T13_BibliotecaEstandar

January 29, 2021

1 Python de cero a experto

Autor: Luis Miguel de la Cruz Salas

1.1 Pythonico es más bonito: Pensando como pythonista (intermedio)

1.1.1 Biblioteca estándar

Módulos de funciones matemáticas `math` y `cmath`

- El módulo **math** provee de las funciones matemáticas definida en la biblioteca estándar de C para números **reales**.
- Más información: https://docs.python.org/3.0/library/math.html
- El módulo **cmath** provee de las funciones matemáticas para números **complejos**.
- Más información: https://docs.python.org/3.0/library/cmath.html

[1]:
```
import math
print(math.sqrt(4))
print(math.sin(90))
```

```
2.0
0.8939966636005579
```

[2]:
```
complejo = 1 + 4j
print(complejo)
```

```
(1+4j)
```

[3]:
```
import cmath
print(cmath.sin(complejo))
```

```
(22.979085577886128+14.744805188558725j)
```

`random`

- https://docs.python.org/3/library/random.html

[4]:
```
import random
print(random.randint(0, 5))
print(random.random() * 100)
myList = [2, 109, False, 10, "Lorem", 482, "Ipsum"]
print(random.choice(myList))
x = [[i] for i in range(10)]
random.shuffle(x)
print(x)
for i in range(3):
    print(random.randrange(0, 101, 5))
```

```
5
2.11527255274353
109
[[3], [4], [9], [1], [8], [7], [0], [5], [2], [6]]
80
55
75
```

`statistics`

- https://docs.python.org/3/library/statistics.html

[5]:
```
import statistics

example_list = [5,2,5,6,1,2,6,7,2,6,3,5,5]

x = statistics.mean(example_list)
print(x)

y = statistics.median(example_list)
print(y)

z = statistics.mode(example_list)
print(z)

a = statistics.stdev(example_list)
print(a)

b = statistics.variance(example_list)
print(b)
```

```
4.230769230769231
5
5
1.9644272343292228
3.858974358974359
```

Servicios `os`

- Este módulo provee una manera portable de interactuar con el sistema operativo.
- Más información: https://docs.python.org/3/library/os.html
- Se puede por ejemplo, crear, eliminar y mover carpetas/archivos, cambiarse de directorio, acceder a los nombres de archivos dentro de una ruta, etc.

[6]:
```
import os

directorio_actual = os.getcwd()
print(directorio_actual)
```

```
/home/luiggi/GitSites/Cursos/Python_cero_a_experto
```

[7]:
```
os.mkdir('nueva_carpeta')
```

[8]:
```
os.chdir('nueva_carpeta')
```

[9]:
```
print(os.getcwd())
```

```
/home/luiggi/GitSites/Cursos/Python_cero_a_experto/nueva_carpeta
```

[10]:
```
os.chdir('../')
```

[11]:
```
print(os.getcwd())
```

```
/home/luiggi/GitSites/Cursos/Python_cero_a_experto
```

[12]:
```
os.rename('nueva_carpeta','nueva')
```

[13]:
```
os.listdir()
```

[13]:
```
['nombres',
 'mi_archivo.txt',
 'T09_LambdaExpressions_Reduce.ipynb',
 'VersionPDF',
 '.git',
 'Presentaciones',
 'T13_BibliotecaEstandar.ipynb',
 'T08_IterablesMapFilter.ipynb',
 'T07_Excepciones.ipynb',
 'T10_Comprehensions.ipynb',
 'T01_Etiquetas_y_Palabras_Reservadas.ipynb',
 'QueLesQuedaALosJovenes.txt',
 'AtractorDeLorenz.ipynb',
 '04_PythonCientifico',
 '.ipynb_checkpoints',
 'T02_Expr_Decla_Tipos_Oper.ipynb',
 'Figuras',
```

```
 'T14_Numpy.ipynb',
 'T12_Decoradores.ipynb',
 'Proyectos',
 '01_Pensando_como_pythonista_1.ipynb',
 '04_ComputoCientifico.ipynb',
 'T03_Estructura_de_Datos.ipynb',
 'gatos.txt',
 'tabla',
 'nueva',
 '02_Pythonico_es_mas_bonito_1.ipynb',
 'T11_IteradoresGeneradores.ipynb',
 'T06_Funciones_y_Documentacion.ipynb',
 'T04_Control_de_flujo.ipynb',
 'README.md',
 'T05_Entrada_Salida_Archivos.ipynb',
 '03_Pythonico_es_mas_bonito_2.ipynb']
```

[14]:
```
os.rmdir('nueva')
```

[15]:
```
os.listdir()
```

[15]:
```
['nombres',
 'mi_archivo.txt',
 'T09_LambdaExpressions_Reduce.ipynb',
 'VersionPDF',
 '.git',
 'Presentaciones',
 'T13_BibliotecaEstandar.ipynb',
 'T08_IterablesMapFilter.ipynb',
 'T07_Excepciones.ipynb',
 'T10_Comprehensions.ipynb',
 'T01_Etiquetas_y_Palabras_Reservadas.ipynb',
 'QueLesQuedaALosJovenes.txt',
 'AtractorDeLorenz.ipynb',
 '04_PythonCientifico',
 '.ipynb_checkpoints',
 'T02_Expr_Decla_Tipos_Oper.ipynb',
 'Figuras',
 'T14_Numpy.ipynb',
 'T12_Decoradores.ipynb',
 'Proyectos',
 '01_Pensando_como_pythonista_1.ipynb',
 '04_ComputoCientifico.ipynb',
 'T03_Estructura_de_Datos.ipynb',
 'gatos.txt',
 'tabla',
 '02_Pythonico_es_mas_bonito_1.ipynb',
```

```
 'T11_IteradoresGeneradores.ipynb',
 'T06_Funciones_y_Documentacion.ipynb',
 'T04_Control_de_flujo.ipynb',
 'README.md',
 'T05_Entrada_Salida_Archivos.ipynb',
 '03_Pythonico_es_mas_bonito_2.ipynb']
```

[16]:
```python
import os

def child():
    print('\nA new child ',  os.getpid())
    os._exit(0)

def parent():
    while True:
        newpid = os.fork()
        if newpid == 0:
            child()
        else:
            pids = (os.getpid(), newpid)
            print("parent: %d, child: %d\n" % pids)
        reply = input("q for quit / c for new fork")
        if reply == 'c':
            continue
        else:
            break

parent()
```

```
parent: 19343, child: 19354


A new child  19354
q for quit / c for new fork c
parent: 19343, child: 19363


A new child  19363
q for quit / c for new fork c
parent: 19343, child: 19372


A new child  19372
q for quit / c for new fork a
```

platform

- https://docs.python.org/3/library/platform.html

[19]:
```
import os, platform
print(platform.system())

if platform.system() == "Windows":
    import msvcrt

def getch():
    if platform.system() == "Darwin" or "Linux":
        os.system("bash -c \"read -n 1\"")
    else:
        msvcrt.getch()

print("Type a key!")
getch()
print("Okay")
```

```
Linux
Type a key!
Okay
```

shutil

- https://docs.python.org/3/library/shutil.html

[20]:
```
import os, shutil

dir_original = os.getcwd()
lista_archivos = os.listdir(dir_original)
os.mkdir('BORRAME')
dir_destino = dir_original + '/BORRAME'
print(dir_destino)
for files in lista_archivos:
    if files.endswith(".txt"):
        shutil.copy(files, dir_destino)
os.listdir('./BORRAME/')
```

```
/home/luiggi/GitSites/Cursos/Python_cero_a_experto/BORRAME
```

[20]:
```
['mi_archivo.txt', 'QueLesQuedaALosJovenes.txt', 'gatos.txt']
```

[21]:
```
os.system('rm -rf ./BORRAME/')
```

[21]:
```
0
```

sys

- Provee información acerca de las constantes, funciones y métodos del interprete de Python.
  - https://docs.python.org/3/library/sys.html

[22]:
```
help()
```

```
Welcome to Python 3.8's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.8/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".


help>  print

Help on built-in function print in module builtins:

print(…)
    print(value, …, sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.


help>  quit


You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)".  Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
```

[24]:
```
import sys
help(sys)
```

[25]:
```
sys.version
```

[25]:
```
'3.8.5 (default, Sep  4 2020, 07:30:14) \n[GCC 7.3.0]'
```

[26]:
```
sys.float_info
```

[26]:
```
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308,
min=2.2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15,
mant_dig=53, epsilon=2.220446049250313e-16, radix=2, rounds=1)
```

[27]:
```
sys.path
```

[27]:
```
['/home/luiggi/GitSites/Cursos/Python_cero_a_experto',
 '/home/luiggi/anaconda3/lib/python38.zip',
 '/home/luiggi/anaconda3/lib/python3.8',
 '/home/luiggi/anaconda3/lib/python3.8/lib-dynload',
 '',
 '/home/luiggi/anaconda3/lib/python3.8/site-packages',
 '/home/luiggi/anaconda3/lib/python3.8/site-packages/IPython/extensions',
 '/home/luiggi/.ipython']
```

[28]:
```
print("Impresión a la salida estándar")

# Salvamos la salida estándar
save_stdout = sys.stdout

fh = open("test_salida_estandar.txt","w")

# Cambiamos la salida estándar
sys.stdout = fh
print("Esta línea va hacia el archivo test_salida_estandar.txt")

# Regresamos la salida estandar a la normalidad
sys.stdout = save_stdout

fh.close()
```

```
Impresión a la salida estándar
```

[29]:
```
sys.stderr
```

[29]:
```
<ipykernel.iostream.OutStream at 0x7f8992b3de50>
```

`time` y `datetime`

- `time` : Acceso al tiempo del sistema y conversiones
  - https://docs.python.org/3/library/time.html
- `datetime` Tipos básicos para el tiempo y la fecha
  - https://docs.python.org/3/library/datetime.html

[30]:
```
import time
import datetime
```

```
print("Time in seconds since the epoch: %s" %time.time())
print("Current date and time: " , datetime.datetime.now())
print("Or like this: " ,datetime.datetime.now().strftime("%y-%m-%d-%H-%M"))


print("Current year: ", datetime.date.today().strftime("%Y"))
print("Month of year: ", datetime.date.today().strftime("%B"))
print("Week number of the year: ", datetime.date.today().strftime("%W"))
print("Weekday of the week: ", datetime.date.today().strftime("%w"))
print("Day of year: ", datetime.date.today().strftime("%j"))
print("Day of the month : ", datetime.date.today().strftime("%d"))
print("Day of week: ", datetime.date.today().strftime("%A"))
```

```
Time in seconds since the epoch: 1611952926.0749996
Current date and time:  2021-01-29 14:42:06.075160
Or like this:  21-01-29-14-42
Current year:  2021
Month of year:  January
Week number of the year:  04
Weekday of the week:  5
Day of year:  029
Day of the month :  29
Day of week:  Friday
```

[31]:
```
hoy = datetime.date.today()
nacimiento = datetime.date(1970, 8, 13)
edad = hoy - nacimiento
edad.days
```

[31]:
```
18432
```

[32]:
```
from timeit import Timer
print(Timer('t=a; a=b; b=t', 'a=1; b=2').timeit())
print(Timer('a,b = b,a', 'a=1; b=2').timeit())
```

```
0.021900733998336364
0.01797821299987845
```

glob

- https://docs.python.org/3/library/glob.html

[35]:
```
import os, glob
metadata = [(f, os.stat(f)) for f in glob.glob('*.ipynb')]
metadata
```

```
[35]: [('T09_LambdaExpressions_Reduce.ipynb',
         os.stat_result(st_mode=33204, st_ino=4459373, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=17198, st_atime=1611952490,
       st_mtime=1611952490, st_ctime=1611952490)),
        ('T13_BibliotecaEstandar.ipynb',
         os.stat_result(st_mode=33204, st_ino=4459377, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=23138, st_atime=1611952773,
       st_mtime=1611952894, st_ctime=1611952894)),
        ('T08_IterablesMapFilter.ipynb',
         os.stat_result(st_mode=33204, st_ino=4459372, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=10100, st_atime=1611952369,
       st_mtime=1611952369, st_ctime=1611952369)),
        ('T07_Excepciones.ipynb',
         os.stat_result(st_mode=33204, st_ino=4459371, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=11368, st_atime=1611952319,
       st_mtime=1611952319, st_ctime=1611952319)),
        ('T10_Comprehensions.ipynb',
         os.stat_result(st_mode=33204, st_ino=4459374, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=23143, st_atime=1611952659,
       st_mtime=1611952659, st_ctime=1611952659)),
        ('T01_Etiquetas_y_Palabras_Reservadas.ipynb',
         os.stat_result(st_mode=33204, st_ino=4459357, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=7269, st_atime=1611949503,
       st_mtime=1611949503, st_ctime=1611949503)),
        ('AtractorDeLorenz.ipynb',
         os.stat_result(st_mode=33204, st_ino=1968285, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=84714, st_atime=1611950997,
       st_mtime=1611942389, st_ctime=1611942389)),
        ('T02_Expr_Decla_Tipos_Oper.ipynb',
         os.stat_result(st_mode=33204, st_ino=4459358, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=29065, st_atime=1611949512,
       st_mtime=1611949512, st_ctime=1611949512)),
        ('T14_Numpy.ipynb',
         os.stat_result(st_mode=33204, st_ino=4459427, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=26171, st_atime=1611856934,
       st_mtime=1611856625, st_ctime=1611856625)),
        ('T12_Decoradores.ipynb',
         os.stat_result(st_mode=33204, st_ino=4459376, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=10722, st_atime=1611952771,
       st_mtime=1611952771, st_ctime=1611952771)),
        ('01_Pensando_como_pythonista_1.ipynb',
         os.stat_result(st_mode=33204, st_ino=4459165, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=14699, st_atime=1611948889,
       st_mtime=1611946215, st_ctime=1611946215)),
        ('04_ComputoCientifico.ipynb',
         os.stat_result(st_mode=33204, st_ino=4459368, st_dev=2057, st_nlink=1,
       st_uid=1000, st_gid=1000, st_size=80492, st_atime=1611942139,
```

```
st_mtime=1611942139, st_ctime=1611942139)),
 ('T03_Estructura_de_Datos.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459359, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=20255, st_atime=1611950028,
st_mtime=1611950028, st_ctime=1611950028)),
 ('02_Pythonico_es_mas_bonito_1.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459340, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=18342, st_atime=1611950966,
st_mtime=1611950966, st_ctime=1611950966)),
 ('T11_IteradoresGeneradores.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459375, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=11550, st_atime=1611952707,
st_mtime=1611952707, st_ctime=1611952707)),
 ('T06_Funciones_y_Documentacion.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459370, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=15895, st_atime=1611952273,
st_mtime=1611952273, st_ctime=1611952273)),
 ('T04_Control_de_flujo.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459360, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=9473, st_atime=1611950186,
st_mtime=1611950186, st_ctime=1611950186)),
 ('T05_Entrada_Salida_Archivos.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459369, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=9589, st_atime=1611950501,
st_mtime=1611950500, st_ctime=1611950500)),
 ('03_Pythonico_es_mas_bonito_2.ipynb',
  os.stat_result(st_mode=33204, st_ino=4459400, st_dev=2057, st_nlink=1,
st_uid=1000, st_gid=1000, st_size=3691, st_atime=1611950981,
st_mtime=1611787640, st_ctime=1611787640))]
```

[36]:
```
metadata_dict = {f:os.stat(f) for f in glob.glob('*.ipynb')}
```

[37]:
```
metadata_dict.keys()
```

[37]:
```
dict_keys(['T09_LambdaExpressions_Reduce.ipynb', 'T13_BibliotecaEstandar.ipynb',
'T08_IterablesMapFilter.ipynb', 'T07_Excepciones.ipynb',
'T10_Comprehensions.ipynb', 'T01_Etiquetas_y_Palabras_Reservadas.ipynb',
'AtractorDeLorenz.ipynb', 'T02_Expr_Decla_Tipos_Oper.ipynb', 'T14_Numpy.ipynb',
'T12_Decoradores.ipynb', '01_Pensando_como_pythonista_1.ipynb',
'04_ComputoCientifico.ipynb', 'T03_Estructura_de_Datos.ipynb',
'02_Pythonico_es_mas_bonito_1.ipynb', 'T11_IteradoresGeneradores.ipynb',
'T06_Funciones_y_Documentacion.ipynb', 'T04_Control_de_flujo.ipynb',
'T05_Entrada_Salida_Archivos.ipynb', '03_Pythonico_es_mas_bonito_2.ipynb'])
```

[39]:
```
metadata_dict['T13_BibliotecaEstandar.ipynb'].st_size
```

[39]:
```
23138
```

urllib

- https://docs.python.org/3/library/urllib.html

[43]:
```
import urllib.request

x = urllib.request.urlopen('http://gmc.geofisica.unam.mx/')
print(x.read())
```

```
b'<html>\n<head>\n<title>PAPIME</title>\n<META HTTP-EQUIV="REFRESH"
CONTENT="0;URL=http://gmc.geofisica.unam.mx/papime2020/">
</head>\n<body>\n</body>\n</html>\n'
```

Misceláneos doctest

[44]:
```
def promedio(valores):
    """Calcula la media aritmética de una lista de números.

    >>> print(promedio([20, 30, 70]))
    40.0
    """
    return sum(valores) / len(valores)

import doctest
doctest.testmod()   # valida automáticamente las pruebas integradas
```

[44]:
```
TestResults(failed=0, attempted=1)
```

[45]:
```
def fib(n):
    """
    Calculates the n-th Fibonacci number iteratively

    >>> fib(0)
    0
    >>> fib(1)
    1
    >>> fib(10)
    55
    >>> fib(15)
    610
    """
    a, b = 0, 1
    for i in range(n):
        a, b = b, a + b
    return a
```

[46]:
```
doctest.testmod()
```

[46]:
```
TestResults(failed=0, attempted=5)
```

zlib

[47]:
```
import zlib
s = b'witch which has which witches wrist watch'
len(s)
```

[47]:
```
41
```

[48]:
```
t = zlib.compress(s)
len(t)
```

[48]:
```
37
```

[49]:
```
zlib.decompress(t)
```

[49]:
```
b'witch which has which witches wrist watch'
```

[50]:
```
f = open('./QueLesQuedaALosJovenes.txt', 'rb')
original_data = f.read()
f.close()
print(original_data)
compressed_data = zlib.compress(original_data, zlib.Z_BEST_COMPRESSION)

compress_ratio = (float(len(original_data)) - float(len(compressed_data))) /␣
 ↪float(len(original_data))

print('Compressed: %d%%' % (100.0 * compress_ratio))
f = open('file.zip','wb')
f.write(compressed_data)
f.close()
```

```
b'\xc2\xbfQu\xc3\xa9 les queda por probar a los j\xc3\xb3venes\nen este mundo de
paciencia y asco?\n\xc2\xbfs\xc3\xb3lo grafitti? \xc2\xbfrock?
\xc2\xbfescepticismo?\ntambi\xc3\xa9n les queda no decir am\xc3\xa9n\nno dejar
que les maten el amor\nrecuperar el habla y la utop\xc3\xada\nser j\xc3\xb3venes
sin prisa y con memoria\nsituarse en una historia que es la suya\nno convertirse
en viejos prematuros\n\xc2\xbfqu\xc3\xa9 les queda por probar a los
j\xc3\xb3venes\nen este mundo de rutina y ruina?\n\xc2\xbfcoca\xc3\xadna?
\xc2\xbfcerveza? \xc2\xbfbarras bravas?\nles queda respirar / abrir los
ojos\ndescubrir las ra\xc3\xadces del horror\ninventar paz as\xc3\xad sea a
ponchazos\nentenderse con la naturaleza\ny con la lluvia y los
rel\xc3\xa1mpagos\ny con el sentimiento y con la muerte\nesa loca de atar y
desatar\n\xc2\xbfqu\xc3\xa9 les queda por probar a los j\xc3\xb3venes\nen este
mundo de consumo y humo?\n\xc2\xbfv\xc3\xa9rtigo? \xc2\xbfasaltos?
\xc2\xbfdiscotecas?\ntambi\xc3\xa9n les queda discutir con dios\ntanto si existe
como si no existe\ntender manos que ayudan / abrir puertas\nentre el
```

```
coraz\xc3\xb3n propio y el ajeno /\nsobre todo les queda hacer futuro\na pesar
de los ruines de pasado\ny los sabios granujas del presente\n'
Compressed: 49%
```

[51]:
```
f = open('./file.zip', 'rb')
original_data = f.read()
f.close()
decompressed_data = zlib.decompress(original_data)
print((decompressed_data))
```

```
b'\xc2\xbfQu\xc3\xa9 les queda por probar a los j\xc3\xb3venes\nen este mundo de
paciencia y asco?\n\xc2\xbfs\xc3\xb3lo grafitti? \xc2\xbfrock?
\xc2\xbfescepticismo?\ntambi\xc3\xa9n les queda no decir am\xc3\xa9n\nno dejar
que les maten el amor\nrecuperar el habla y la utop\xc3\xada\nser j\xc3\xb3venes
sin prisa y con memoria\nsituarse en una historia que es la suya\nno convertirse
en viejos prematuros\n\xc2\xbfqu\xc3\xa9 les queda por probar a los
j\xc3\xb3venes\nen este mundo de rutina y ruina?\n\xc2\xbfcoca\xc3\xadna?
\xc2\xbfcerveza? \xc2\xbfbarras bravas?\nles queda respirar / abrir los
ojos\ndescubrir las ra\xc3\xadces del horror\ninventar paz as\xc3\xad sea a
ponchazos\nentenderse con la naturaleza\ny con la lluvia y los
rel\xc3\xa1mpagos\ny con el sentimiento y con la muerte\nesa loca de atar y
desatar\n\xc2\xbfqu\xc3\xa9 les queda por probar a los j\xc3\xb3venes\nen este
mundo de consumo y humo?\n\xc2\xbfv\xc3\xa9rtigo? \xc2\xbfasaltos?
\xc2\xbfdiscotecas?\ntambi\xc3\xa9n les queda discutir con dios\ntanto si existe
como si no existe\ntender manos que ayudan / abrir puertas\nentre el
coraz\xc3\xb3n propio y el ajeno /\nsobre todo les queda hacer futuro\na pesar
de los ruines de pasado\ny los sabios granujas del presente\n'
```

[ ]:
```

```