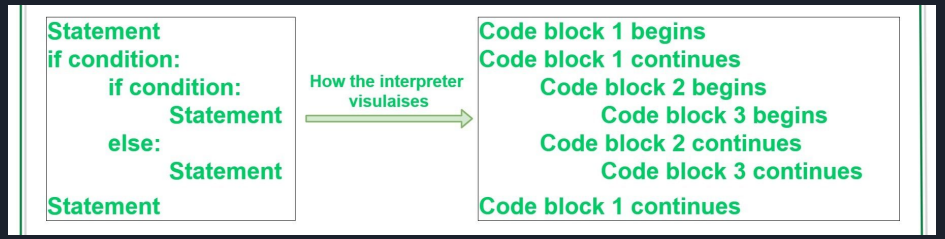
Introducción a Python

Listas, Ciclos, Tuplas, Conjuntos y Diccionarios

Prof. Jhoan Steven Delgado V.

Identación en Python



- La indentación es el espacio en blanco inicial (espacios y tabulaciones) antes de cualquier declaración en Python.
- Python trata las sentencias con el mismo nivel de identación (sentencias con el mismo número de espacios en blanco antes de ellas) como un único bloque de código.
- La indentación en Python es una forma de indicar al intérprete de Python que el grupo de sentencias pertenece a un bloque de código concreto.

```
number = 50

if(number != 0):
 if(number % 2 == 0):
   print("Given number is Even")
 else:
   print("Given number is Odd")
else:
 print("Given number is neither even nor odd")
```

Errores comunes de identación

Incorrecto

```
if( 1 == 1):
print("This is test code")
print("This is test code1")
```

Correcto

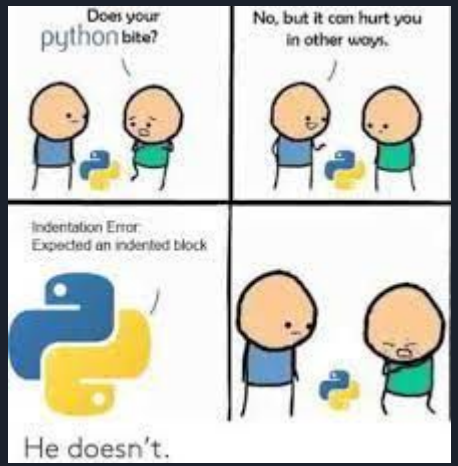
```
if( 1 == 1):
 print("This is test code")
 print("This is test code1")
```

```
if( 1 == 1):
   print("This is test code")

 print("This is test code1")
```

```
if( 1 == 1):
   print("This is test code")

   print("This is test code1")
```

```
if( 1 == 1):
 print("This is test code")
     print("This is test code1")
```

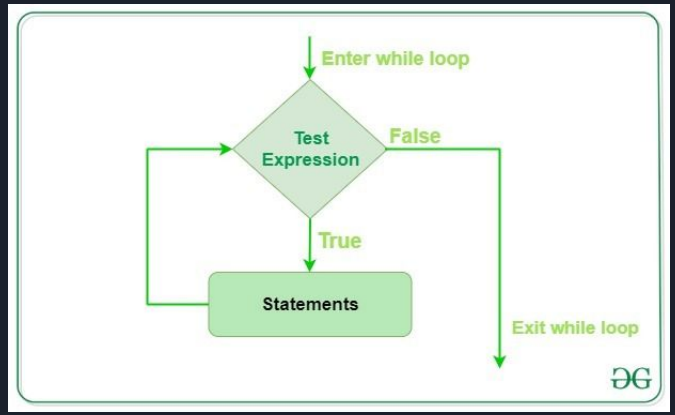
```
if( 1 == 1):
 print("This is test code")
 print("This is test code1")
```



Ciclos

Estructuras de control repetitivas (Ciclos)

- Los ciclos permiten repetir un conjunto de instrucciones mientras una condición (expresión booleana) sea verdadera
- Son similares a los condicionales (if) en tanto que el bloque de código dentro de ambos (if y ciclos) se ejecuta si la condición es verdadera
- La gran diferencia es que los condicionales evalúan la condición solamente una (1) vez, mientras que los ciclos la evaluarán hasta que sea falsa ejecutando las instrucciones en su interior mientras sea verdadera




```
i = 1
while i<10:
    print (“Python es lo mejor”)
    i = i + 1
```

Estructuras de control repetitivas (Ciclos) claúsula break statement

- Con la sentencia break podemos detener el bucle incluso si la condición while es verdadera

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

DEMO

Listas y Ciclo For

Listas

- Existen varios tipos de datos compuestos, los cuales son utilizados para agrupar varios valores en un solo lugar.
- En Python, el más versátil es la Lista, que puede escribirse como una secuencia de valores separados por coma encerrados por corchetes.
- Las posiciones del índice comienzan en 0, no en 1

```
cuadrados = [1, 4, 9, 16, 25]
print (cuadrados)
```

- Las listas están indexadas, es decir, se puede acceder a cada posición utilizando un índice.

```
print (cuadrados[3])
print (cuadrados[0])
```

- Las listas pueden ser rebanadas (sliced), se puede obtener una porción de ellas con el operador **:**

cuadrados[inicio:final-1]

```
print (cuadrados[1:3])
```

Operaciones en Listas

```
cuadrados = [1, 4, 9, 16, 25]
print (cuadrados)
```

- Cuando se rebana una lista, no son obligatorios los índice inicial ni final

```
print (cuadrados[2:])
print (cuadrados[:3])
print (cuadrados[:])
```

- Concatenación

```
cuadrados = cuadrados + [36, 49, 63]
print (cuadrados)
```

- Modificación de valores en una posición.

```
cuadrados [7] = 64
```

- Agregar nuevos valores a la lista con la operación append

```
cuadrados.append(81)
cuadrados.append(100)
print (cuadrados)
```

Operaciones en Listas

- De manera similar a las cadenas de texto (strings), podemos conocer la cantidad de elementos con la función len().

```
cuadrados = [1, 4, 9, 16, 25]
print (cuadrados)
```

```
print (len(cuadrados))
```

- El operador “in” de python para evaluar la membresía de un elemento de una lista.

```
1 in [1, 2, 3]    # True
0 in [1, 2, 3]    # False
```

- También es posible anidar listas.

```
a = [0,1,2,3,4]
b = [5,6,7,8,9]
numbers = [a,b]
print (numbers)
```

Cadenas de texto (Strings) como Listas

- Las cadenas de texto pueden ser tratadas como listas.

```
saludo = “Hola a todos”
print (saludo[0])
print (saludo[3])
print (saludo[len(saludo)-1])
print (saludo[1:5])
print (saludo[-3])
print (saludo[:-1])
```

- Sin embargo, a diferencia de las listas, las posiciones de una cadena son inmutables

```
frase = “Python nació en 1991”
frase[0] = “P”
frase[-1] = 2
frase[:6] = “C++”
```

El Ciclo For

- La cláusula for es utilizada en Python para iterar sobre cualquier secuencia (lista o cadena de texto, por ejemplo)

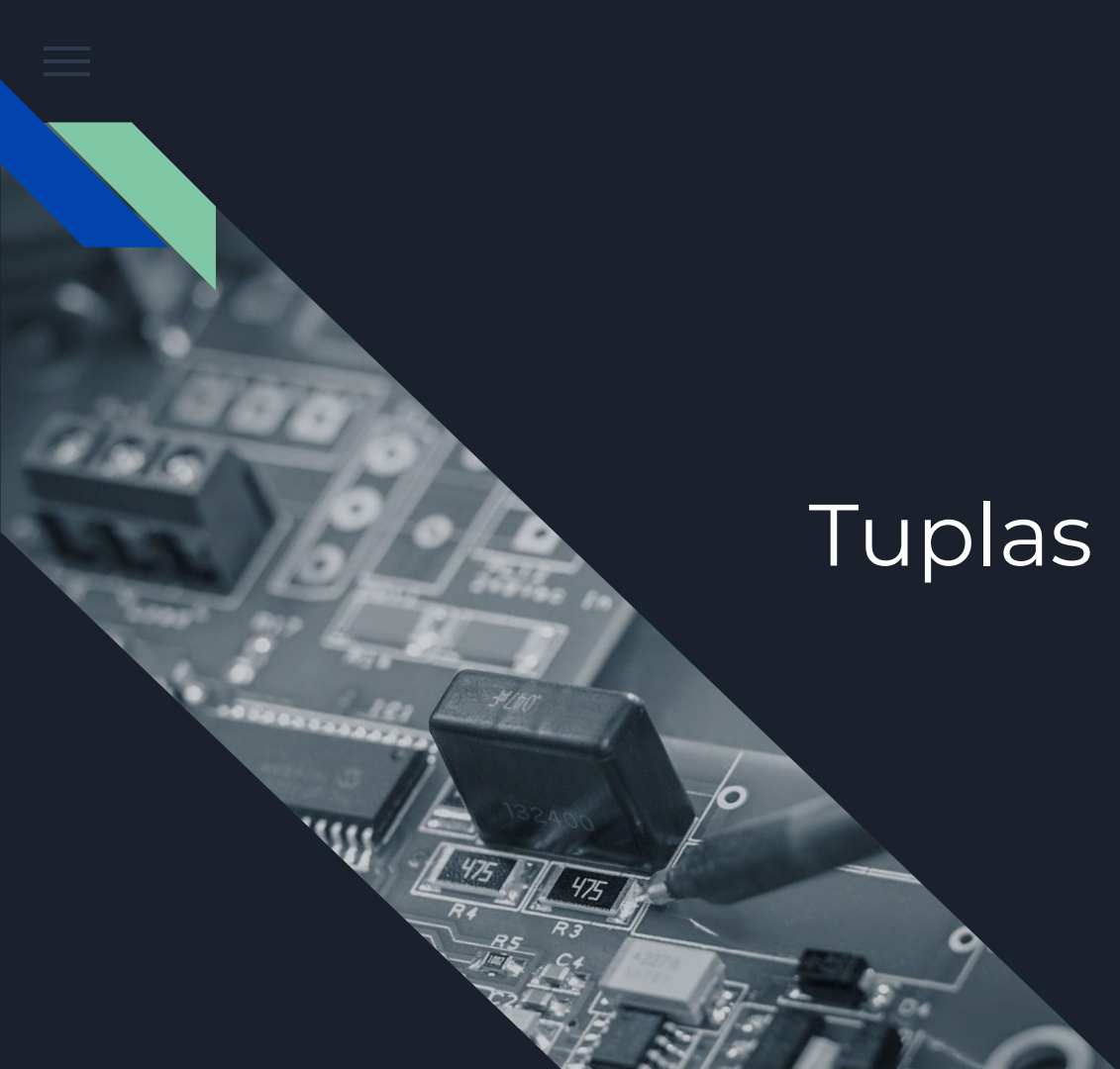
```
animales = [“gato”,”perro”,”loro”]
for animal in animales:
    print (animal, len(animal))
```

- Se puede combinar su uso con la función range que genera progresiones aritméticas

```
for i in range(6):
  print (i)

for i in range (5,10):
  print (i)

print (list(range(3,2,12)))
```

DEMO

Tuplas



Tuplas

- Son tipos de datos de la categoría secuencias en Python, similares a las listas. Con la diferencia de que estas son Inmutables.
- Son representadas rodeadas por paréntesis (o sin nada), en lugar de corchetes como las listas.

```
persona = (“Ana”,”Paz”,35)
print (persona)
```

```
tupla = 34,58,63
```

- Soportan la concatenación

```
tupla = tupla + (87,45)
```

- Los elementos de una tupla pueden ser otras tuplas o valores de cualquier otro tipo de dato.

```
familia = (
    (“Luis”,”padre”,”amarillo”),
    ("Felipe","hijo",True),
    ("Doris","prima",[5,6,7,8])
)
```

- Puedes acceder a las posiciones de una tupla con corchetes igual que las listas y cadenas de texto

```
print(persona[1]); print(familia[2][2])
```

Tuplas

- Las posiciones en una tupla no se pueden modificar, pero si las posiciones de un elemento mutable (como las listas) al interior de una tupla.

```
gastos = ([150,320,474],[42,86])
gastos[1][0] = 56
```

- Packing y Unpacking

```
x = 1,2,3,4 #packing
```

```
a,b,c,d = x
```

DEMO

Conjuntos

Conjuntos

- Es una colección de elementos **no repetidos** donde el orden no es importante.

```
cesta = {‘Limón’,’Naranja’,’Limón’}
```

- Podemos utilizar las operaciones clásicas de conjuntos.

```
a = set(“abcdefghi”)
b = set(“acegijklm”)
print (a|b); print (a&b)
print (a^b); print (a-b)
```

- Iterar sobre los elementos de un conjunto.

```
for elem in a:
  print(elem)
```

- Verificar si un elemento está en un conjunto.

```
‘Naranja’ in cesta
```

DEMO

Diccionarios

Diccionarios

- Son conocidos en otros lenguajes también como arreglos asociativos.
- Asocia los valores con las claves y permite recuperar rápidamente el valor correspondiente a una clave determinada
- A diferencia de las secuencias, que son arreglos indexados por números, los diccionarios están indexados por una clave que puede ser de cualquier tipo inmutable, por ejemplo cadenas de texto

```
casa = {‘color’:’rojo’,’area’:500}
print (casa)
print (casa[‘color’])
```

```
tweet = {
    "user" : "joelgrus",
    "text" : "Data Science is Awesome",
    "retweet_count" : 100,
    "hashtags" : ["#data", "#science", "#datascience", "#awesome", "#yolo"]
}
```

```
vuelo = dict(origen=‘Cali’,
destino=‘Bogotá’, distancia=350)
```

Recorrer Diccionarios

```
salud = {‘peso’:62,
’altura’:1.72,’edad’:28}


for clave in list(salud):
  print(clave, salud[clave])
```

```
for clave, valor in salud.items():
  print(clave, valor)


for clave in salud:
  print(clave, salud[clave])
```

DEMO

Referencias

1. https://www.scaler.com/topics/python/indentation-in-python/
2. https://www.geeksforgeeks.org/indentation-in-python/
3. Introducción a la Programación con Python. Profesor Juan Manuel Reyes, nivelatorio en Python MCD, Universidad Icesi.
4. Data Science from Scratch: First Principles with Python. Joel Grus
5. Python Crash Course: A Hands-On, Project-Based Introduction to Programming. Eric Matthes