Universidad Nacional de General Sarmiento

Introducción a la Programación

Listas

Universidad Nacional de General Sarmiento

Tipos de Variables aprendidas

```
x       = 10
pi      = 3.141592
calle   = "9 de Julio"
y       = x

aumento= sueldo * 0.19
z       = (2*a + c)* b**2
```

entera
decimal o flotante
cadena (caracteres)
depende del tipo de x

Fórmulas, tipo según resultado de fórmula (ej.entero o flotante)

```
x       = True
y       = False
w       = x>9
```

Variables lógicas o booleanas (valor de verdadero o falso)

Expresión lógica devuelve True/False (w=formato booleano)

Listas

Hasta ahora teníamos `enteros`, `floats`, `cadenas`, `booleanos` y desde hoy contamos con el tipo `lista`.
Las listas son secuencias ordenadas de valores.
Permiten guardar una cantidad ilimitada de elementos, accediéndolos por su posición.
Se escriben así:

```
[elemento0, elemento1,...,elementoN]
```

```
#por ejemplo
numeros = [15, -1, 13, 11]
animales = ['gato', 'perro', 'raton']
vacia = []
```

Usando listas

La utilidad de las listas está en sus elementos, pero ¿cómo manejamos esos elementos?

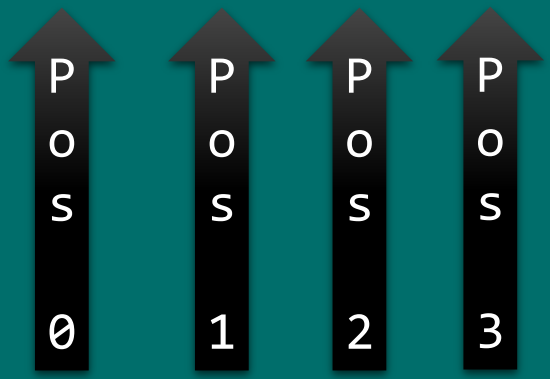
Lo hacemos por posición, indicando

```
nombreLista[posicion]
```

Sin olvidar que la primera posición es la 0

Por ejemplo:

```
numeros = [15, -1, 13, 11]
```




```
Lista con 4 elementos

Longitud de la lista?
Len(numeros)=4

1er elemento, pos=0
último elemento pos=3
```

Usando listas

La utilidad de las listas está en sus elementos, pero ¿cómo manejamos esos elementos?

Lo hacemos por posición, indicando

`nombreLista[posicion]`

Sin olvidar que la primera posición es la `0`

Por ejemplo:

```
numeros = [15, -1, 13, 11]
print(numeros[0]) # imprime 15
print(numeros[3]) # imprime 11
print(numeros[4]) # error
```

También podemos imprimir la lista entera

```
print(numeros)
```

```
[15, -1, 13, 11]
```

Usando listas (2)

Podemos cambiar el valor de sus elementos escribiendo:

```
nombreLista[posicion] = valor
```

Por ejemplo:

```
animales = ['gato', 'perro', 'raton']
animales[0] = 'raton'
print(animales)
#imprime: ['raton', 'perro', 'raton']
```

Podemos preguntar su tamaño

```
len(nombreLista)
longitud = len(animales)
print (longitud)   #imprime: 3
```

Usando listas (3)

Podemos agregar y quitar elementos:

```
#por ejemplo
animales = ['gato', 'perro', 'raton']
#con nombreLista.append(valor)  agregamos elementos
animales.append('pato')  lo agrega al final
print(animales)
#imprime :  ['gato', 'perro', 'raton', 'pato']
#con nombreLista.pop(posicion) quitamos elementos
animales.pop(1)
print(animales)
#imprime:  ['gato', 'raton', 'pato']
```

Usando listas (4)

Podemos unir varias listas:

```
lista1 + lista2
```

Por ejemplo...

```
animales = ['gato', 'perro', 'raton']
nombres = ['Tom', 'Spike', 'Jerry']
todo = animales + nombres
print(todo)
#imprime
```

```
['gato','perro','raton','Tom','Spike','Jerry']
```

Usando listas (5)

Podemos recorrerlas con while y con for:

```
animales = ['gato', 'perro', 'raton']
i = 0
while (i < len(animales)):
   print(animales[i])
   i = i + 1

for i in range(len(animales)):
   print (animales[i])

for elemento in animales:
    print (elemento)
```

Recorre por indice

Recorre por elemento

Usando listas (6)

Por ejemplo:

Hacer un programa que dado un número construye una lista de sus divisores propios

```
a=int(input("Ingrese un numero"))
divisoresPropios=[]
for i in range (1,a):
    if (a%i==0):
        divisoresPropios.append(i)
print(divisoresPropios)
```

(Pasarlo a una función!!!)

Ejercicios

1. Hacer una función que recibe una lista y un entero e indique si el entero está en la lista.
2. Hacer una función que dado un entero y una lista de enteros indique cuantas veces aparece el entero en la lista.
3. Hacer una función que reciba una lista de enteros y devuelva el máximo.
4. Hacer una función que reciba una lista y devuelva otra solo con los elementos sin repeticiones.