T03_Estructura_de_Datos

January 29, 2021

1 Python de cero a experto

Autor: Luis Miguel de la Cruz Salas

1.1 Pythonico es más bonito

1.1.1 Estructura de datos

Hay cuatro tipos de estructuras de datos, también conocidas como *colecciones* :

Tipo	Ordenada	Inmutable	Indexable	Duplicidad
List	SI	SI	SI	SI
Tuple	SI	NO	SI	SI
Sets	NO	SI	NO	NO
Dict	NO	SI	SI	NO

Cuando se selecciona un tipo de colección, es importante conocer sus propiedades para incrementar la eficiencia y/o la seguridad de los datos.

Listas

[1]:
```
gatos = ['Persa', 'Sphynx', 'Ragdoll','Siamés']
```

[2]:
```
origen = ['Irán', 'Toronto', 'California', 'Tailandia']
```

[3]:
```
pelo_largo = [True, False, True, True]
```

[4]:
```
pelo_corto = [False, False, False, True]
```

[5]:
```
peso_minimo = [2.3, 3.5, 5.4, 2.5]
peso_maximo = [6.8, 7.0, 9.1, 4.5]
```

[6]:
```
type(gatos)
```

[6]:
```
list
```

[7]:
```
print(id(gatos), id(gatos[0]))
```

```
139970037711936 139970037360112
```

[8]:
```
gatos
```

[8]:
```
['Persa', 'Sphynx', 'Ragdoll', 'Siamés']
```

Indexado

[9]:
```
gatos[-1] # Último elemento
```

[9]:
```
'Siamés'
```

[10]:
```
gatos[1:4] # Todos los elementos, menos el primero
```

[10]:
```
['Sphynx', 'Ragdoll', 'Siamés']
```

[11]:
```
gatos[::-1] # Todos los elementos en reversa
```

[11]:
```
['Siamés', 'Ragdoll', 'Sphynx', 'Persa']
```

[12]:
```
type(gatos[0])
```

[12]:
```
str
```

[13]:
```
id(gatos[0])
```

[13]:
```
139970037360112
```

Operaciones y Funciones sobre las listas

[14]:
```
len(gatos)
```

[14]:
```
4
```

[15]:
```
max(gatos)
```

[15]:
```
'Sphynx'
```

[16]:
```
min(gatos)
```

[16]:
```
'Persa'
```

[17]:
```
sin_pelo = pelo_largo or pelo_corto
sin_pelo
```

[17]:
```
[True, False, True, True]
```

[18]:
```
peso_minimo + peso_maximo
```

[18]:
```
[2.3, 3.5, 5.4, 2.5, 6.8, 7.0, 9.1, 4.5]
```

[19]:
```
origen * 2
```

[19]:
```
['Irán',
 'Toronto',
 'California',
 'Tailandia',
 'Irán',
 'Toronto',
 'California',
 'Tailandia']
```

[20]:
```
'Siames' in gatos
```

[20]:
```
False
```

Métodos de las listas (comportamiento)

[21]:
```
gatos.append('Siberiano')
```

[22]:
```
gatos
```

[22]:
```
['Persa', 'Sphynx', 'Ragdoll', 'Siamés', 'Siberiano']
```

[23]:
```
gatos.append('Persa')
```

[24]:
```
gatos
```

[24]:
```
['Persa', 'Sphynx', 'Ragdoll', 'Siamés', 'Siberiano', 'Persa']
```

[25]:
```
gatos.remove('Persa')
```

[26]:
```
gatos
```

[26]:
```
['Sphynx', 'Ragdoll', 'Siamés', 'Siberiano', 'Persa']
```

[27]:
```
gatos.insert(0,'Persa')
```

[28]:
```
gatos
```

[28]:
```
['Persa', 'Sphynx', 'Ragdoll', 'Siamés', 'Siberiano', 'Persa']
```

[29]:
```
gatos.pop()
```

[29]:
```
'Persa'
```

[30]:
```
gatos
```

[30]:
```
['Persa', 'Sphynx', 'Ragdoll', 'Siamés', 'Siberiano']
```

[31]:
```
gatos.sort()
```

[32]:
```
gatos
```

[32]:
```
['Persa', 'Ragdoll', 'Siamés', 'Siberiano', 'Sphynx']
```

[33]:
```
gatos.reverse()
```

[34]:
```
gatos
```

[34]:
```
['Sphynx', 'Siberiano', 'Siamés', 'Ragdoll', 'Persa']
```

Copiando listas

[35]:
```
gatitos = gatos
```

[36]:
```
gatitos
```

[36]:
```
['Sphynx', 'Siberiano', 'Siamés', 'Ragdoll', 'Persa']
```

[37]:
```
gatos
```

[37]:
```
['Sphynx', 'Siberiano', 'Siamés', 'Ragdoll', 'Persa']
```

[38]:
```
gatitos[0] = 'Singapur'
```

[39]:
```
gatitos
```

[39]:
```
['Singapur', 'Siberiano', 'Siamés', 'Ragdoll', 'Persa']
```

[40]:
```
gatos
```

[40]:
```
['Singapur', 'Siberiano', 'Siamés', 'Ragdoll', 'Persa']
```

[41]:
```
print(id(gatitos), id(gatos))
```

```
139970037711936 139970037711936
```

[42]:
```
# Forma 1 para copiar listas
gatitos = []   # lista vacía
gatitos[:] = gatos[:]   # copia
```

[43]:
```
print(id(gatitos), id(gatos))
```

```
139970037055680 139970037711936
```

[44]:
```
# Forma 2 para copiar listas
gatitos = gatos.copy()
```

[45]:
```
print(id(gatitos), id(gatos))
```

```
139970088580544 139970037711936
```

[46]:
```
# Forma 3 para copiar listas
gatitos = list(gatos)
```

[47]:
```
print(id(gatitos), id(gatos))
```

```
139970036989504 139970037711936
```

Listas con distintos tipos de elementos

[48]:
```
superlista = ['México', 3.141592, 20, 1j, [1,2,3,'lista']]
```

[49]:
```
superlista
```

[49]:
```
['México', 3.141592, 20, 1j, [1, 2, 3, 'lista']]
```

[50]:
```
superlista[4][2]
```

[50]:
```
3
```

1.1.2 Tuplas

- Tuplas: es una colección que es ordenada, **NO** modificable (inmutable), indexable y permite miembros duplicados.

[51]:
```
datos1 = () # tupla vacía
print(type(datos1), id(datos1))
```

```
<class 'tuple'> 139970124693568
```

[52]:
```
datos2 = (1)
```

[53]:
```
type(datos2)
```

[53]:
```
int
```

[54]:
```
datos3 = (1,)
```

[55]:
```
type(datos3)
```

[55]:
```
tuple
```

[56]:
```
datos4 = (1,2,3)
```

[57]:
```
print(datos4, type(datos4))
```

```
(1, 2, 3) <class 'tuple'>
```

Todas las acciones que se pueden realizar con las listas aplican similarmente para las tuplas, excepto que las tuplas NO son mutables:

[58]:
```
datos4[1]
```

[58]:
```
2
```

[59]:
```
datos4[1] =  20 # Inválido debido a la inmutablidad
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-59-0ed7bb860755> in <module>
----> 1 datos4[1] =  20 # Inválido debido a la inmutablidad

TypeError: 'tuple' object does not support item assignment
```

Transformación entre tuplas y listas

[60]:
```
type(datos4)
```

[60]:
```
tuple
```

[61]:
```
lista_de_datos = list(datos4)
```

[62]:
```
print(lista_de_datos, type(lista_de_datos))
```

```
[1, 2, 3] <class 'list'>
```

[63]:
```
lista_de_datos[1] = 20
```

[64]:
```
nuevos_datos = tuple(lista_de_datos)
```

[65]:
```
print(nuevos_datos, type(nuevos_datos))
```

```
(1, 20, 3) <class 'tuple'>
```

1.1.3 Conjuntos

- Conjuntos: es una colección que **NO** es ordenada, modificable, **NO** indexable y **NO** permite miembros duplicados.

[66]:
```
conjunto = {4,1,8,0,4,20}
```

[67]:
```
print(conjunto, type(conjunto))
```

```
{0, 1, 4, 8, 20} <class 'set'>
```

[68]:
```
conjunto.add(-8)
```

[69]:
```
print(conjunto)
```

```
{0, 1, 4, 8, 20, -8}
```

[70]:
```
gatos_conjunto = set(gatos)
```

[71]:
```
print(gatos_conjunto)
```

```
{'Ragdoll', 'Singapur', 'Siberiano', 'Persa', 'Siamés'}
```

[72]:
```
gatos_conjunto.add('Siberiano')
```

[73]:
```
print(gatos_conjunto)
```

```
{'Ragdoll', 'Singapur', 'Siberiano', 'Persa', 'Siamés'}
```

[74]:
```
print('singapur' in gatos_conjunto)
```

```
False
```

[75]:
```
gatos_conjunto.update(nuevos_datos)
```

[76]:
```
gatos_conjunto
```

[76]:
```
{1, 20, 3, 'Persa', 'Ragdoll', 'Siamés', 'Siberiano', 'Singapur'}
```

[77]:
```
len(gatos_conjunto)
```

[77]:
```
8
```

Operaciones sobre conjuntos

[78]:
```
A = set('Taza')
B = set('Casa')
```

[79]:
```
print(A)
print(B)
```

```
{'a', 'z', 'T'}
{'a', 's', 'C'}
```

[80]:
```
A - B # elementos en A, pero no en B
```

[80]: {'T', 'z'}

[81]:
```
A | B # elementos en A o en B o en ambos
```

[81]: {'C', 'T', 'a', 's', 'z'}

[82]:
```
A & B # elementos en ambos conjuntos
```

[82]: {'a'}

[83]:
```
A ^ B # elementos en A o en B, pero no en ambos
```

[83]: {'C', 'T', 's', 'z'}

1.1.4 Diccionarios

- Diccionarios: es una colección que **NO** es ordenada, modificable, indexable y **NO** permite miembros duplicados.

[84]:
```
dicc = {'Luis': 20, 'Miguel': 25}
```

[85]:
```
print(dicc, type(dicc))
```

```
{'Luis': 20, 'Miguel': 25} <class 'dict'>
```

[86]:
```
dicc['Luis'] # acceder a un elemento del diccionario
```

[86]: 20

[87]:
```
dicc['Juan'] = 30 # agregar un nuevo elemento al diccionario
```

[88]:
```
print(dicc)
```

```
{'Luis': 20, 'Miguel': 25, 'Juan': 30}
```

[89]:
```
'Luis' in dicc
```

[89]: True

[90]:
```
otro =  dict(nuevo='estrellas', viejo='cosmos', edad=15000000)
```

[91]:
```
otro
```

[91]:
```
{'nuevo': 'estrellas', 'viejo': 'cosmos', 'edad': 15000000}
```

[92]:
```
dict(zip('abcd',[1,2,3,4,5]))
```

[92]:
```
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

Transformación de diccionarios a otras colecciones

[93]:
```
list(dicc)
```

[93]:
```
['Luis', 'Miguel', 'Juan']
```

[94]:
```
list(dicc.values())
```

[94]:
```
[20, 25, 30]
```

[95]:
```
set(dicc)
```

[95]:
```
{'Juan', 'Luis', 'Miguel'}
```

[96]:
```
tuple(dicc)
```

[96]:
```
('Luis', 'Miguel', 'Juan')
```

Operaciones sobre diccionarios

[97]:
```
dicc
```

[97]:
```
{'Luis': 20, 'Miguel': 25, 'Juan': 30}
```

[98]:
```
len(dicc)
```

[98]:
```
3
```

[99]:
```
dicc.keys()
```

[99]:
```
dict_keys(['Luis', 'Miguel', 'Juan'])
```

[100]:
```
dicc.values()
```

[100]:
```
dict_values([20, 25, 30])
```

[101]:
```
dicc['fulano'] = 100
```

[102]:
```
dicc
```

[102]:
```
{'Luis': 20, 'Miguel': 25, 'Juan': 30, 'fulano': 100}
```

[103]:
```
del dicc['Miguel']
```

[104]:
```
dicc
```

[104]:
```
{'Luis': 20, 'Juan': 30, 'fulano': 100}
```

[105]:
```
otro
```

[105]:
```
{'nuevo': 'estrellas', 'viejo': 'cosmos', 'edad': 15000000}
```

[106]:
```
dicc.update(otro)
```

[107]:
```
dicc
```

[107]:
```
{'Luis': 20,
 'Juan': 30,
 'fulano': 100,
 'nuevo': 'estrellas',
 'viejo': 'cosmos',
 'edad': 15000000}
```

[108]:
```
nuevo = {'Luis':512, 'viejo':2.1}
```

[109]:
```
dicc.update(nuevo)
```

[110]:
```
dicc
```

[110]:
```
{'Luis': 512,
 'Juan': 30,
 'fulano': 100,
 'nuevo': 'estrellas',
 'viejo': 2.1,
 'edad': 15000000}
```

[ ]:
```

```