prog_datasci_3_apython_ejerciciosResueltos

October 4, 2020

1 Programación para *Data Science*

1.1 Unidad 3: Estructuras de control y funciones en Python

1.2 Ejercicios para practicar

Los siguientes 5 ejercicios no puntúan para la PEC, pero os recomendamos que los intentéis resolver antes de pasar a los ejercicios propios de la PEC. También encontraréis las soluciones a estos ejercicios al final del Notebook.

1.2.1 Ejercicio 1

El ejercicio siguiente consiste en pasar un número en base 16 (hexadecimal, 0-9 / A-F) a base 10 (decimal). Para ello, debéis crear una **función** que dado un *string* que representa un número en hexadecimal, por ejemplo, `AE3F`, devuelva el número natural correspondiente. En este caso el resultado sería `44607`. NM

[1]:
```
# Respuesta
```

1.2.2 Ejercicio 2

Escribir una función que dado un número entero positivo, `N`, genere un archivo con el nombre`output.txt` que contendrá `N` líneas, donde cada línea deberá mostrar un número consecutivo de letras`A`. NM

Por ejemplo, si `N = 4`, el archivo generado deberá tener el siguiente contenido:

```
A
AA
AAA
AAAA
```

[2]:
```
# Respuesta
```

1.2.3 Ejercicio 3

Completad el código necesario para calcular el número de vocales y de consonantes respectivamente de un texto. NM

[3]:
```
def contar_vocales_y_consonantes(texto):
    # Cuenta las vocales contenidas en el string texto y también las␣
 ↪consonantes.
    num_vocales = 0
    num_consonantes = 0

    # Código que hay que completar.

    return (num_vocales, num_consonantes)
```

[4]:
```
# Respuesta
```

1.2.4 Ejercicio 4

Escribe una función con argumento en forma de lista $x = [a, b, c, \ldots]$ que **devuelva una función** que calcule el polinomio $f(x) = a + b \cdot x + c \cdot x^2 + \cdots$. Puedes ayudarte de la función `eval()` y la función `zip()`EI

[ ]:
```
# Respuesta
def creador(x):
    # Definición de la función

f = creador([1,2,3,4])
f(3)
```

1.2.5 Ejercicio 5

Un número primo es aquél que solo es divisible por él mismo y por 1.

Escribe un código que compruebe si un número `x = 15` es solo divisible por 1 o por el mismo. Escribe este código usando un iterador (un `for` o un `while`) que barra todos los valores desde `2` a `x-1`. Crea una variable `divisible` que tenga por defecto valor `False` y asigne el valor `True` si a lo largo de la iteración encuentra un número natural divisible. Puedes usar el operador modulo `a % b` para saber si un numero `b` es divisible por `a`. NM

[ ]:
```
# Respuesta:
```

1.3 Soluciones ejercicios para practicar

1.3.1 Ejercicio 1

El ejercicio siguiente consiste en pasar un número en base 16 (hexadecimal, 0-9 / A-F) a base 10 (decimal). Para ello, debe crear una **función** que dado un *string* que representa un número en hexadecimal, por ejemplo, `AE3F`, devuelva el número natural correspondiente. En este caso el resultado sería `44607`.

Respuesta

La formula para convertir un número hexadecimal a un número decimal, tomando como ejemplo el número AE3F, es: `A * 16**3 + E * 16**2 + 3 * 16**1 + F * 16**0 = 10 * 16**3 + 14 * 16**2 + 3 * 16**1 + 15 * 16**0`

[6]:
```
def hex_to_dec(numero_hexadecimal):

    hex_mapping = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5,
                   '6': 6, '7': 7, '8': 8, '9': 9, 'A': 10, 'B': 11,
                   'C': 12, 'D': 13, 'E': 14, 'F': 15}

    # Primero, comprobamos que el número que se pasa por parámetro es␣
↪hexadecimal
    all_hex = True
    for c in numero_hexadecimal:
        if not c in hex_mapping.keys():
            all_hex = False

    if all_hex:

        # Definimos la base para realizar las operaciones
        base = 16
        numero_decimal = 0

        # Invertimos el número hexadecimal para que nos sea más fácil trabajar␣
↪con los índices
        numero_hexadecimal = numero_hexadecimal[::-1]

        for i in range(len(numero_hexadecimal)):
            # Por cada carácter hexadecimal, aplicamos la formula c * base ** y,
            # donde c es la representación decimal del carácter y
            # sumamos el resultado al resultado obtenido en la iteracó anterior
            numero_decimal = numero_decimal +␣
↪hex_mapping[numero_hexadecimal[i]] * base**i

        return numero_decimal
    else:
        print("El número introducido no es correcto")

print(hex_to_dec('AE3F'))
print(hex_to_dec('FFF'))
print(hex_to_dec('123'))
```

```
44607
4095
291
```

1.3.2 Ejercicio 2

Escribid una función que dado un número entero positivo, `N`, genere un archivo con el nombre`output.txt` que contendrá `N` líneas, donde cada línea deberá mostrar un número consecutivo de letras`A`.

Por ejemplo, si `N = 4`, el archivo generado deberá tener el siguiente contenido:

```
A
AA
AAA
AAAA
```

Respuesta

[18]:
```
# Definimos una función que recibirá un número entero por parámetro
def generar_archivo(N):
    # Abrimos el archivo output.txt en modo escritura.
    # El parámetro 'w' hará que si el archivo existe, éste se sobrescribirá
    with open('output.txt', 'w') as fd:

        # Recorremos el rango [0 ... N-1]
        for i in range(N):
            # Generamos una cadena con un número ascendente de caracteres A
            linea = 'A' * (i + 1)
            # Escribimos cada cadena generada en el fichero, añadiendo un salto␣
 ↪de línea
            fd.write(linea + '\n')

generar_archivo(4)
```

1.3.3 Ejercicio 3

Completad el código necesario para calcular el número de vocales y de consonantes respectivamente de un texto.

Respuesta

[19]:
```
def contar_vocales_y_consonantes(text):
    # Cuenta las vocales contenidas en el string texto y también las␣
 ↪consonantes.
    num_vocales = 0
    num_consonantes = 0

    # Definimos una lista con las vocales
    vocales = ['a', 'e', 'i', 'o', 'u']

    for c in text.lower(): # Podemos convertir el texto en minúsculas para␣
 ↪simplificar los cálculos
        if c in vocales:
```

```
            num_vocales = num_vocales + 1
        elif c > 'a' and c <= 'z':
            num_consonantes = num_consonantes + 1

    return (num_vocales, num_consonantes)

texto = "Orbiting Earth in the spaceship, I saw how beautiful our planet is. \
            People, let us preserve and increase this beauty, not destroy it!"

num_vocales, num_consonantes = contar_vocales_y_consonantes(texto)
print( "El numero de vocales es %d." % num_vocales)
print( "El numero de consonantes es %d." % num_consonantes)
```

```
El numero de vocales és 44.
El numero de consonantes és 62.
```

1.3.4 Ejercicio 4

Escribe una función con argumento en forma de lista $lista = [a, b, c, \ldots]$ que **devuelva una función** que calcule el polinomio $f(x) = a + b \cdot x + c \cdot x^2 + \cdots$. Hay muchas formas de resolver este ejercicio. Puedes ayudarte de la función `eval()`, un `for` funciones lambda, map/reduce o la función `zip()`EI

[59]:
```
# Respuesta

# Solución en base a un for
def creador(lista):
    # Definición de la función
    def pol(x):
        toeval = ""
        for coef,expo in enumerate(lista):
            toeval += '+'+ str(expo)+ "*x**"+str(coef) + ' '
        print(toeval)
        return(eval(toeval))
    return(pol)

f = creador([1,2,3,4])
f(3)
```

```
+1*x**0 +2*x**1 +3*x**2 +4*x**3
```

[59]:
```
142
```

[58]:
```
# Respuesta
from functools import reduce #  debemos importar reduce

def creador(lista):
    # Esta solución usa map/reduce/zip en vez un for
    def concat(a,b): #función para concatenar  term1+term2
```

```
        return(a+' + '+b)
    def element(t): # función para construir un término del polinomio c*x^d
        return( str(t[0])+"*x**"+str(t[1]) )
    def pol(x):
        toeval= reduce(concat,map(element,
                          zip(lista,range(len(lista))))) # zip devuelve tuplas␣
↪con los elemetos de pares de cada lista
        print(toeval)
        return(eval(toeval))
    return(pol)

f = creador([1,2,3,4])
f(3)
```

```
1*x**0 + 2*x**1 + 3*x**2 + 4*x**3
```

[58]:
```
142
```

[62]:
```
# Respuesta
# Una versión más compacta con definiciones lambda
from functools import reduce #  debemos importar reduce

def creador(lista):
    # Definición de la función
    def pol(x):
        out = eval(reduce(lambda a,b: a+'+'+b,
                    map(lambda t: str(t[0])+"*x**"+str(t[1]),
                        zip(lista,range(len(lista))))))
        return(out)
    return(pol)

f = creador([1,2,3,4])
f(3)
```

[62]:
```
142
```

1.3.5 Ejercicio 5

Un número primo es aquél que solo es divisible por él mismo y por 1.

Escribe un código que compruebe si un número `x = 15` es solo divisible por 1 o por el mismo. Escribe este código usando un iterador (un `for` o un `while`) que barra todos los valores desde `2` a `x-1`. Crea una variable `divisible` que tenga por defecto valor `False` y asigne el valor `True` si a lo largo de la iteración encuentra un número natural divisible. Puedes usar el operador modulo `a % b` para saber si un numero `b` es divisible por `a`. NM

[ ]:
```
# Definimos primero x como el 15 y la etiqueta divisible

```

```python
x = 15

divisible = False
# Respuesta


# Creamos un iterador desde 2 a x-1 (range(2,x) donde x no estará incluido)
# Miramos, para cada valor del iterador, si es divisible con
#       la función modulo %, si es así, la función modulo
#       nos devolvera 0 y asignaremos divisible a True
#       En este caso el número no sera primo

for i in range(2,x):
    if not x%i:
        divisible = True

print(divisible)
```