados: Permiten operar sobre matrices.

```
1  for i in range(3):
2    for j in range(3):
3      print("( i , j ) = (",i,",", j,")")
4
5  # >>> ( i , j ) = ( 0 , 0 )
6  # >>> ( i , j ) = ( 0 , 1 )
7  # >>> ( i , j ) = ( 0 , 2 )
8  # >>> ( i , j ) = ( 1 , 0 )
9  # >>> ( i , j ) = ( 1 , 1 )
10 # >>> ( i , j ) = ( 1 , 2 )
11 # >>> ( i , j ) = ( 2 , 0 )
12 # >>> ( i , j ) = ( 2 , 1 )
13 # >>> ( i , j ) = ( 2 , 2 )
```

Listas: For (usos típicos)

Ejemplo: Mostrar tablero SUDOKU.

```
1 def mostrar_tablero():
2   for i in range(9):
3     for j in range(9):
4       print(sudoku.obtener(i,j),end="")
5     print()
```

Listas: For (usos típicos)

Hablando del SUDOKU... ¿cómo representaba el tablero la librería `sudoku.py`?

Listas: For (usos típicos)

Hablando del SUDOKU... ¿cómo representaba el tablero la librería `sudoku.py`?

```
tablero=[
[0,0,0,0,0,6,0,0,8],
[0,0,5,4,0,2,7,0,6],
[8,7,6,9,5,3,4,0,0],
[0,4,8,2,0,0,0,7,3],
[0,0,0,8,0,5,2,0,0],
[0,2,0,7,3,0,6,0,0],
[0,8,0,6,4,7,3,0,0],
[0,0,4,3,2,0,8,6,0],
[0,6,3,0,9,8,0,2,4]]
```

Listas: For (usos típicos)

Hablando del SUDOKU... ¿cómo representaba el tablero la librería `sudoku.py`?

```
tablero=[
[0,0,0,0,0,6,0,0,8],
[0,0,5,4,0,2,7,0,6],
[8,7,6,9,5,3,4,0,0],
[0,4,8,2,0,0,0,7,3],
[0,0,0,8,0,5,2,0,0],
[0,2,0,7,3,0,6,0,0],
[0,8,0,6,4,7,3,0,0],
[0,0,4,3,2,0,8,6,0],
[0,6,3,0,9,8,0,2,4]]
```

¿Cómo obtengo el valor en la casilla (3,5)?

Listas: For (usos típicos)

Hablando del SUDOKU... ¿cómo representaba el tablero la librería `sudoku.py`?

```
1  tablero=[
2  [0,0,0,0,0,6,0,0,8],
3  [0,0,5,4,0,2,7,0,6],
4  [8,7,6,9,5,3,4,0,0],
5  [0,4,8,2,0,0,0,7,3],
6  [0,0,0,8,0,5,2,0,0],
7  [0,2,0,7,3,0,6,0,0],
8  [0,8,0,6,4,7,3,0,0],
9  [0,0,4,3,2,0,8,6,0],
10 [0,6,3,0,9,8,0,2,4]]
```

¿Cómo obtengo el valor en la casilla (3,5)?

```
12 tablero[3][5]
```

Listas: For (usos típicos)

Loops anidados: Tabla del 1 al 10

Listas: For (usos típicos)

Loops anidados: Tabla del 1 al 10

```
1 for i in range(1,11):
2   print("------------------")
3   for j in range(1,11):
4     print(i,"x",j,"=",i*j)
```

Listas: For vs While

¿Cuál es la diferencia entre un *for* y un *while*?

Listas: For vs While

¿Cuál es la diferencia entre un *for* y un *while*?

- Ambos realizan la misma tarea.
- **while**: usado para loops cuya finalización depende de que se cumpla una condición.
- **for**: usado para iterar sobre listas o ejecutar loops un número determinado de veces.

Listas: For vs While

Con while...

```
1 n = int(input("ingrese número: "))
2 i = 0
3 while(i < n):
4   print(i)
5   i += 1
```

Con for...

```
1 n = int(input("ingrese número: "))
2 for i in range(n):
3   print(i)
```

Listas: For vs While

Con while...

```
1 i = 1
2 while(i < 11):
3   print("------------")
4   j = 1
5   while(j < 11):
6     print(i,"x",j,"=",i*j)
7     j += 1
8   i += 1
```

Con for...

```
1 for i in range(1,11):
2   print("------------")
3   for j in range(1,11):
4     print(i,"x",j,"=",i*j)
```

Listas: Funciones principales

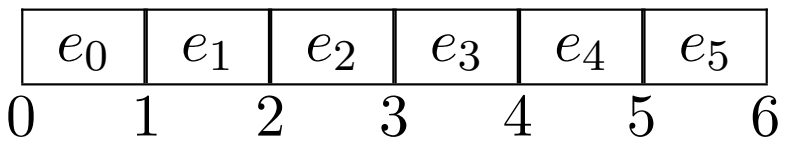
Sublistas: Para obtener una sublista se usa **l[i:j:k]**.

Listas: Funciones principales

Sublistas: Para obtener una sublista se usa **l[i:j:k]**.

Sintaxis

Lista[i:j] → retorna la sublista entre i y j.

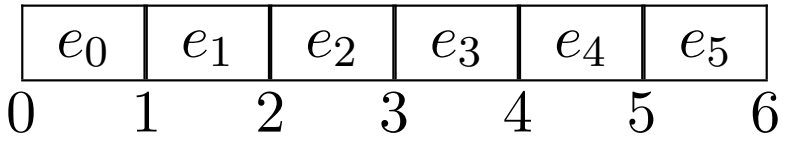


Listas: Funciones principales

Sublistas: Para obtener una sublista se usa **l[i:j:k]**.

Sintaxis

Lista[i:j] → retorna la sublista entre i y j.




```
1 l = [3,4,2,4,9,6]   # lista de números
2 print(l[1:4])       # >>> [4,2,4]
3 print(l[3:])        # >>> [4,9,6]
4 print(l[:2])        # >>> [3,4]
```

Listas: Funciones principales

También es posible agregar un tercer parámetro para dar saltos.

```
1 l = [3,4,2,4,9,6]   # lista de números
2 print(l[::2])       # >>> [3, 2, 9]
3 print(l[1::2])      # >>> [4, 4, 6]
4 print(l[::-1])      # >>> [6, 9, 4, 2, 4, 3]
```

Listas: Funciones principales

len(l): Retorna el tamaño de la lista.

sum(l): Retorna la suma de los elementos de **l**.

max(l): Retorna el máximo en **l**.

min(l): Retorna el mínimo en **l**.

Listas: Funciones principales

len(l): Retorna el tamaño de la lista.

sum(l): Retorna la suma de los elementos de **l**.

max(l): Retorna el máximo en **l**.

min(l): Retorna el mínimo en **l**.

```
1 l = [3,4,2,4,9,6]  # lista de números
2
3 print(len(l))      # >>> 6
4 print(max(l))      # >>> 9
5 print(min(l))      # >>> 2
6 print(sum(l))      # >>> 28
```

Listas: Funciones principales

l_1==l_2: Retorna **True** ssi los elementos de l_1 y l_2 son iguales y aparecen en el mismo orden.

l.count(x): Retorna el número de ocurrencias de **x** en **l**.

Listas: Funciones principales

l_1==l_2: Retorna **True** ssi los elementos de l_1 y l_2 son iguales y aparecen en el mismo orden.

l.count(x): Retorna el número de ocurrencias de **x** en **l**.

```
1 l1 = [3,4,2,4,9,6]   # lista de números 1
2 l2 = [3,4,2,4,9,6]   # lista de números 2
3 l3 = [3,2,4,4,9,6]   # lista de números 3
4
5 print(l1 == l2)      # >>> True
6 print(l1 == l3)      # >>> False
7 print(l1.count(4))   # >>> 2
8 print(l1.count(8))   # >>> 0
```

Listas: Funciones principales

l.append(x): Agrega **x** al final de **l**.

l_1**.extend(**l_2**)**: Agrega elementos de l_2 a l_1.

Listas: Funciones principales

l.append(x): Agrega **x** al final de **l**.

l_1**.extend(**l_2**)**: Agrega elementos de l_2 a l_1.

```
1 l1 = [1,2,3]     # lista de números 1
2 l2 = [5,6]       # lista de números 2
3
4 l1.append(4)
5 print(l1)        # >>> [1,2,3,4]
6 l1.extend(l2)
7 print(l1)        # >>> [1,2,3,4,5,6]
```

Listas: Funciones principales

Observación: $l_1 + l_2 \neq l_1.append(l_2) \neq l_1.extend(l_2)$

Listas: Funciones principales

Observación: $l_1 + l_2 \neq l_1.append(l_2) \neq l_1.extend(l_2)$

- $l_1 + l_2$ retorna la concatenación de l_1 con l_2.
- $l_1.append(l_2)$ agrega l_2 a l_1.
- $l_1.extend(l_2)$ agrega los elementos de l_2 a l_1.

Listas: Funciones principales

Observación: $l_1 + l_2 \neq l_1.append(l_2) \neq l_1.extend(l_2)$

- $l_1 + l_2$ retorna la concatenación de l_1 con l_2.
- $l_1.append(l_2)$ agrega l_2 a l_1.
- $l_1.extend(l_2)$ agrega los elementos de l_2 a l_1.

```
1 l1 = [1,2,3]    # lista de números 1
2 l2 = [4,5,6]    # lista de números 2
3
4 print(l1+l2)    # >>> [1,2,3,4,5,6]
5 l1.extend(l2)
6 print(l1)       # >>> [1,2,3,4,5,6]
7 l1.append(l2)
8 print(l1)       # >>> [1,2,3,4,5,6,[4,5,6]]
```

Listas: Funciones principales

l.index(x): Retorna el id de la primera ocurrencia de **x** en **l** (si $\mathbf{x} \notin \mathbf{l} \rightarrow$ **Error**).

l.insert(i,x): Inserta **x** en **l[i]** (si $\mathbf{i} \geq$ **len(l)** lo agrega al final).

Listas: Funciones principales

l.index(x): Retorna el id de la primera ocurrencia de **x** en **l** (si $\mathbf{x} \notin \mathbf{l} \rightarrow$ **Error**).

l.insert(i,x): Inserta **x** en **l[i]** (si $\mathbf{i} \geq$ **len(l)** lo agrega al final).

```
1 l = [3,2,4,4,9,6]      # lista de números
2
3 print(l.index(4))      # >>> 2
4 l.insert(1,4)
5 print(l)               # >>> [3,4,2,4,4,9,6]
6 print(l.index(4))      # >>> 1
```

Listas: Funciones principales

l.index(x): Retorna el id de la primera ocurrencia de **x** en **l** (si **x** $\notin$ **l** $\rightarrow$ **Error**).

l.insert(i,x): Inserta **x** en **l[i]** (si **i** $\geq$ **len(l)** lo agrega al final).

```
1 l = [3,2,4,4,9,6]      # lista de números
2
3 print(l.index(4))      # >>> 2
4 l.insert(1,4)
5 print(l)               # >>> [3,4,2,4,4,9,6]
6 print(l.index(4))      # >>> 1
```

Importante: $(l.insert(i,x)) \neq (l[i] = x)$

Listas: Funciones principales

l.remove(x): Elimina de **l** la primera ocurrencia de **x** (si **x** $\notin$ **l** $\rightarrow$ **Error**).

del l[i]: Elimina elemento **l[i]** de **l** (si **i** $\geq$ **len(l)** $\rightarrow$ **Error**).

Listas: Funciones principales

l.remove(x): Elimina de **l** la primera ocurrencia de **x** (si **x** $\notin$ **l** $\rightarrow$ **Error**).

del l[i]: Elimina elemento **l[i]** de **l** (si **i** $\geq$ **len(l)** $\rightarrow$ **Error**).

```
1 l = ["ola","k","ase","?"]    # lista de números
2
3 l.remove("k")
4 print(l)                     # >>> ['ola','ase','?']
5 del l[1]
6 print(l)                     # >>> ['ola','?']
```

Listas: Funciones principales

l.reverse(): Invierte **l**.

l.sort(): Ordena los valores de **l** (si **l** es una lista mixta, primero ordena por tipo y luego por valor).

Listas: Funciones principales

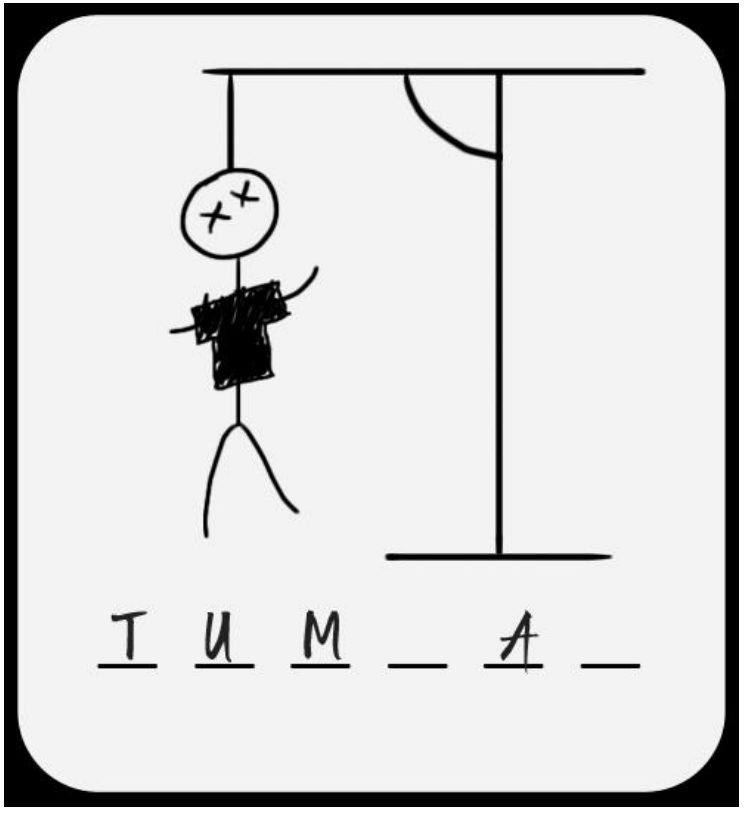
l.reverse(): Invierte **l**.

l.sort(): Ordena los valores de **l** (si **l** es una lista mixta, primero ordena por tipo y luego por valor).

```
1 l = [3,4,2,4,9,6]   # lista de números
2
3 l.sort()
4 print(l)            # >>> [2, 3, 4, 4, 6, 9]
5 l.reverse()
6 print(l)            # >>> [9, 6, 4, 4, 3, 2]
```

Ejemplos

"Programe el colgado."



Ejemplos

Para dibujar el colgado usamos una lista de string con sus distintas fases (desde *colgado[0]* hasta *colgado[7]*).

```
4  # Retorna lista con los distintos dibujos del colgado
5  def obtener_colgado():
6    colgado = ["     ____\n    |     |\n          |\n          |\n          |\n
               -|-",
7              "     ____\n    |     |\n    o     |\n          |\n          |\n
               -|-",
8              "     ____\n    |     |\n    o     |\n     |    |\n          |\n
               -|-",
9              "     ____\n    |     |\n    o     |\n    /|    |\n          |\n
               -|-",
10             "     ____\n    |     |\n    o     |\n    /|\   |\n          |\n
               -|-",
11             "     ____\n    |     |\n    o     |\n    /|\   |\n    /     |\n
               -|-",
12             "     ____\n    |     |\n    o     |\n    /|\   |\n    / \   |\n
               -|-"]
13   return colgado
```

Ejemplos

Para pedir una palabra secreta usamos la librería *getpass*.

```
15 # Retorna la palabra secreta que ingresa uno de los
       usuarios (en minúscula)
16 def obtener_palabra_secreta():
17   return getpass.getpass("Ingrese palabra: ").lower()
```

Ejemplos

Para pedir una palabra secreta usamos la librería *getpass*.

```
15 # Retorna la palabra secreta que ingresa uno de los
       usuarios (en minúscula)
16 def obtener_palabra_secreta():
17   return getpass.getpass("Ingrese palabra: ").lower()
```

Para enmascarar la palabra con '_' creamos una función.

```
19 # Recibe una palabra y la retorna en guiones bajos
20 def enmascara_palabra(p):
21   ret = ""
22   for c in p:
23     ret += "_"
24   return ret
```

Ejemplos

Al iniciar el juego llamamos a las funciones y creamos una lista vacía para guardar los caracteres que se digan.

Ejemplos

Al iniciar el juego llamamos a las funciones y creamos una lista vacía para guardar los caracteres que se digan.

```
40 # Función que inicia el juego
41 def juego():
42
43   # Obtengo la lista de dibujos del colgado
44   colgado = obtener_colgado()
45   # Pida la palabra secreta al usuario
46   palabra = obtener_palabra_secreta()
47   # Enmascado palabra secreta
48   actual = enmascara_palabra(palabra)
49
50   # Creo una lista vacía para las letras ya dichas
51   dichas = []
```

Ejemplos

En loop principal:

- Mostramos el colgado.
- Pedimos input a usuario.
- Realizamos jugada.
- Agregamos letra a lista *dichas*.
- Verificamos si el juego acabó.

Ejemplos

```
53    id = 0
54    while(id < len(colgado) - 1):
55
56      mostrar_colgado(colgado,id,actual,dichas)
57      c = input("Ingrese caracter: ").lower()[0]
58      nuevo = jugada(actual, palabra, c)
59
60      # veo si algo cambió
61      if(nuevo != actual): actual = nuevo
62      else: id += 1
63
64      # Agrego 'c' a la lista de caracters dichos
65      if(c not in dichas):
66        dichas += [c]
67        # las ordeno
68        dichas.sort()
69
70      # veo si ganó
71      if("_" not in actual): break
```

Ejemplos

Función para mostrar al colgado:

```
26 # Muestra el colgado según el id actual
27 def mostrar_colgado(colgado,id,actual,dichas):
28   print(colgado[id])
29   print("Actual:", actual)
30   print("Dichas:", dichas)
```

Ejemplos

Función para mostrar al colgado:

```
26 # Muestra el colgado según el id actual
27 def mostrar_colgado(colgado,id,actual,dichas):
28   print(colgado[id])
29   print("Actual:", actual)
30   print("Dichas:", dichas)
```

Función para descubrir letra de palabra enmascarada:

```
32 # Maneja la jugada del caracter 'c'
33 def jugada(actual, palabra, c):
34   nuevo = ""
35   for i in range(len(palabra)):
36     if(palabra[i] == c): nuevo += palabra[i]
37     else: nuevo += actual[i]
38   return nuevo
```

Ejemplos

"Realice un programa que pida al usuario preguntas de selección múltiple, y luego las muestre para que otra persona las resuelva."

Ejemplos

Guardaremos las preguntas en una lista.

```
1 preguntas = []
```

Ejemplos

Guardaremos las preguntas en una lista.

```
1 preguntas = []
```

Cada elementos de la lista será una pregunta $[x,y,z]$, donde:

- x es un *str* con la pregunta.
- y es una lista de *strs* con las alternativas.
- z es el id de la respuesta correcta.

Ejemplos

Guardaremos las preguntas en una lista.

```
1 preguntas = []
```

Cada elementos de la lista será una pregunta $[x,y,z]$, donde:

- x es un *str* con la pregunta.
- y es una lista de *strs* con las alternativas.
- z es el id de la respuesta correcta.

Ejemplo:

```
1 preguntas = [['pregunta 1',['alt1a','alt1b'],1],
2              ['pregunta 2',['alt2a','alt2b'],2],
3              ['pregunta 3',['alt3a','alt3b'],1]]
```

Ejemplos

Para pedir preguntas entramos a un loop:

- Pedimos pregunta y la guardamos en *pregunta*.
- Pedimos alternativas y las guardamos en una lista *alternativas*.
- Pedimos alternativa correcta y la guardamos en *ok*.
- Realizamos: preguntas.append([*pregunta*, *alternativas*,*ok*]).

Ejemplos

```
3  # obtengo las preguntas
4  while(True):
5
6    # pido siguiente pregunta
7    pregunta = input("\nIngrese una nueva pregunta: ")
8    if(len(pregunta) == 0):
9      break
10
11   # pido las alternativas
12   print()
13   alternativas = []
14   while(True):
15     a = input("Ingrese alternativa: ")
16     if(len(a) == 0):
17       break
18     alternativas.append(a)
```

(continua...)

Ejemplos

```
19
20    # pido la alternativa correcta
21    for i in range(len(alternativas)):
22      print(str(i+1)+"-",alternativas[i])
23    ok = int(input("\nIngrese número de la alternativa
        correcta:"))
24
25    # Agrego la pregunta al set de preguntas
26    preguntas.append([pregunta, alternativas,ok])
```

Ejemplos

Realizar preguntas recorro la lista *preguntas*:

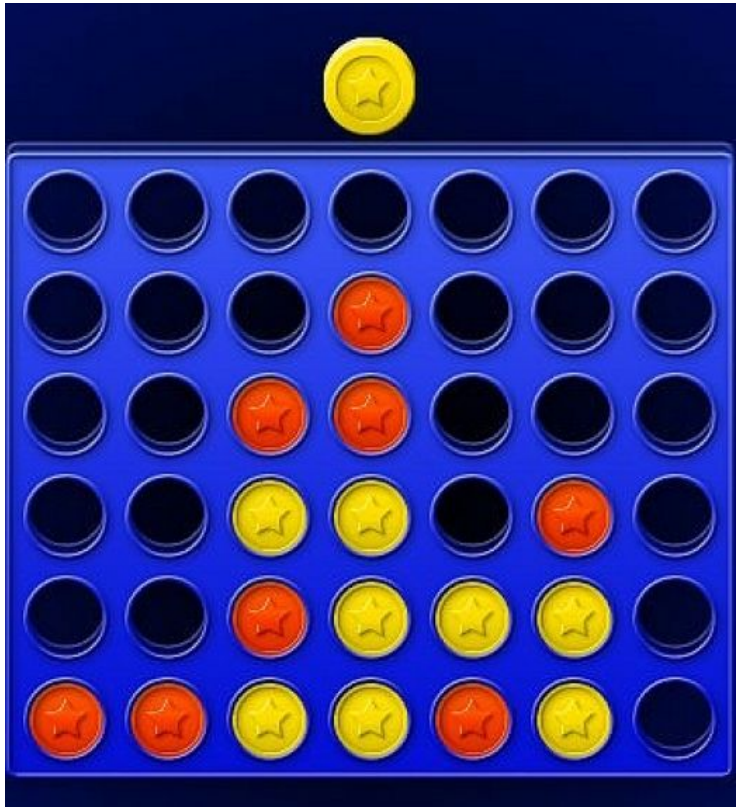
- Muestro la pregunta.
- Muestro alternativas.
- Pido respuesta.
- Verifico si respuesta fue correcta o no.

Ejemplos

```
28 # Realizo las preguntas
29 correctas = 0
30 for i in range(len(preguntas)):
31   print("\n----------------------")
32   print("Pregunta",str(i+1)+":",preguntas[i][0],"\n")
33
34   for j in range(len(preguntas[i][1])):
35     print(str(j+1)+"-",preguntas[i][1][j])
36
37   if(int(input("\nRespuesta? ")) == preguntas[i][2]):
38     correctas += 1
39
40 # Muestro el puntaje
41 print("\n" + str(correctas) + " de " + str(len(
       preguntas)) + " correctas")
```

Ejemplos

"Programe el juego conocido como connect-4 para un tablero de $N \times M$. Considere sólo líneas horizontales y verticales para ganar"



Ejemplos

Necesitamos un tablero de $M \times N$ que creamos con una lista de listas.

```
12 def iniciar_tablero(M,N):
13   tablero = []
14   for i in range(M):
15     fila = []
16     for j in range(N):
17       fila.append(0)
18     tablero.append(fila)
19   return tablero
```

Ejemplos

Para mostrar este tablero:

```
21 def mostrar_tablero(tablero):
22   print()
23   for i in range(len(tablero)):
24     for j in range(len(tablero[i])):
25       print(tablero[i][j],end=' ')
26     print()
27
28   print()
29   for i in range(len(tablero[0])):
30     print(i,end=' ')
31   print()
```

Ejemplos

Al iniciar el juego creamos un nuevo tablero y definimos una variable *turno* que recuerde quién tiene el turno actual.

```
64 def jugar():
65   # Creo Tablero
66   M = 6
67   N = 7
68   tablero = iniciar_tablero(M,N)
69
70   # Para dar color al tablero
71   amarillo = '\033[93m'
72   rojo = '\033[91m'; fin_color = '\033[0m'
73
74   x = amarillo + "O" + fin_color;
75   o = rojo + "O" + fin_color;
76   turno = x
```

Ejemplos

Loop:

- Pido jugada.
- Agrego jugada al tablero.
- Verifico si existe ganador.
- Verifico si es empate.
- Cambio de turno.

Ejemplos

```
78    while(True):
79      mostrar_tablero(tablero)
80      pos = int(input("Juega " + turno + ": "))
81      if(pos >= N): continue
82
83      # Agrego jugada
84      for i in range(M):
85        if(tablero[M-1-i][pos] == 0):
86          tablero[M-1-i][pos] = turno
87          break
```

(continua...)

Ejemplos

```
89      # Verifico si turno ganó
90      if(es_ganador(tablero)):
91        print("\nGanó",turno)
92        break
93
94      # Verifico si es empate
95      if(fin_juego(tablero)):
96        print("\nEmpate!")
97        break
98
99      # Cambio de turno
100     if(turno == x): turno = o
101     else: turno = x
```

Ejemplos

Para verificar si es un empate basta con encontrar algún 0 en el tablero.

Ejemplos

Para verificar si es un empate basta con encontrar algún 0 en el tablero.

```
54 def fin_juego(tablero):
55   # reviso si quedan movimientos posibles
56   # Recorro filas
57   for i in range(len(tablero)):
58     # Recorro columnas
59     for j in range(len(tablero[0])):
60       if(tablero[i][j] == 0):
61         return False
62   return True
```

Ejemplos

Para verificar si hay un ganador debemos chequear cada horizontal y vertical de largo 4 en el tablero.

Ejemplos

Para verificar si hay un ganador debemos chequear cada horizontal y vertical de largo 4 en el tablero.

Estrategia: Recorro cada casilla del tablero. Por cada una de ellas:

- Veo si las 3 casillas a la derecha son iguales a la casilla actual (y distintas de cero).
- Veo si las 3 casillas inferiores son iguales a la casilla actual (y distintas de cero).

Ejemplos

0	0	0	0	0
0	0	0	0	0
O	**X**	0	0	0
X	**O**	**O**	**O**	0
O	**X**	**X**	**X**	**X**

Ejemplos

```
39 def es_ganador(tablero):
40   # Busco 4 fichas iguales en sentido vertical y
       horizontal
41
42   # Recorro filas
43   for i in range(len(tablero)):
44     # Recorro columnas
45     for j in range(len(tablero[0])):
46       if(check_columna(tablero,i,j)):
47         return True
48       if(check_fila(tablero,i,j)):
49         return True
50
51   # Si llegué aquí es porque no habían 4 conectadas
52   return False
```

Ejemplos

```
33 def check_columna(tablero,i,j):
34   return j+3 < len(tablero[0]) and tablero[i][j] ==
       tablero[i][j+1] == tablero[i][j+2] == tablero[i][j
       +3] != 0
35
36 def check_fila(tablero,i,j):
37   return i+3 < len(tablero) and tablero[i][j] ==
       tablero[i+1][j] == tablero[i+2][j] == tablero[i
       +3][j] != 0
```

Ejercicios

1) Defina la matriz [[1,4,3],[5,2,4],[8,5,1]]. Para ella muestre su transpuesta y calcule:

- Suma de sus elementos.
- Multiplicación de su diagonal

2) Programe los ejemplos vistos en la clase.

3) Agregue a *connect-4* chequeo de diagonales.

4) Realice un programa para mantener una lista de compras para el supermercado. El programa debe permitir agregar productos a la lista, eliminar productos, ver la lista y ordenarla alfabéticamente.