

Práctica de laboratorio: Realice el desafío de Python

Este es un ejercicio opcional para probar su conocimiento de los principios básicos de Python. Sin embargo, recomendamos fervientemente que el estudiante complete estos ejercicios para prepararse para el resto de este curso. Si no sabe cómo resolverlos, fíjese en las lecciones de Python disponibles en la carpeta de Materiales del curso/tutoriales y demostraciones.

```
import os
cwd=os.getcwd()
print(cwd)
```

/drive/notebooks/cuaderno

Responda las preguntas o complete las tareas detalladas a continuación; utilice el método específico descrito, si corresponde.

1) ¿Cuánto es 3 a la potencia de 5?

```
# Code cell 1
3**5
```

243

2) Cree una variable, 's', que contenga la cadena "¡Este curso es increíble!". Con la variable, divida la cadena en una lista.

```
# Code cell 2
s = "¡Este curso es increíble!"
palabras=s.split()
print(palabras)
```

```
['¡Este', 'curso', 'es', 'increíble!']
```

3) Dadas las variables altura y montaña, use .format() para imprimir la cadena siguiente:

```
# Code cell 3
mountain = "Mt. Everest"
height = 8848
#print("...".format( ? , ? ))
print("La altura del {} es de {}".format(mountain,height))
```

La altura del Mt. Everest es de 8848.

4) Dada la lista anidada siguiente, use la indexación para tomar la palabra "esto".

Code cell 4

```
lst = ['a', 'b', [4, 10, 11], ['c', [1, 66, ['this']], 2, 111], 'e', 7]
print(lst[3][1][2][0])
```

this

5) Dado el diccionario anidado siguiente, tome la palabra "eso". Este ejercicio es un poco más difícil.

Code cell 5

```
d = {'k1': ['val1', 'val2', 'val3', {'we': ['need', 'to', 'go', {'deeper': [1, 2, 3, 'that']}]}]}
that=d['k1'][3]['we'][3]['deeper'][3]
print(that)
```

that

6) ¿Cuál es la diferencia principal entre una tupla y una lista?

listas: dinámicas si permite crud, usamos [], menos rápidas
tuplas: estáticas, no permite crud, usamos (), más rápidas

7) Cree una función, GetDomain(), que tome el dominio del sitio web de correo electrónico de una cadena en la forma: 'user@domain.com'.

Por ejemplo, el paso de "user@domain.com" daría: domain.com

```
def getDomain(email):
    cadenas=email.split("@")
    return cadenas[1]
```

```
getDomain("user@domain.com")
```

'domain.com'

8) Cree una función básica, findInternet(), que dé una devolución de True si la palabra 'Internet' se incluye en la cadena de entrada. No se preocupe por los casos de perímetro como la puntuación que se asocia con la palabra, pero tenga en cuenta el uso de mayúsculas.

(Sugerencia: vea <https://docs.python.org/2/reference/expressions.html#in>)

```
def findInternet(texto):
    textoLower= texto.lower()
    if textoLower.find("internet")!=-1:
        return True
    else:
        return False
```

```
findInternet("The Internet Engineering Task Force was created in 1986")
```

True

9) Cree una función, `countIoT()`, que cuente la cantidad de veces que la palabra "IoT" aparece en una cadena. Ignore los casos de perímetro pero tenga en cuenta el uso de mayúsculas.

```
def countIoT(texto):  
    #textoLower= texto.lower()  
    return texto.count("IoT")+texto.count("iot")
```

```
countIoT('I don\'t know how to spell IoT ! Is it IoT or iot ? What  
does iot mean anyway?')  
#countIoT('I don\'t know how to spell IoT ! Is it IoT')
```

4

10) Utilice las expresiones `lambda` y la función `filter()` para filtrar las palabras de una lista que no comiencen con la letra 'd'. Por ejemplo:

```
sec = ["datos", "sal", "diario", "gato", "perro"]
```

debe ser filtrado a:

```
['datos', 'diario']
```

```
sec = ["datos", "sal", "diario", "gato", "perro"]  
filtrado = list(filter(lambda x: x.startswith("d")==True, sec))  
print(filtrado)  
#debe ser filtrado a: ['datos', 'diario']
```

```
['datos', 'diario']
```

11) Utilice las expresiones `lambda` y la función `map()` para convertir una lista de palabras a mayúsculas. Por ejemplo:

```
sec = ["datos", "sal", "diario", "gato", "perro"]
```

debe ser:

```
["DATOS", "SAL", "DIARIO", "GATO", "PERRO"]
```

```
sec = ["datos", "sal", "diario", "gato", "perro"]
```

```
#debe ser:  
#[ "DATOS", "SAL", "DIARIO", "GATO", "PERRO"]  
sec2 = list(map(lambda x: x.upper(), sec))  
print(sec2)
```

```
['DATOS', 'SAL', 'DIARIO', 'GATO', 'PERRO']
```

12) Imagine un termostato inteligente conectado a la puerta que pueda detectar, además de la temperatura, el momento en el que las personas entran o salen de la casa.

Escriba una función que, cuando la temperatura sea inferior a 20 °C y haya personas en la casa (codificado como valor booleano que se envía como parámetro a la función), inicie la calefacción mediante la devolución de la cadena "calefacción encendida". Cuando la

temperatura llegue a 23 grados o no haya personas en la casa, la función devuelve la cadena "calefacción apagada". Cuando no se cumpla ninguna de estas condiciones, la función es "No hacer nada".

```
def smart_thermostat(temp, people_in):  
#  
    ...  
    if temp<20 and people_in==True:  
        print("calefacción encendida")  
    elif temp==23 or people_in==False:  
        print("calefacción apagada")  
    else:  
        print("Sin funciones activadas ;)")
```

```
smart_thermostat(21,True)  
smart_thermostat(10,False)  
smart_thermostat(23,True)  
smart_thermostat(10,True)  
smart_thermostat(21,False)
```

```
Sin funciones activadas ;)  
calefacción apagada  
calefacción apagada  
calefacción encendida  
calefacción apagada
```

13) La función zip(list1, list2) devuelve una lista de tuplas, donde la tupla i-th contiene el elemento i-th de cada una de las listas de argumento. Utilice la función zip para crear la siguiente lista de tuplas:

```
'comprimido' = [(“Estacionamiento”, -1), (“Negocios”, 0), (“Área de restaurantes”, 1),  
                (“oficinas”, 2)]
```

```
# Code cell 14  
floor_types = ['Parking', 'Shops', 'Food Court', 'Offices']  
floor_numbers = range(-1,3)  
zipped = zip(floor_types,floor_numbers)  
print(zipped)  
for z in zipped:  
    print(z)
```

```
<zip object at 0x2237428>  
( 'Parking', -1)  
( 'Shops', 0)  
( 'Food Court', 1)  
( 'Offices', 2)
```

14) Utilice la función zip y dict() para crear un diccionario, elevator_dict, donde las teclas sean los tipos de piso y los valores sean el número correspondiente del piso, de modo que:

```
elevator_dict[- 1] = “Estacionamiento”
```

```

floor_types = ['Parking', 'Shops', 'Food Court', 'Offices']
floors_numbers = range(-1,3)
z=zip(floors_numbers,floor_types)

```

```

elevator_dict = dict(z)
elevator_dict[-1]
#elevator_dict[0]

```

'Parking'

15) Cree una clase de 'Ascensor'. El constructor acepta la lista de cadenas 'floor_types' y la lista de números enteros 'floor_numbers'. La clase implementa los métodos 'ask_which_floor' y 'go_to_floor'. La salida de estos métodos debe verse de la siguiente manera:

```

`floor_types = ['Estacionamiento', 'Negocios', 'Área de restaurantes', 'Oficinas']
floors_numbers = rango(-1,4)

```

```

el = Elevador(floor_numbers, floor_types)

```

```

el.go_to_floor(1)`

```

¡Vaya al piso del área de restaurantes!

```

el.go_to_floor(-2)

```

En este edificio está el piso número -2.

```

El.ask_which_floor('Oficinas')

```

El piso de oficinas es el número: 2

```

El.ask_which_floor('Piscina')

```

No hay ningún piso con piscina en este edificio.

```

class Elevator:

```

```

    def __init__(self, floor_numbers, floor_types):
        self._floor_numbers = floor_numbers
        self._floor_types = floor_types
        self._number_to_type_dict = dict(zip(floor_numbers,
floor_types))
        self._type_to_number_dict = dict(zip(floor_types,
floor_numbers))
        self.mivariable=0

```

```

    def ask_which_floor(self, floor_type):
        if floor_type in self._floor_types:
            print('The {} floor is the number: {}'.format(floor_type,
self._type_to_number_dict[floor_type]))
        else:

```

```

        print('There is no {} floor in this
building.'.format(floor_type))

    #def go_to_floor(self, floor_number):
    def go_to_floor(self, floor_number):
        if floor_number in self._floor_numbers:
            print('floor number:{} so Go to {}'.format(floor_number,
self._number_to_type_dict [floor_number]))
        else:
            print('There is no such floor number: {}  in this
building.'.format(floor_number))

```

```

floor_types = ['Parking', 'Shops', 'Food Court', 'Offices']
floors_numbers = range(-1,3)
miElevator=Elevator(floors_numbers,floor_types)
miElevator.go_to_floor(1)
miElevator.ask_which_floor("Offices")
miElevator.go_to_floor(-2)
miElevator.ask_which_floor("Swimming Pool")

```

```

floor number:1 so Go to Food Court.
The Offices floor is the number: 2.
There is no such floor number: -2  in this building.
There is no Swimming Pool floor in this building.

```

```

# Verify code cell 18
el = Elevator(floor_numbers, floor_types)
el.go_to_floor(1)

```

```

floor number:1 so Go to Food Court.

```

```

# Verify code cell 19
el.go_to_floor(-2)

```

```

There is no such floor number: -2  in this building.

```

```

# Verify code cell 20
el.ask_which_floor('Offices')

```

```

The Offices floor is the number: 2.

```

```

# Verify code cell 21
el.ask_which_floor('Swimming Pool')

```

```

There is no Swimming Pool floor in this building.

```

¡Excelente trabajo!

yeah !!! julix julian.galindo@epn.edu.ec

© 2017 Cisco y/o sus filiales. Todos los derechos reservados. Este documento es información pública de Cisco.