Anexo 4. Unidad 4. Validación de datos

Se van a ver distintas formas de validación de datos con distintos módulos y paquetes.

DATETIME

#Funcion para validar la fecha introducida from datetime import datetime

```
def validoFecha(date):
  buena=False
  try:
    datetime.strptime(date,'%d/%m/%Y')
    buena=True
  except ValueError:
    buena=False
  return buena
```

o también:

from datetime import datetime while True:

```
  fecha_str = input("Introduzca la fecha de operación
  'dd/mm/aaaa'")
  try:
    fecha = datetime.strptime(fecha_str, '%d/%m/%Y')
  except ValueError:
    print("No ha ingresado una fecha válida")
  else:
    break
```

EXPRESIONES REGULARES

Expresión regular=> expresión que identifica un número de cadenas que deben coincidir con ella.

(\d{d})=> \d=la cadena coincide con cualquier dígito decimal Unicode [0-9]; {d}=especifica que exactamente m copias, de la expresión regular anterior, deben coincidir.

Ejemplo: a{6}=> la cadena debe coincidir con seis caracteres a.

re. **compile**(*pattern*, *flags=0*)=>compila un patrón de expresión regular en un objeto de expresión regular, que puede ser usado para las coincidencias usando match(), search() y otros métodos.

re. **fullmatch**(*pattern*, *string*, *flags=0*)=>si toda la *string* («cadena») coincide con el *pattern*

(«patrón») de la expresión regular, retorna un correspondiente objeto match. Retorna None si la cadena no coincide con el patrón; notar que esto es diferente de una coincidencia de longitud cero.

```
import re
fecha = re.compile(r'(\d{2})/(\d{2})/(\d{4})')
tel = re.compile(r'(\d{9})')
nif = re.compile(r'(\d{8})([A-Z])')


comprobar(nif,input('NIF:  '))
comprobar(tel,input('Teléfono: '))
comprobar(fecha,input('Fecha:   '))

def comprobar(formato,dato):
  while formato.fullmatch(dato) == None :
    print("El dato introducido no tiene el formato adecuado")
    formato = input('Por favor, vuelva a introducir el dato: ')
  return formato


# Funcion que comprubea el DNI


def validoDNI(dni):
  tabla = "TRWAGMYFPDXBNJZSQVHLCKE"
  numeros = "1234567890"
  dni = dni.upper()
  if len(dni) == 9:
    dig_control = dni[8]
    dni = dni[:8]
    return len(dni) == len([n for n in dni if n in numeros]) \
             and   tabla[int(dni)%23]   ==   dig_control
    print("El DNIintroducido no es correcto !!!!!!")
    return False
```

Esta función se encarga de comprar si el DNI que recibe es un DNI correcto, en ese caso devuelve True en caso contrario devuelve False.

```
#Funcion para validar el EMAL.


def validoEMail(correo):
  validado=False
  if re.match('^[(a-z0-9\_\-\.)]+@[(a-z0-9\_\-\.)]+\.[(a-
  z)]{2,15}$', correo.lower()):
    validado=True
  else:
    print("El Email introducido no es correcto !!!!!!")
    validado=False
    return validado
```

Esta función se encarga de comparar si el Email que recibe es un Email correcto, en ese caso devuelve True en caso contrario devuelve False.

*re. **match**(pattern, string, flags=0)=>si cero o más caracteres al principio de la string («cadena») coinciden con el pattern(«patrón») de la expresión regular, retorna un objeto match correspondiente. Retorna None si la cadena no coincide con el patrón; notar que esto es diferente de una coincidencia de longitud cero.*

Notar que incluso en el modo MULTILINE, re.match() sólo coincidirá al principio de la cadena y no al principio de cada línea.

Si se quiere localizar una coincidencia en cualquier lugar de la string («cadena»), se utilizara search() en su lugar (ver también search() vs. match()).

Directiva	Significado	Ejemplo	Notas
%a	Día de la semana como nombre abreviado según la configuración regional.	*Sun, Mon, …, Sat (en_US);* *So, Mo, …, Sa (de_DE)*	(1)
%A	Día de la semana como nombre completo de la localidad.	*Sunday, Monday, …, Saturday (en_US);* *Sonntag, Montag, …, Samstag (de_DE)*	(1)
%w	Día de la semana como un número decimal, donde 0 es domingo y 6 es sábado.	0, 1, …, 6	
%d	Día del mes como un número decimal rellenado con ceros.	01, 02, …, 31	(9)

%b	Mes como nombre abreviado según la configuración regional.	*Jan, Feb, …, Dec (en_US);* *Jan, Feb, …, Dez (de_DE)*	(1)
%B	Mes como nombre completo según la configuración regional.	*January, February, …, December (en_US);* *Januar, Februar, …, Dezember (de_DE)*	(1)
%m	Mes como un número decimal rellenado con ceros.	01, 02, …, 12	(9)
%y	Año sin siglo como un número decimal rellenado con ceros.	00, 01, …, 99	(9)
%Y	Año con siglo como número decimal.	0001, 0002, …, 2013, 2014, …, 9998, 9999	(2)
%H	Hora (reloj de 24 horas) como un número decimal rellenado con ceros.	00, 01, …, 23	(9)
%I	Hora (reloj de 12 horas) como un número decimal rellenado con ceros.	01, 02, …, 12	(9)
%p	El equivalente de la configuración regional de AM o PM.	AM, PM (en_US); am, pm (de_DE)	(1), (3)
%M	Minuto como un número decimal rellenado con ceros.	00, 01, …, 59	(9)
%S	Segundo como un número decimal rellenado con ceros.	00, 01, …, 59	(4), (9)
%f	Microsegundo como un número decimal, rellenado con ceros a la izquierda.	000000, 000001, …, 999999	(5)
%z	Desplazamiento (*offset*) UTC en la forma `±HHMM[SS[.ffffff]]` (cadena de caracteres vacía si el objeto es naíf (*naive*)).	(vacío), +0000, -0400, +1030, +063415, -030712.345216	(6)
%Z	Nombre de zona horaria (cadena de caracteres vacía si el objeto es naíf (*naive*)).	(vacío), UTC, GMT	(6)
%j	Día del año como un número decimal rellenado con ceros.	001, 002, …, 366	(9)
%U	Número de semana del año (domingo como primer día de la semana) como un número decimal rellenado con ceros. Todos los días en un nuevo año anterior al primer domingo se consideran en la semana 0.	00, 01, …, 53	(7), (9)
%W	Número de semana del año (lunes como primer día de la semana) como número decimal. Todos los días en un nuevo año anterior al primer lunes se consideran en la semana 0.	00, 01, …, 53	(7), (9)
%c	Representación apropiada de fecha y hora de la configuración regional.	*Tue Aug 16 21:30:00 1988 (en_US);* *Di 16 Aug 21:30:00 1988*	(1)

MÉTODOS DE CADENA

VALIDAR NIF

- Debe contener 9 caracteres.
- Los 8 primeros deben ser números.
- El 9 caracter, debe ser una letra.
- Si se superan todas las validaciones, el dato es correcto: Return True.

```
def nif_correcto(nif):
  if len(nif) != 9:
    print("El nif debe contener 9 caracteres")
    return False
  # - Valida que los ocho primeros caracteres sean numeros.
  if not nif[0:8].isdigit():
    print("Los primeros ocho caracteres deben ser numeros")
    return False


# - Valida que el ultimo caracter sea una letra mayusculas A-Z.
  if not 'A' <=nif[8] <= 'Z':
    print("El último caracter debe de ser una letra mayúscula")
    return False return True
```

VALIDAR TELEFONO

- Debe contener 9 caracteres, todos digitos.
- Si se superan todas las validaciones, el dato es correcto: Return True.

```
def telefono_correcto(telefono):
      if len(telefono) != 9:
            print("El telefono debe contener 9 caracteres")
            return False else:
      if not telefono.isdigit():
            print("El telefono debe contener solo digitos")
            return False
      return True
```

VALIDAR CORREO.

- Debe contener el caracter arroba ('@').
- Si se superan todas las validaciones, el dato es correcto: Return True.

```
def correo_correcto(correo):
  if '@' not in correo:
    print("La dirección de correo no tiene arroba")
    return False return True
```

VALIDAR FECHA

- Debe de tener 10 caracteres.
- Los caracteres de la posición 2 y 5 deben ser barras '/'.
- Los otros caracteres deben ser numeros
- El dia no debe de superar el 31.
- El mes debe de ser entre 1 y 12
- El año debe ser igual a 2020. def fecha_alta_correcta(f_alta):

```
if len(f_alta)!= 10:
  print("La fecha debe tener 10 caracteres")
  return False
if  f_alta[2]  !=  '/'  or  f_alta[5]  !=  '/':
  print("Formato incorrecto: faltan las barras")
  return False
for x in range(0, 10):
  if x == 2 or x == 5:
  continuecontinue
  elif not f_alta[x].isdigit():
    print("La fecha debe contener solo numeros")
    return False
#- previamente validados que son numeros, se desglosa en dia,
# mes y año.

dia  = int (f_alta[0:2])
mes  = int (f_alta[3:5])
anyo = int (f_alta[6:10])
if dia > 31:
       print("Formato dia incorrecto (> 31)")
       return False
if mes < 1 or mes > 12:
  print("Formato mes incorrecto [1-12]")
  return False
if anyo != 2020:
  print("Año debe ser 2020")
  return False return True
```