Curso profesional de Python

Marcos Antonio Serrano M.

Módulo I - Estructura y elementos del Lenguaje

Marcos Antonio Serrano M.

Introducción a Python

- Apareció en 1991
- Python es un lenguaje de programación interpretado
- Su filosofía es hacer hincapié en una sintaxis que favorezca un código legible
- Es un lenguaje de programación multiparadigma (Orientado a objetos, imperativa y funcional)
- Es multiplataforma
- Es de código abierto
- Tiene una comunidad muy grande

Comentarios en Python

```
“””
        Hola mundo!
        Este comentario es multilineas
 “””
# Hola mundo
```

Variables en Python

Nombre_de_variable = valor o operación

A = 2

B = “Hola”

Tipos de datos

Numeros:

Tipo	Ejemplo
Int	23
Long	23L
Octal	027
hexadecimal	0x17

Tipos de datos

Cadenas:

- Comillas simples

  ‘Texto entre comillas simples’

- Comillas dobles

  “Texto con comillas dobles”

- Cadena con codigo escapes

  ‘Texto entre \n\tcomillas simples’

- Cadena multilinea

  “”” Testo linea1

  Linea 2

  .

  .

  “””

Tipos de datos

Boleanos:

- True
- False

Listas:

- A = [“coche”, “moto”, 21, 300]

Tuplas:

- (1, 2, 3, 4)

Diccionario:

- {“nombre”: “Pedro”,
  “Apellidos”: “Galindo”,
  “Edad”: 24}

Tipos de datos

Operador	Descripción	Ejemplo
+	Suma	a = 3 + 3 # resultado 6
-	Resta	a = 3 - 2 # resultado 1
	Negación	a = -3 # resultado -3
*	Multiplicación	a = 2 * 2 # resultado 5
**	Exponente	a = 2 ** 6 # resultado 12
/	División	a = 3.5 / 2 # resultado 1.75
//	Divisíon entera	a = 3.5 / 2 # resultado 1.0
%	Módulo	a = 7 % 2 # resultado 1

Operadores Aritméticos

Operador	Descripción	Ejemplo
==	¿Es igual a y b?	10 == 15 # False
!=	¿Es distinto a y b?	3 != 19 # True
<	¿Es a menor que b?	8 < 9 # True
>	¿Es a mayor que b?	10 > 40 # False
<=	¿Es a menor o igual que b?	10 <= 10 # True
>=	¿Es a mayor o igual que b?	15 >= 10 # True

Estructura de control de flujo e identación

- Identación
- PEP 8
- Encoding

```
# -*- coding: utf-8 -*-
```

- Asignación múltiple

```
a, b, c = 19, 'hola', False
```

Estructuras de control de flujo condicionales

Los condicionales nos permiten comprobar condiciones y hacer que nuestro programa se comporte de una forma u otra, que ejecute un fragmento de código u otro, dependiendo de esta condición

- If
- Else
- Elif

Estructuras de control iterativas

Nos permiten ejecutar un mismo código, de manera repetida, mientras se cumpla una condición.

En Python se dispone de dos estructuras cíclicas:

- El bucle while
- El bucle for

Módulo II - Métodos principales del objeto string

Marcos Antonio Serrano M.

Metodos de formato

Metodo	Retorna
capitalize()	Copia de la cadena con la primera letra en mayúsculas.
lower()	Copia de la cadena en minúsculas
upper()	Copia de la cadena en mayúsculas.
swapcase()	Copia de la cadena convertidas las mayúsculas en minúsculas y viceversa.
title()	Copia de la cadena con la primera de cada palabra en mayúsculas.
center(longitud, "caracter de relleno")	Copia de la cadena centrada.

Metodos de formato

Metodo	Retorna
ljust(longitud, "caracter de relleno")	Copia de la cadena alineada a la izquierda.
rjust(longitud, "caracter de relleno")	Copia de la cadena alineada a la derecha.
zfill(longitud)	Copia de la cadena rellena con ceros a la izquierda hasta alcanzar la longitud final indicada.

Métodos de conversión

Los métodos de conversión nos permiten transformar un tipo de valor a otro tipo:

- int()
- str()
- float()
- tuple()
- list()

Métodos de Búsqueda

Metodo	Retorna
count(elemento)	Cantidad de apariciones de un elemento.
index(elemento, indice_inicio, indice_fin)	Numero de indice en el cual se encuentra la busqueda.

Métodos de Validación

Metodo	Retorna
startswith("subcadena" , posicion_inicio, posicion_fin)	Válida si una cadena comienza por una subcadena específica.
endswith("subcadena" , posicion_inicio, posicion_fin)	Válida si una cadena termina por una subcadena específica.
isalnum()	Válida si una cadena es alfanumérico.
isalpha()	Válida si una cadena tiene solo letras.
isdigit()	Válida si solo son números.
islower()	Válida si la cadena solo contiene minúsculas.

Métodos de Validación

Metodo	Retorna
isupper()	Válida si la cadena solo contiene mayúsculas.
isspace()	Válida si una cadena contiene solo espacios en blanco.
istitle()	Válida si una cadena tiene formato de título.

Métodos de Sustitución

Metodo	Retorna
format(*args, **kwargs)	Dar formato a una cadena, sustituyendo texto dinámicamente.
replace("subcadena a buscar", "subcadena por la cual reemplazar")	Busca una cadena específica y la reemplaza por otra.
strip("caracter")	Elimina caracteres a la derecha y a la izquierda de una cadena.
lstrip("caracter")	Elimina caracteres a la izquierda de una cadena.
rstrip("caracter")	Elimina caracteres a la derecha de una cadena.

Métodos de unión y división

Metodo	Retorna
join(iterable)	Unir una cadena de forma iterativa.
partition("separador")	Parte una cadena en tres, utilizando un separador.
split("separador")	Parte una cadena en varias partes, utilizando un separador.
splitlines()	Parte una cadena que contenga líneas.

Encoding

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

u = 'idzie wąż wąską dróżką'
uu = u.decode('utf8')
s = uu.encode('cp1250')
print(s)
```

Módulo III - Tipos de datos complejos

Marcos Antonio Serrano M.

Tuplas

- Las tuplas son secuencias o listas de elementos.
- No son mutables, es decir que no se pueden modificar.
- Su acceso y eso consume menos recurso por ende son mucho más rápido que las listas o diccionarios.

A = (‘elemento1’, 2, 3,)

Listas

- Las listas son compuestas por datos cuya cantidad o valor varían.
- Son mutables, es decir que se pueden modificar en cualquier momento.
- Están dotadas de una variedad de operaciones muy útiles.

A = [‘elemento1’, 2, 3,]

Listas

Metodo	Retorna
append("nuevo elemento")	Permite agregar un nuevo elemento.
extend(otra_lista)	Agregar varios elemento al final de la lista.
insert(posición, "nuevo elemento")	Agregar un elemento en la posición deseada.
pop()	Eliminar el último elemento de la lista.
pop(índice)	Eliminar el elemento por su índice.

Listas

Metodo	Retorna
remove("valor")	Elimina un elemento por su valor.
reverse()	Ordena una lista en reverso.
sort()	Ordena una lista en ascendente.
count(elemento)	Cuenta la cantidad de apariciones de un elemento.
index(elemento[, indice_inicio, indice_fin])	Nos da el índice donde se encuentra un elemento.

Listas

Metodo	Retorna
len(element)	Cuenta la cantidad de elementos de una lista.
max(lista)	El valor maximo.
min(lista)	El valor minimo.

Diccionarios

- Los diccionarios son mutables, es decir que se pueden modificar.
- Se construye por medio de {}
- Tienen una llave y un valor

Coche = {'puertas': 2, 'tipo': 2, 'matricula': ‘554CD’}

Diccionarios

Metodo	Retorna
diccionario.clear()	Vacía el diccionario.
dict.copy()	Copia un diccionario.
update(diccionario)	Concatenar diccionarios.
get(clave, "valor x defecto si la clave no existe")	Retorna el valor de un elemento buscado por su clave.

Diccionario

Metodo	Retorna
has_key(clave)	Permite saber si una clave existe.
iteritems() - items()	Obtener claves y valores de un diccionario.
keys()	Claves de un diccionario.
values()	Valores de un diccionario.
len(diccionario)	Cantidad de elementos de un diccionario.

Módulo IV - Funciones definidas

Marcos Antonio Serrano M.

Definiendo funciones

- Una función es una forma de agrupar expresiones o sentencias que realizan determinadas acciones.
- Las funciones se ejecutan solo cuando son llamadas.
- Las funciones pueden recibir y retornar valores.

Definiendo funciones

```
def mi_funcion():
    # aquí el algoritmo
```

Para ejecutar la función solo tenemos que llamarla por su nombre:

```
mi_funcion()
```

Definiendo funciones

Las funciones pueden recibir parámetros y estos pueden tener un valor por defecto.

```
def mi_funcion(a, b):
        # aquí el algoritmo

def funcion_a(a=1, b=2):
        # mi codigo
```