Clase 13 – Ejercicios Listas

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

15-10-2019

Resumen de la clase

Podemos definir una lista de listas de la siguiente manera:

```
matriz =

 [[“a”,”b”,”c”],[“d”,”e”,”f”],[“g”,”h”,”i”]]
```

La variable matriz es una lista que tiene 3 elementos. Estos tres elementos son listas. Cada una de estas listas tiene a su vez tres elementos:
El primer elemento es una lista que contiene los elementos a,b, y c.
El segundo elemento es una lista que contiene los elementos d,e, y f.
El tercer elemento es una lista que contiene los elementos g,h, y i.

	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i

De forma general, podemos obtener un elemento de una variable `M` que representa a una lista de listas de la siguiente manera:

```
M[i][j]
```

Donde `i` y `j` representa a una fila y a una columna específica de la lista de listas (recordemos que las filas y columnas se empiezan a contar desde 0).

Resumen de la clase

¿Podemos crear una lista de `n x m` con ciertos valores en cada celda?
Digamos que el valor que queremos que esté en cada celda es k. Para hacerlo, podemos ocupar el siguiente comando:

```
M = [[k for x in range(m)] for x in range(n)]
```

De forma general, podemos modificar un elemento de una variable M que representa a una lista de listas de la siguiente manera:

```
M[i][j] = k
```

Donde `i` y `j` representa a una fila y a una columna específica de la lista de listas (recordemos que las filas y columnas se empiezan a contar desde 0). k representa el nuevo elemento que modificará lo que estaba en la fila `i` y columna `j`.

Imaginemos que recibimos una matriz `M` ya creada. No obstante, no sabemos a priori cuántas filas o columnas tiene. En forma general, si tenemos una matriz `M` con la misma cantidad de columnas por fila:

Para poder saber el número de filas: `len(M)`
Para poder saber el número de columnas: `len(M[0])`

Ejercicio P1 IIC1103 2018-2

Pregunta 1

Estás realizando tu práctica en `GeoTree`, un emprendimiento UC que usa Python para armar árboles genealógicos. Sin embargo, están teniendo problemas en la implementación, por lo que te piden a ti, flamante alumno de Introducción a la Programación, que los ayudes a terminar su programa.

Hasta ahora, lo único que tienen es una representación con *listas* del árbol de una familia, y la declaración de las funciones que vas a necesitar. El árbol está representado por una lista donde, en cada elemento se guarda un *string* con el nombre del miembro de la familia (no hay *strings* repetidos), seguido de una *lista* de las posiciones de sus hijos dentro de la lista. Puedes ver un ejemplo de la lista en el código más abajo. Las funciones que necesitan que implementes son las siguientes:

(a) **(30 puntos)** `hijos(a,n)`: recibe una *lista* `a` con el árbol genealógico y un *string* `n` con el nombre de un integrante de la familia. Retorna una *lista* de *strings* con los nombres de los hijos de ese integrante. Si el integrante no tiene hijos, retorna una *lista* vacía.

(b) **(30 puntos)** `padres(a,n)`: recibe una *lista* `a` con el árbol genealógico y un *string* `n` con el nombre de un integrante de la familia. Retorna una *lista* de *strings* con los nombres de los padres de ese integrante. Si el integrante no tiene padre, retorna una *lista* vacía.

A continuación, se encuentra el código con el ejemplo de la familia Simpson y el *output* esperado escrito en los comentarios. Tu deber es completar las funciones `hijos(a,n)` y `padres(a,n)`, para que los `print` del final del código impriman lo solicitado. Ten en cuenta que no es necesario que la lista de los nombres salgan exactamente en el orden del *output* esperado.

```python
def hijos(a, n):
    #Completar

def padres(a, n):
    #Completar

simp = [
    ["Abraham",[1,2]],     #0
    ["Herb",[]],           #1
    ["Homer",[3,4,5]],     #2
    ["Bart",[]],           #3
    ["Maggie",[]],         #4
    ["Lisa",[]],           #5
    ["Marge",[3,4,5]],     #6
    ["Mona",[2]],          #7
    ["Clancy",[6,10,11]], #8
    ["Jackie",[6,10,11]], #9
    ["Selma",[12]],        #10
    ["Patty",[]],          #11
    ["Ling",[]]            #12
    ]

print(hijos(simp,'Marge'))       #['Bart', 'Maggie', 'Lisa']
print(hijos(simp,'Clancy'))      #['Marge', 'Patty', 'Selma']
print(hijos(simp, 'Maggie'))     #[]
print(padres(simp, 'Maggie'))    #['Homer', 'Marge']
print(padres(simp, 'Homer'))     #['Abraham', 'Mona']
print(padres(simp, 'Ling'))      #['Selma']
print(padres(simp, 'Abraham'))   #[]
```

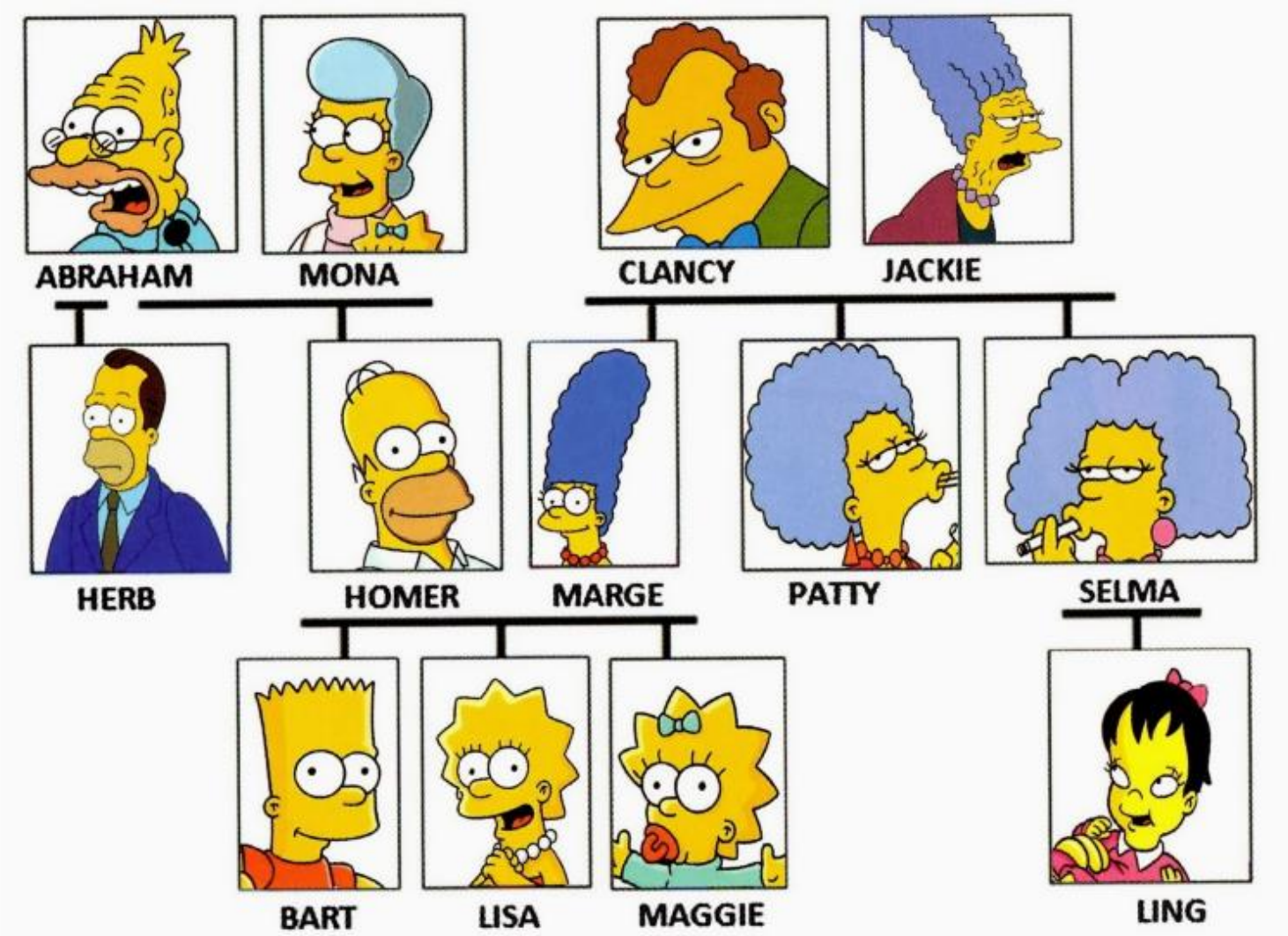

Figura 1: Árbol genealógico de Los Simpson

Ejercicio P1 IIC1103 2018-2

```
def hijos(a,n):
  hijos = []
  for i in a:
    if i[0] == n:
      for j in i[1]:
        hijos.append(a[j][0])
  return hijos
```

Ejercicio P1 IIC1103 2018-2

```python
def padres(a,n):
  pos_hijo = -1
  for i in range(len(a)):
    if a[i][0] == n:
      pos_hijo = i

  padres = []
  for j in a:
    if pos_hijo in j[1]:
      padres.append(j[0])

  return padres
```

Ejercicio P3 IIC1103 2019-1

Pregunta 3

Con todos los compromisos sociales, carretes, y encuentros para estudiar, organizar tu fin de semana es un infierno. Así es que decides crear tu propio sistema para gestionarlo. Para ello, decides tener un sistema de bloques para agendar actividades. El fin de semana constará de 24 bloques numerados del 0 al 23, y cada bloque tiene una duración de dos horas. Considerando esto, se concluye que el sábado tiene los bloques del 0 al 11, y el domingo tiene los bloques del 12 al 23. Para llevar un bloque a su respectiva hora y día, se tiene que el bloque 0 es el sábado entre 00:00 y 1:59 hrs, bloque 1 es el sábado entre 02:00 y 3:59 hrs, y así sucesivamente, luego el bloque 12 es el domingo entre 00:00 y 1:59, el bloque 13 es el domingo entre 02:00 y 3:59, y así sucesivamente.

Se te pide implementar un programa utilizando listas que permita hacer lo siguiente:

a) Cree un nuevo fin de semana vacío.

b) Pregunte por la cantidad de compromisos que se quieren incluir.

c) Agregue cada uno de los compromisos a los bloques correspondientes si es que es posible.

d) Cada vez que se intente agregar una actividad, se imprima el calendario para cada uno de los dos días por separado.

e) Al final debe mostrar el itinerario final del día sábado y domingo por separado, y una lista de todas las actividades que no pudieron ser agregadas.

Para esto, al momento de pedir los compromisos, se le debe pedir al usuario el nombre del compromiso, en qué bloque comienza y la duración de dicho compromiso en bloques. Un compromiso no cabe en el itinerario si es que cualquiera de los bloques necesarios para realizar dicho compromiso ya está ocupado por otra actividad ingresada anteriormente. Los bloques de un compromiso deben ser continuos y no tener saltos en el tiempo, por lo que si se tiene un compromiso que comienza en el bloque 12 y dura 4 bloques, para poder agendarlo deben estar disponible los bloques 12, 13, 14 y 15. A continuación, se muestra un ejemplo del flujo del programa. Los caracteres en negrita son los datos que son ingresados por el usuario:

```
¿Cuántos compromisos desea intentar agregar? 7
Nombre compromiso 1: Ir al cine
Duración compromiso 1: 2
Bloque de inicio compromiso 1: 8
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "", ""]
dgo = ["", "", "", "", "", "", "", "", "", "", "", ""]

Nombre compromiso 2: Dormir
Duración compromiso 2: 7
Bloque de inicio compromiso 2: 10
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "", "", "", "", "", ""]

Nombre compromiso 3: Almuerzo familiar
Duración compromiso 3: 1
Bloque de inicio compromiso 3: 18
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "Almuerzo familiar", "", "", "", "", ""]

Nombre compromiso 4: Salir con amigos
Duración compromiso 4: 2
Bloque de inicio compromiso 4: 9
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "Almuerzo familiar", "", "", "", "", ""]

Nombre compromiso 5: Ver documental de Game of Thrones
Duración compromiso 5: 1
Bloque de inicio compromiso 1: 22
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "Almuerzo familiar",
"", "", "", "Ver documental de Game of Thrones", ""]
```

Nombre compromiso 6: **Estudiar para IIC1103**
Duración compromiso 6: **2**
Bloque de inicio compromiso 6: **19**

```
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "Almuerzo familiar",
"Estudiar para IIC1103", "Estudiar para IIC1103", "",  "Ver documental de Game of Thrones", ""]
```

Nombre compromiso 7: **Descansar**
Duración compromiso 7: **6**
Bloque de inicio compromiso 7: **23**

```
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "Almuerzo familiar",
"Estudiar para IIC1103", "Estudiar para IIC1103", "",  "Ver documental de Game of Thrones", ""]
```

```
Los siguientes compromisos no podrán realizarse:
- Salir con amigos
- Descansar

Tu programación del fin de semana es:
Sabado:
00 hrs: ""
02 hrs: ""
04 hrs: ""
06 hrs: ""
08 hrs: ""
10 hrs: ""
12 hrs: ""
14 hrs: ""
16 hrs: "Ir al cine"
18 hrs: "Ir al cine"
20 hrs: "Dormir"
22 hrs: "Dormir"

Domingo:
00 hrs: "Dormir"
02 hrs: "Dormir"
04 hrs: "Dormir"
06 hrs: "Dormir"
08 hrs: "Dormir"
10 hrs: ""
12 hrs: "Almuerzo familiar"
14 hrs: "Estudiar para IIC1103"
16 hrs: "Estudiar para IIC1103"
18 hrs: ""
20 hrs: "Ver documental de Game of Thrones"
22 hrs: ""
```

Nota que las actividades se pueden realizar ocupando bloques del sábado y domingo (en el ejemplo, dormir), pero no se pueden agendar desde domingo hasta el lunes (en el ejemplo, descansar). Por otra parte, nota que luego de cada intento de ingreso, se imprime una lista con el itinerario del sábado y otra con el itinerario del domingo, por separado.

Ejercicio P1 IIC1103 2018-2

```
horario = [""]*24
compromisos_no_agregados = []

cant_compromisos = int(input("¿Cuántos compromisos desea intentar agregar?"))
```

Ejercicio P1 IIC1103 2018-2

```
for i in range(cant_compromisos):
  nombre = input("Nombre compromiso "+str(i+1)+":")
  duracion = int(input("Duración compromiso "+str(i+1)+":"))
  bloque_inicio = int(input("Bloque de inicio compromiso "+str(i+1)+":"))

  if horario[bloque_inicio:bloque_inicio+duracion] == [""]*duracion:
    horario[bloque_inicio:bloque_inicio+duracion] = [nombre]*duracion
  else:
    compromisos_no_agregados.append(nombre)

  print("sab =",horario[0:12])
  print("dgo =",horario[12:24])
```

Ejercicio P1 IIC1103 2018-2

```
print("Los siguientes compromisos no podrán realizarse:")
for i in compromisos_no_agregados:
  print("-",i)

print("Tu programación del fin de semana es:")
print("Sábado:")
for hora in range(0,12):
  print(hora,"hrs:",horario[hora])

print("Domingo:")
for hora in range(12,24):
  print(hora,"hrs:",horario[hora])
```

Links

- https://repl.it/@FelipeLopez/IIC1103EjerciciosListas que contiene todos los ejemplos de la clase.

Clase 13 – Ejercicios Listas

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

15-10-2019