UNIVERSIDAD DE ALCALÁ

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

LABORATORIO
FUNDAMENTOS DE LA PROGRAMACIÓN
1ª PARTE TEMA 3

3. TIPO ENUMERADO

Lista ordenada de identificadores.

SINTAXIS

#Llamada a libreria

from enum import Enum

 # Cuerpo del programa

Identif_conjunto = Enum (‘ Identif_conjunto ‘, ‘ identif1, identif2,…,identifn ‘)

```
#Llamada a libreria
from enum import Enum
# Cuerpo del programa
meses = Enum ('meses', ‘enero, febrero, marzo, abril, mayo, junio, julio, agosto, septiembre, octubre, noviembre, diciembre’)
for m in meses:
   print (m) #Muestra toda la variable
for m in meses:
   print (m.name) #Muestra nombres
for m in meses:
   print (m.value) #Muestra valores
```

Operaciones que admite:
Comparacion is, not is

```
if meses.enero   is meses.febrero:
        print('enero es menor que Febrero')
else:
        print ('no es')
```

de comparación == o ¡=:

```
print(meses.enero == 1)
print(meses.enero ¡= 1)
```

3.1 TIPO SUBRANGO

Declaran un intervalo de valores.

SINTAXIS

range(n) = range(0, n) = [0, 1, ... , n-1].

Nota: Para ver los valores de la lista creada con range(), es necesario convertirla a lista mediante el tipo list().
>>> range(3)
range(0, 3)
>>> list(range(3)) [0, 1, 2]

range(m, n) = [m, m+1, ... , n-1]
>>> range(5, 10) [5, 6, 7, 8, 9] >>> list(range(-5, 1)) [-5, -4, -3, -2, -1, 0]
Si n es menor o igual que m, se crea una lista vacía.
>>> list(range(5, 1)) [] >>> list(range(3, 3)) []

range(m, n, p) y crea una lista que empieza en m y acaba antes de llegar a n, avanzando de p en p
>>> list(range(5, 21, 3)) [5, 8, 11, 14, 17, 20] >>> list(range(10, 0, -2)) [10, 8, 6, 4, 2]

Resumen:

m: el valor inicial
n: el valor final (que no se alcanza nunca)
p: el paso (la cantidad que se avanza cada vez).
Si se escriben sólo dos argumentos, Python le asigna a p el valor 1.
range(m, n) es lo mismo que range(m, n, 1)
Si se escribe sólo un argumento, Python, le asigna a m el valor 0 y a p el valor 1.
range(n) es lo mismo que range(0, n, 1)

3.2. VECTORES Y MATRICES : ARRAY

Estructura en la que se almacena un número finito de datos del mismo tipo, teniendo importancia el orden en el que se sitúan los datos.

1. Se almacenan en posiciones contiguas de memoria.

2. Se asigna un único nombre de variable y se trabaja con cada elemento a través de las posiciones.

3. El acceso a cada elemento es directo.

4. Unidimensionales ≡ Vectores.
   Multidimensionales ≡ Matrices.

VECTORES : SINTAXIS

Para trabajar con arrays en Python necesitaremos:

1) Instalar librería **NUMPY**:

   http://www.cdlibre.org/consultar/catalogo/Python_Bibliotecas.html
   Desde aquí se accede a **NumPy 1.9.2 (py 3.4),** nos redirecciona a otra página para su ejecución.

2) Una vez instalada la librería, si necesitamos trabajar con arrays necesitaremos obligatoriamente :

   #1. Invocar a la librería en bloque declarativo del programa_
   import numpy **as** np

   #2. Inicializar vector en el cuerpo principal del programa:

   v = np.empty((Max,), dtype=np.int)

#Mediante esta función inicializo todas las componentes del vector CON BASURA.
#Max es una constante que especifica la dimensión máxima del vector.
#dtype detalla el tipo de datos del vector, en el ejemplo es un entero.

Los vectores pueden tener diferentes tipos de elementos, esto se especifica mediante atributo dtype:

- **dtype=np.int** #Especificación del tipo para crear vectores de enteros
- **dtype=np.float** #Especificación del tipo para crear vectores números reales
- **dtype=np.bool** #Especificación del tipo para crear vectores de tipo booleano
- **dtype=np.str** #Especificación del tipo para crear vectores de tipo carácter
- **dtype=np.object** #Especificación del tipo para crear vectores de tipo cadena de caracteres

Nota: También se podría inicializar a ceros pero daría errores si el tipo de datos no es numérico y si alguna componente del vector es cero.

v = np.zeros((Max,), dtype=np.int) #Mediante esta función inicializo a ceros solo vale para vectores numéricos

Ejemplos:

#Bloque declarativo

```
import numpy as np
Max =12
```
#Cuerpo principal

```
v = np.empty((Max,), dtype=np.float)
v2 = np.empty((Max,), dtype=np.float)
v[1] = float(input('Introduzca real que ocupa posición 1: '))
print(v[1])
v2[8] =v[1]
print(v2[8])
```

Ejemplo: Programa que lee una secuencia de 50 elementos como máximo y los imprime en orden inverso al de entrada.

#Bloque declarativo

```
import numpy as np
Maxi =5

def dim():
    while True:
        n = int(input(' Dime cuantos elementos vas a introducir en el vector'))
        if (n>0)and (n <Maxi+1):
            break
   return n

def leer (v, n):
    for i in range(n):
        v[i] = int(input('Introduzca el número: '))
    return v

def escribir (v,n):
    for i in range(n-1,-1,-1):
        print (v[i], end =" ,")
```

#Cuerpo principal

```
numeros = np.empty((Maxi,), dtype=np.int)
n = dim()
numeros = leer(numeros,n)
escribir (numeros,n)
```

Ordenación de vectores mediante los métodos de Burbuja e Insección:

El siguiente programa elige un método de ordenación entre el de la burbuja (opción 1,2 ó 3) y el de inserción (opción 4).

NOTA: La opción 3 del método de la burbuja es el más eficiente la 2ª el siguiente y el primero es el menos eficiente.

```
import numpy as np
Max = 15

def dim():
    '''()<- n devuelve la dimension del vector'''
    while True:
        n = int(input('Dimensión del vector: '))
        if (n>0) and (n < Max+1):
            break
    return n

def leer(v,n):
    '''v,n <-  v  Dado un vector vacio de dimensión n devuelve el vector relleno'''
    for i in range(n):
        v[i] = int(input('Introduzca número: '))
    return v

def escribir(v,n):
    '''v,n <-  Imprime por pantalla un vector de dimensión n'''
    for i in range(n):
       print(v[i])

def cambiar(a,b):
    '''a,b <-  a,b Intercambia los valores de a y b'''
    aux = a
    a = b
    b = aux
    return a,b

def ordenar_burbuja1(v,n):
    '''v,n <-  v  Dado un vector de dimensión n ordena sus componentes por el método de
    la burbuja'''
    for i in range(n-1):
        for j in range(n-1):
            if v[j] > v[j+1]:
               v[j],v[j+1] = cambiar(v[j],v[j+1])
    return v

def ordenar_burbuja2(v,n):
    '''v,n <-  v  Dado un vector de dimensión n ordena sus componentes por el método de
     la burbuja mejorado (2)'''
```

```
    for i in range(n-1):
        for j in range(n-i):
            if v[j] > v[j+1]:
                v[j],v[j+1] = cambiar(v[j],v[j+1])
    return v

def ordenar_burbuja3(v,n):
    '''v,n <-  v  Dado un vector de dimensión n ordena sus componentes por el método de
     la burbuja mejorando la eficiencia de los otros métodos de la burbuja'''
    b = False
    i = 1
    while (b == False) and (i<n):
        b =  True
        for j in range(n-i):
            if v[j] > v[j+1]:
                v[j],v[j+1] = cambiar(v[j],v[j+1])
                b = False
        i =i+1
    return v

def insercion(v,n):
    '''v,n <-  v  Dado n datos los va insertando en el vector v'''
    for i in range(n):
        aux = int(input('Dime dato'))
        j = i-1
        while (aux <v[j]) and (j>=0):
            v[j+1] = v[j]
            j = j-1
        v[j+1] = aux
    return v
def seguir():
    while True:
        s = input('Quieres salir?')
        if (s.upper() == 'SI') or (s.upper() == 'NO'):
                break
    return (s.upper() == 'SI')
def elige():
    ''' Procedimiento que lee un vector y lo ordena por los 4 métodos implementados
    anteriormente'''
    vector =np.empty((Max,),dtype =np.int)
    n = dim()
    while True:
        m = int(input('Elige opción ordenación Burbuja 1,2,3 ó inserción 4 : '))
        if m>0 and m<5:
            break
    if m ==1:
        vector = leer(vector,n)
        vector = ordenar_burbuja1(vector,n)
        print(' El vector ordenado es burbuja1: ')
        escribir(vector,n)
```

```
    elif m ==2:
        vector = leer(vector,n)
        vector = ordenar_burbuja2(vector,n)
        print(' El vector ordenado es burbuja2: ')
        escribir(vector,n)
    elif m ==3:
        vector = leer(vector,n)
        vector = ordenar_burbuja3(vector,n)
        print(' El vector ordenado es burbuja3: ')
        escribir(vector,n)
    else:
        vector =insercion(vector,n)
        escribir(vector,n)
#Cuerpo del programa
while True:
    elige()
    b =seguir()
    if b:
        break
```

MATRICES : SINTAXIS

Para trabajar con arrays bidimensionales (Matrices) en Python necesitaremos, lo mismo que en el caso de los vectores:

1) Instalar librería **NUMPY**:

2) Una vez instalada la librería, si necesitamos en un programa trabajar con matrices necesitaremos obligatoriamente escribir dos instrucciones:

#1. Invocar a la librería

import numpy **as** np

#2. Inicializar la matriz en el cuerpo principal del programa:

m = np.empty((Max,Max,), dtype=np.int)

#Mediante esta función inicializo todas las componentes de la matriz CON BASURA.
#(Max,Max,) constantes que especifican la dimensión máxima de la matriz.
#dtype me detalla el tipo de datos de la matriz.

Las matrices pueden tener diferentes tipos de elementos, esto se especifica mediante atributo dtype:

- **dtype=np.int** #Especificación del tipo para crear vectores de enteros
- **dtype=np.float** #Especificación del tipo para crear vectores números reales
- **dtype=np.bool** #Especificación del tipo para crear vectores de tipo booleano
- **dtype=np.str** #Especificación del tipo para crear vectores de tipo carácter
- **dtype=np.object** #Especificación del tipo para crear vectores de tipo cadena de caracteres

Nota: También se podría inicializar a ceros pero daría errores si el tipo de datos no es numérico y si alguna componente de la matriz es cero.

m = np.zeros((Max,Max,), dtype=np.int) #Mediante esta función inicializo a ceros solo vale para matrices de enteros

Ejemplo 1 .Inicializar variable tipo matriz de 3 filas y 4 columnas

```
#Bloque declarativo
import numpy as np
Maxf = 3
Maxc =4

#Cuerpo principal del programa
m = np.empty((Maxf,Maxc,), dtype=np.int)
```

Ejemplo 2

```
#Bloque declarativo
import numpy as np

Fil = range(8,16)  # Fijo rango de elementos como constante
Dias = ('l','m', 'x', 'j' ,'v', 's', 'd') # Fijo una tupla de elementos como constante

def leer(m,nf,nc):
    for i in range(nf):
        for j in range(nc):
            m[i,j] = int(input('Introduzca el número fila: '+ str(i+8) + ' columna: '+
            Dias[j]+ "\n" ))#Formato de salida  rango filas sumo 8 pq rango
                           #comienza en 8 columnas muestro datos de tuplas
    return m

def escribir(m,nf,nc):

    for i in range(nf):
        for j in range(nc):
            print(m[i,j],end=' ') # m[i,j] es igual que m[i][j]
        print()

#Cuerpo principal
m = np.empty((len(Fil),len(Dias),), dtype=np.float)
m = leer(m, len(Fil), len(Dias))
escribir( m, len(Fil), len(Dias))
```

	L	M	.....	V
8	50.8	60.1		
9	100	25.3		
10	65.3	52.3		
....				
15				26.3

ENUNCIADOS EJERCICIOS

TEMA3 PARTE1: TIPOS DEFINIDOS POR EL USUARIO

VECTORES Y MATRICES

TEMA3:

ENUNCIADOS

1. Escribir un programa en Pascal que rellene un array con los números pares comprendidos entre 1 y 10.

2. Escribir un programa en Pascal que rellene un array con los números comprendidos entre 25 y 35 divididos por 3.

3. Escribir un programa en Pascal que rellene un array con cinco números enteros consecutivos y haga una copia de ese array en otro.

4. Escribir un programa que pida al usuario n números enteros para almacenarlos en un array de dimensión mayor ó igual que n. Posteriormente, el programa deberá escribir dichos números en pantalla en el orden inverso al que se introdujeron, indicando cual es el menor y el número de apariciones de este así como el lugar (ó los lugares) que ocupaba en la lista inicial. (Ej. si la lista de enteros es: 2,-3,-3,4,5,2,6,7,8,9,-3,2,1,0,0 , el menor es -3 que ocupa las posiciones 2, 3 y 11).

5. Escribir un programa que lea una serie de letras mayúsculas del alfabeto inglés, comprobando que se introducen ordenadas, y cree con ellas un vector. Deben almacenarse en el vector sólo los datos correctos, es decir, letras mayúsculas del alfabeto inglés y en orden, rechazándose el resto. A continuación el programa pedirá una nueva letra y comprobará si está en el vector. Utilizar un valor centinela para el fin de datos. (Ej: Datos de entrada: A E D F T H U V Z $ → [A, E, F, T, U,V,Z])

6. Realizar el ejercicio anterior considerando letras mayúsculas del alfabeto castellano.

7. Escribir un programa que Lea los datos de un vector completo y los visualice, posteriormente gire los datos del vector n posiciones hacia la derecha, comenzando de nuevo por el principio si es necesario. Ejemplo: Dado V: [4,5,6,7,8,0] tras girar los datos 2 posiciones a la derecha se obtiene: [8,0,4,5,6,7]

8. Escribir un programa que lea los datos de una matriz y los visualice por filas, indicando el número de la fila correspondiente.
   Ejemplo: 1, 2, 3, 4
   5, 6, 7, 8
   9,10,11,12
   Resultado: Fila número 1: 1, 2, 3, 4
   Fila número 2: 5, 6, 7, 8
   .....

Solicitar un dato y recorrer la matriz para localizarlo en la misma indicando si el dato se ha localizado y, en caso afirmativo, devolver la primera posición en la que se encuentra (fila y columna).

9. Escribir un programa que defina los tipos de datos necesarios para leer las ventas en cinco días diferentes de los 10 empleados de una empresa y obtenga el total de ventas por empleado. Definir y utilizar un tipo de datos enumerado para los días de la semana.

10. Escribir un programa que lea una matriz cuadrada de orden menor o igual que 10, la visualice, busque sus elementos menor y mayor y la presenta en pantalla de nuevo con esos dos elementos intercambiados.

11. Escribir un programa que lea dos matrices de la pantalla de dimensiones NXM y MXR respectivamente, (N, M y R máximo 10) y obtenga la matriz producto de ambas. La salida del programa será cada una de las matrices de entrada y el producto obtenido.