Unidad IV

Introducción a Python - Parte 1

27 de abril del 2020

Agenda

Clases y objetos
- Herencias

Exceptions
- Capturando excepciones
- Lanzando excepciones

Agenda

Clases y objetos

Herencias

Exceptions

Capturando excepciones

Lanzando excepciones

Clases y objetos

- Clase es como una plantilla para poder crear objetos.
- En esta plantilla se tiene la información necesaria, de las información que va a tener y poder guardar los objetos, y las acciones.
- Un objeto es una instancia de la clase. Es la “materialización” de la clase y quien realiza las “acciones”.

Clases y objetos

- Clases (y objetos) están compuestas por: atributos y métodos.
- atributos == variables
- métodos == funciones

Clases y objetos

```
class Persona:
    def __init__(self, nombre, apellido):
        self.nombre = nombre
        self.apellido = apellido
        self.edad = 0

    def set_edad(self, edad):
        self.edad = edad

    def print_edad(self):
        print("%s %s tiene %d" % (self.nombre,
                                  self.apellido,
                                  self.edad))
```

Clases y objetos

```
persona1 = Persona("Emmanuel", "Arias")
persona1.print_edad()
persona1.set_edad(28)
persona1.print_edad()
persona2 = Persona("Nombre", "Apellido")
persona2.print_edad()
persona2.set_edad(21)
persona2.print_edad()
```

Agenda

Clases y objetos

Herencias

Exceptions

- Capturando excepciones
- Lanzando excepciones

Herencias

- Herencia es uno de los pilares de la POO, que nos permite crear una clase general y luego crear clases más especializada.
- Se utiliza para reutilizar código.

Herencias

```
class UsuarioComun(Persona):
    def __init__(self, nombre, apellido):
        Persona.__init__(self, nombre, apellido)
        self.fecha_inicio = "2020-01-01"

    def set_edad(self, edad):
        if edad < 18:
            print("Error: Tiene que"
                  "ser mayor de edad")
        else:
            self.edad = edad

    def get_data_user(self):
        print(self.nombre, self.apellido)
```

Agenda

Clases y objetos

Herencias

Exceptions

- Capturando excepciones
- Lanzando excepciones

Exceptions

- Los errores que se producen en “tiempo de ejecución” se denominan *Excepciones*.
- Las excepciones cuando se van a propagar, hasta la función que la llamo, y si nadie la trata, se va a propagar hasta la función que llamó a esa función, y así hasta interrumpir el programa.
- Para evitar que la excepción se propague, debemos “capturarla”.
- Esto se hace usando *try ... except*

Agenda

Clases y objetos

Herencias

Exceptions

Capturando excepciones

Lanzando excepciones

Capturando excepciones

```
try:
    x = 3 / 0
except ZeroDivisionError as err:
    print(err)
```

Capturando excepciones

```
denominador = input("ingrese el denominador")
try:
    x = 3 / denominador
    with open("test.txt", "w") as f:
        f.write(x)
except TypeError as err:
    print("No se puede escribir entros con write(
except ZeroDivisionError as err:
        print("No puede usar cero como denominado
except Exception as err:
    print("Exception no controlada)
```

Agenda

Lanzando excepciones

- También podemos nosotros lanzar una excepción y crear nuestra propia excepción.
- Esto se hace cuando queres evitar alguna condición y obviamente surja la necesidad.
- La idea no es crear excepciones cada vez que no se cumpla una condición.
- Para lanzar una excepción usamos *raise*
- Para crear una excepción definimos una nueva clase y heredamos de alguna clase *Error* cuya clase base es *Exception*
- `https://docs.python.org/3/library/exceptions.html`

Lanzando excepciones

```
def set_edad(self, edad):
    if edad < 18:
        raise ValueError("No puede ser"
                         "menor de edad")
    self.edad = edad

>>> ValueError: no puede ser menor de edad
```