

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.

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?

```
In [1]: # Code cell 1
3**5
```

```
Out[1]: 243
```

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

```
In [2]: # Code cell 2
#s = ?
#?.split()

s="Este curos es increíble! "
lista=s.split()
lista
```

```
Out[2]: ['Este', 'curos', 'es', 'increíble!']
```

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

‘La altura del Monte Everest es de 8848 metros’.

```
In [3]: # Code cell 3
mountain = "Mt. Everest"
height = 8848
#print("..."format( ? , ? ))

print ("La altura del {0} es de {1}.".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".

```
In [4]: # Code cell 4
lst = ['a', 'b', [4, 10, 11], ['c', [1, 66, ['this']], 2, 111], 'e', 7]
#lst[?][?][?][?]

print(lst[3])

['c', [1, 66, ['this']], 2, 111]
```

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

```
In [ ]: # Code cell 5
d = {'k1': ['val1', 'val2', 'val3'], {'we': ['need', 'to', 'go'], {'deeper': [1, 2, 3, 'that']}}}
#d[?][?][?][?][?][?]
```

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

La lista se puede modificar y la tupla no puede modificarse

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

```
In [24]: # Code cell 6

import re
domain = re.search("@[\w.]+", correo)
print(domain.group())
GetDomain('user@domain.com')
```

```
File "C:\Users\camil\AppData\Local\Temp\ipykernel_27344\3720170672.py", line 3
    import re GetDomain(correo)
            ^
SyntaxError: invalid syntax
```

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>)

```
In [12]: # Code cell 7
def findInternet(cadena):
    if cadena.find("Internet")>0:
        print("Se ha encontrado la palabra internet en la cadena")
    else:
        print("No se ha encontrad ola palabra internet en la cadena")
findInternet('The Internet Engineering Task Force was created in 1986')
```

Se ha encontrado la palabra internet en la cadena

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.

```
In [13]: # Code cell 8
def countIoT(cadena):
    print("La palabra IoT se repite",cadena.count('IoT'),"veces")
countIoT('I don\'t know how to spell IoT ! Is it IoT or iot ? What does iot mean anywa
```

La palabra IoT se repite 2 veces

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']
```

```
In [14]: # Code cell 9
```

```
File "C:\Users\camil\AppData\Local\Temp\ipykernel_27344\2119845683.py", line 3
    mayusculas=list(map(lambda x:x.upper(),sec))
                        ^
```

SyntaxError: invalid syntax

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"]
```

```
In [20]: # Code cell 10
#list(map())
```

```
sec=['datos', 'sal', 'lácteos', 'gato', 'perro']
mayusculas=list(map(lambda x:x.upper(),sec))
print(mayusculas)
```

```
['DATOS', 'SAL', 'LÁCTEOS', '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".

```
In [27]: # Code cell 11
temperatura=int(input("İngrese la temperatura: "))
personas= (input("İngrese la temperatura: "))
def termostato(temp, people_in):
    if(temperatura<20 and personas =='true'):
        print("Calefaccion encendida")
    if(temperatura==23 or personas =='false'):
        print("Calefaccion apagada")
    else:print("No hacer nada")
termostato(temperatura,personas)
# ...
# return command
```

```
İngrese la temperatura: 24
İngrese la temperatura: 25
No hacer nada
```

```
In [29]: # Code cell 12
Verify termostato()

termostato(21,True)
```

```
File "C:\Users\camil\AppData\Local\Temp\ipykernel_27344\3059742909.py", line 2
    Verify termostato()
    ^
SyntaxError: invalid syntax
```

```
In [30]: # Code cell 13
Verify termostato()

termostato(21,False)
```

```
File "C:\Users\camil\AppData\Local\Temp\ipykernel_27344\1029548828.py", line 2
    Verify termostato()
    ^
SyntaxError: invalid syntax
```

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)]
```

```
In [31]: # Code cell 14

floor_types = ['Parking', 'Shops', 'Food Court', 'Offices']
floor_numbers = range(-1,3)
zipped =zip(floor_types,floor_numbers)
print(tuple(zipped))
```

```
(( '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"
```

```
In [32]: # Code cell 15
floor_types = ['Parking', 'Shops', 'Food Court', 'Offices']
floor_numbers = range(-1,3)
elevator_dict = dict(zip(floor_types, floor_numbers))
print(elevator_dict)

{'Parking': -1, 'Shops': 0, 'Food Court': 1, 'Offices': 2}
```

```
In [33]: # Code cell 16
# Verify elevator_dict[-1]
elevator_dict[-1]
```

```
-----
KeyError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_27344\3467140828.py in <module>
      1 # Code cell 16
      2 # Verify elevator_dict[-1]
----> 3 elevator_dict[-1]

KeyError: -1
```

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.
```

```
In [36]: # Code cell 17
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))

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

def go_to_floor(self, floor_number):
    if floor_number in self._floor_types:
        print("Going to {}".format(floor_numbers))
    else:
        print("There is no floor number {} in this building".format(floor_number))

```

In [34]: *# Verify code cell 18*
el = Elevator(floor_numbers, floor_types)
el.go_to_floor("Offices")

```

-----
NameError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_27344\1903391285.py in <module>
      1 # Verify code cell 18
----> 2 el = Elevator(floor_numbers, floor_types)
      3 el.go_to_floor("Offices")

NameError: name 'Elevator' is not defined

```

In [38]: *# Verify code cell 19*
el.go_to_floor(10)

```

-----
NameError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_27344\3261191770.py in <module>
      1 # Verify code cell 19
----> 2 el.go_to_floor(10)

NameError: name 'el' is not defined

```

In [37]: *# Verify code cell 20*
el.ask_which_floor('Offices')

```

-----
NameError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_27344\1113725302.py in <module>
      1 # Verify code cell 20
----> 2 el.ask_which_floor('Offices')

NameError: name 'el' is not defined

```

In [39]: *# Verify code cell 21*
el.ask_which_floor('Swimming Pool')

```
-----  
NameError                                Traceback (most recent call last)  
~\AppData\Local\Temp\ipykernel_27344\3318732474.py in <module>  
      1 # Verify code cell 21  
----> 2 el.ask_which_floor('Swimming Pool')  
  
NameError: name 'el' is not defined
```

¡Excelente trabajo!

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