_1type/_1strings.py

```
"""                                     Strings                                 """
# --- input("Siempre es String") ---
# --- Mapping ---
mapping = {'nombre': 'Juan', 'edad': 30}
mensaje = "Mi nombre es {nombre} y tengo {edad} años.".format_map(mapping)
# --- Slicing ---
language = 'Python'
first_three = language[:3]    # Pyt
last_three1 = language[-3:]   # hon
last_three2 = language[3:]    # hon
salto= language[::2]          # Pto
reverse= language [::-1]      # nohtyP
# --- Adjusting ---
company= "Coding for all"
print(company.casefold())                   #mejor que lower ß raras
print(company.lower())                      #coding for all
print(company.upper())                      #CODING FOR ALL
print(company.capitalize())                 #Coding for all
print(company.title())                      #Coding For All
print(company.swapcase())                   #cODING fOR aLL
print(company.expandtabs(4))                #Coding for    All
# --- Strip ---
desnuda = "###Hola, esto es una cadena de ejemplo###"
cadena_limpia = desnuda.strip("#") # Hola, esto es una cadena de ejemplo
cadenaca = "   Hola Mundo   "
cadena_strip = cadenaca.strip()    # "Hola Mundo"
cadena_lstrip = cadenaca.lstrip()  # "Hola Mundo   "
cadena_rstrip = cadenaca.rstrip()  # "   Hola Mundo"
# --- Filters ---
cadena = "Hola, hola mundo Rio do Janeiro!"
print(cadena.count("Rio",0,18))             # 1
print(cadena.index("ne"))                   #21   -1 si no encuentra
print(cadena.rindex('Ja'))                  #19   -1. Derecha izquierda
print(cadena.find('All'))                   #11   ValueError si no
print(cadena.rfind('l'))                    #13   Derecha a izquierda
# --- Replace ---
new_cadena = cadena.replace("mundo", "Python")  #Hola, Python Rio do Janeiro!
cocoa=cadena+new_cadena                     #Las junta
# --- Fill and Adjust ---
cadena14 = "123"
cadena_rellenada = cadena14.zfill(5)                # 00123
cadena1 = "Hola"
cadena_centralizada = cadena1.center(10, '-')       # ---Hola---
cadena_jusstificada = cadena1.ljust(10, '-')        # Hola------
cadena_justificada_derecha = cadena1.rjust(10, '-')     # ------Hola
# --- Split & Join---
cadena2 = "Hola\nMundo\nFeliz"
sliced_company = cadena2.split('\n', 1)[1]  # Feliz  marca por donde cortar (",") (" ")
("#"), [1] la tajada
subcadenas = cadena2.rsplit('\n', 1)        # ['Hola\nMundo', 'Feliz'] 1 nº de tajadas
lineas = cadena2.splitlines()               # ['Hola', 'Mundo', 'Feliz']
holita = "HolaholitaMundo"
```

```
particion = holita.partition('a')           # ('Hol', 'a', 'holitaMundo')
particion_ = holita.rpartition('a')         # ('Holaholit', 'a', 'Mundo')
acronym= ''.join(word[0] for word in company.split())    # CFA acronym
# --- Translate ---
tabla = str.maketrans('aeiou', '12345', 'xyz')
cadena6 = "Hola Mundo"
cadena_traducida = cadena6.translate(tabla)              # H4l1 M5nd4
cadena_codificada = cadena1.encode(encoding='utf-8')     # b'Hola' encode bytes

#--- Booleans Methods---
isinstance("Paco", (str,int))      # True
cadena = "especifico para cadenas Strings"

print(cadena.isalpha())          # False, ya que la cadena contiene caracteres que no son solo letras.
print(cadena.isalnum())          # True, porque la cadena contiene solo letras y números.
print(cadena.isnumeric())        # False, porque la cadena contiene caracteres que no son solo numéricos.
print(cadena.isdigit())          # False, ya que la cadena contiene caracteres que no son solo dígitos.
print(cadena.isdecimal())        # False, ya que la cadena contiene caracteres que no son decimales.
print(cadena.islower())          # False, porque la cadena contiene al menos una letra en mayúscula.
print(cadena.isupper())          # False, porque la cadena contiene al menos una letra en minúscula.
print(cadena.isspace())          # False, porque la cadena no contiene solo espacios en blanco.
print(cadena.isascii())          # True, porque todos los caracteres de la cadena son ASCII.
print(cadena.isprintable())      # True, porque todos los caracteres de la cadena son imprimibles.
print(cadena.startswith("Ofi"))  # True, porque la cadena comienza con "Ofi".
print(cadena.endswith("2020"))   # True, porque la cadena termina con "2020".
print('30DaysOfPython'.isidentifier()) # False: Comenzar con letra o guion bajo, sin palabras reservadas
print('thirty_days_of_python___'.isidentifier())# True: Cumple con las reglas de un identificador

#----------Email_validator---------
import re

def email_validator(email):
    # Expresión regular para validar dirección de correo electrónico
    patron = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}VSCODE_PRINT_CONTENT#x27;

    # Comprueba si la dirección de correo coincide con el patrón
    if re.match(patron, email):
        return True
    else:
        return False
#---------Palabras unicas-----------
frase = " Cuñao, hola Hola HOLA, soy Edu, sOy feliz navidad"
set_ = set()
frase_unica = []

```

```
97  for palabra in frase.split():
98      palabra_minusculas = palabra.casefold()
99      if palabra_minusculas not in set_:
100         set_.add(palabra_minusculas)
101         frase_unica.append(palabra)
102
103 print(" ".join(frase_unica))
```