Programación para Data Science

Unidad 3: Conceptos avanzados de Python

En este Notebook encontraréis dos conjuntos de ejercicios: un primer conjunto de ejercicios para practicar y que no puntuan, pero que recomendamos intentar resolver y un segundo conjunto que evaluaremos como PEC. Además, encontraréis un ejercicio opcional que aunque no puntua directamente para la PEC, sí que nos sirve para subir la nota al final de la asignatura.

Ejercicios para practicar

Los siguientes 3 ejercicios no puntuan para la PEC, pero os recomendamos que los intentéis resolver antes de pasar a los ejercicios propios de la PEC. También podéis encontrar las soluciones a estos ejercicios al final del Notebook.

Ejercicio 1

El siguiente ejercicio consiste en pasar un número en base 16 (hexadecimal, 0-9/A-F) a base 10 (decimal). Para hacerlo, 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`.

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

Ejercicio 2

Las excepciones son errores detectados en tiempo de ejecución. Pueden y deben ser manejadas por el programador para minimizar el riesgo de que un determinado programa falle de forma no controlada. Escribid, en lenguaje Python, cómo generar y capturar la siguiente excepción: **ZeroDivisionError**.

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

Ejercicio 3

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

In [2]:
```
def contar_vocales_y_consonantes(texto):
    # Cuenta las vocales contenidas en el string texto y también las cons
onantes.
    num_vocales = 0
    num_consonantes = 0

    # Código que hay que completar.

    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 número de vocales es %d." % num_vocales
print "El número de consonantes es %d." % num_consonantes
```

```
El número de vocales es 0.
El número de consonantes es 0.
```

Ejercicios y preguntas teóricas para la PEC

A continuación encontraréis los **ejercicios y preguntas teóricas que debéis completar en esta PEC** y que forman parte de la evaluación de esta unidad.

Pregunta 1

Las funciones **range** y **xrange** pueden utilizarse con la misma finalidad, pero su funcionamiento es diferente. Poned un ejemplo donde sería recomendable intercambiar la función **range** por la función **xrange**. **(0.5 puntos)**

Respuesta:

Escribid vuestra respuesta aquí

Pregunta 2

a) Explicad brevemente cada línea de código del siguiente bloque (añadid comentarios en el mismo bloque de código):

In [3]:
```
# Añadid vuestros comentarios de código en este mismo bloque

def create_generator():
    for i in range(10):
        yield i

num_generator = create_generator()
for i in num_generator:
    print "Primera iteración: número generado =", i

for j in num_generator:
    print "Segunda iteración: número generado =", j
```

```
Primera iteración: número generado = 0
Primera iteración: número generado = 1
Primera iteración: número generado = 2
Primera iteración: número generado = 3
Primera iteración: número generado = 4
Primera iteración: número generado = 5
Primera iteración: número generado = 6
Primera iteración: número generado = 7
Primera iteración: número generado = 8
Primera iteración: número generado = 9
```

b) Explicad brevemente la salida por pantalla que observamos al ejecutar el código anterior.

(1.5 puntos)

Respuesta

Escribid vuestra respuesta aquí

Ejercicio 1

Escribid una función que dada una lista de planetas del sistema solar, pregunte al usuario que introduzca una posición y muestre el plante correspondiente a dicha posición. Por ejemplo, si tenemos la siguiente lista: `['Mercurio', 'Venus', 'Tierra', 'Marte']` y el usuario nos ha introducido la posición `3`, hemos de mostrar como resultado por pantalla: `Tierra`.

Consideraciones:

- La posición que introduzca el usuario tiene que ser un número entero estrictamente positivo.
- La función debe controlar el acceso a una posición fuera de la lista mediante una **excepción**. Por ejemplo, en el caso anterior debemos mostrar una mensaje de error si el usuario pide acceder a la posición 10.

(1.5 puntos)

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

Ejercicio 2

Dada una lista de planetas del sistema solar, determinad cuales de estos planetas tienen una masa superior a la de la Tierra. Por ejemplo, si la lista inicial es `['Venus', 'Marte', 'Saturno']`, el resultado que mostraríamos por pantalla sería `['Saturno']` ya que el planeta Saturno tiene una masa `95.2` veces superior a la Tierra.

Consideraciones:

- Debéis tener en cuenta que el nombre de los planetas que nos pasan por parámetro puede estar en minúsculas, mayúsculas o una combinación de ambas.
- Podéis asumir que no habrá acentos en el nombre de los planetas.
- Debéis determinar aquellos planetas que tiene una massa estrictamente superior a la de la Tierra.
- No habrá planetas repetidos en la lista que nos pasan por parámetro.

(1.5 puntos)

In [1]:
```
# Masas medidas con respecto a la Tierra
# Es decir, un valor de 14.6 representaria una masa 14.6 veces superior a
 la de la Tierra
masas = {'Mercurio': 0.06, 'Venus': 0.82, 'Tierra': 1, 'Marte': 0.11, 'Ju
piter': 317.8,
          'Saturno': 95.2, 'Urano': 14.6, 'Neptuno': 17.2, 'Pluto': 0.002
2}

def planetas_mas_grandes_que_Tierra(planetas):
    """
    Planetas con una masa superior a la de la Tierra
    """
    planetas_masa_superior = []

    # Código a completar

    return planetas_masa_superior

# Ejemplos de uso de la función anterior
print planetas_mas_grandes_que_Tierra(['Venus', 'Mercurio', 'Marte'])
print planetas_mas_grandes_que_Tierra(['Jupiter', 'Saturno', 'Pluto'])
print planetas_mas_grandes_que_Tierra(['urano', 'tierra', 'neptuno', 'mar
te', 'Venus'])
print planetas_mas_grandes_que_Tierra(['Tierra', 'MeRcUrIo', 'PLUTO', 'SA
TURNO'])

# Podéis añadir más ejemplos si lo consideráis oportuno
```

```
[]
[]
[]
[]
```

Ejercicio 3

Completad las siguientes funciones y documentad el código si lo consideráis oportuno. Finalmente, escribid al menos un ejemplo de uso para cada función.

(1.5 puntos)

In [1]:
```
# Completad las funciones matemáticas siguientes
import math

"""Función que calcula la altura en un movimiento de caída libre

Suponemos que dejamos caer un objeto des de un edificio de altura descono
cida.
El parámetro duracion_caida nos indica el tiempo (en segundos) que tarda
 el objeto en llegar a
la tierra. La función debería calcular la altura del edificio des del cua
l se ha
lanzado el objeto.

Podéis encontrar más información sobre el movimiento de caída libre en el
 siguiente
enlace: https://www.fisicalab.com/apartado/caida-libre#contenidos.
"""

def calcular_altura_caida_libre(duracion_caida):
    # Completar
    return 0.


"""Función que calcula las coordenadas cartesianas de un punto representa
do en coordenadas polares

Dado un punto representado por sus coordenadas polares (radio y angulo),
 la función debería calcular
las correspondientes coordenadas cartesianas y devolver una tupla con su
 valor.

Podéis encontrar más información sobre el sistema de coordenadas polares
 y su conversión al sistema
cartesiano en el siguiente enlace: https://es.wikipedia.org/wiki/Coordena
das_polares.
"""

def calcular_coordenadas_cartesianas(radio, angulo_en_grados):
    # Completar
    return 0., 0.

# Escribid aquí al menos un ejemplo de uso utilizando las funciones anter
iores. Por ejemplo:
# print "El objeto se ha dejado caer des de una altura de %f metros" % ca
lcular_altura_caida_libre(10)
```

Ejercicio 4

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

Por ejemplo, si `N = 4`, el fichero generado deberá contener el siguiente contenido:

```
A
AA
AAA
AAAA
```

(1.5 puntos)

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

Ejercicio 5

Dada una cadena de caracteres, `s`, de longitud `n` y un número entero positivo `k`, siendo `k` un divisor de `n`, podemos dividir la cadena `s` en `n / k` sub-cadenas de la misma longitud.

Escribid una función que, dada una cadena `s` y un número entero `k`, devuelva las `n/k` sub-cadenas teniendo en cuenta las siguientes consideraciones:

- El orden de los caracteres en las sub-cadenas debe ser el mismo que en la cadena original.
- Todos los caracteres de las sub-cadenas deben aparecer una única vez. Es decir, si un caracter se repite dentro de una sub-cadena, sólo hemos de mostrar la primera ocurrencia.

Por ejemplo, si tenemmos
`s = AABCCAADA`
`k = 3`

el resultado a mostrar por pantalla sería:
`AB`
`CA`
`AD`

Tenemos que la longitud de la cadena es 9 y por lo tanto, podemos formar 3 sub-cadenas:

`AAB -> AB` (el caracter A se repite dos veces)

`CCA -> CA` (el caracter C se repite dos veces)

`ADA -> AD` (el caracter A se repite dos veces)

(2 puntos)

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

Ejercicio 6 (Opcional)

Al final de la Edad Media, en Francia, el diplomático francés Blaise de Vigenère desarrollo un algoritmo para cifrar mensajes que nadie fue capaz de romper durante aproximadamente 250 años. El algoritmo se conoce con el nombre de cifrado de Vigenère (https://es.wikipedia.org/wiki/Cifrado_de_Vigen%C3%A8re).

El cifrado de Vigenère consiste en añadir a cada una de las letras de un texto un desplazamiento a partir de una clave secreta para conseguir una nueva letra diferente de la original. Veamos un ejemplo:

Si asignamos el número 1 a la primera letra del abecedario, A, 2 a la siguiente, B, etc., imaginad que tenemos el siguiente mensaje:

```
ABC
123
```

y la siguiente clave secreta:

```
DEF
456
```

A cada letra del mensaje original aplicamos un desplazamiento en función de la misma posición dentro de la clave secreta. Por lo tanto, el mensaje cifrado quedaría de la siguiente forma:

```
   E       G       I
(1 + 4) (2 + 5) (3 + 6)
```

Escribid una función que, dado un mensaje y una clave secreta, calcule y devuelva el mensaje cifrado.

Consideraciones.

- Utilizad como alfabeto de entrada **el alfabeto inglés en mayúsculas**.
- El valor por defecto de la clave secreta será **DATASCI**.

In [3]:
```
def cifrado_vigenere(mensaje, clave="DATASCI"):
    """
    Cifra el mensaje utilizando el cifrado de Vigenère
    """
    mensaje_cifrado = ""

    # Código que hay que completar

    return mensaje_cifrado

# Aquí podéis añadir más ejemplos:
print cifrado_vigenere("ATACAREMOS AL AMANECER")
```

Soluciones ejercicios para practicar

Ejercicio 1

El siguiente ejercicio consiste en pasar un número en base 16 (hexadecimal, 0-9/A-F) a base 10 (decimal). Para hacerlo, 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`.

Respuesta

En Python disponemos de una función muy útil que nos permite pasar a un número decimal des de cualquier base (`int(x, base=y)`). Dado que el objetivo es jugar un poco con el lenguaje Python, vamos a usar dicha función sólo de forma parcial para calcular el número decimal correspondiente a cada carácter hexadecimal individualmente.

La fórmula 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
```

In [4]:
```
# Importamos el string '0123456789abcdefABCDEF' que nos puede ser muy útil para comprobar el formato
from string import hexdigits

def hex_to_dec(numero_hexadecimal):
    # Primero, comprobamos que el número que se pasa por parámetro es hexadecimal
    if all(c in hexdigits for c in numero_hexadecimal):
        # 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)):
            # Para cada carácter hexadecimal aplicamos la formula c * base ** i,
            # donde c es la representación decimal del carácter y
            # sumamos el resultado al resultado obtenido en la iteración anterior
            numero_decimal += int(numero_hexadecimal[i], 16) * base**i

        return numero_decimal
    else:
        print 'El número introducido no es hexadecimal'

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

```
44607
4095
291
```

Ejercicio 2

Las excepciones son errores detectados en tiempo de ejecución. Pueden y deben ser manejadas por el programador para minimizar el riesgo de que un determinado programa falle de forma no controlada. Escribid, en lenguaje Python, cómo generar y capturar la siguiente excepción: **ZeroDivisionError**.

Respuesta

En Pyhon podemos utilizar el bloque try ... except para capturar excepciones. Primero se intentará ejecutar el código dentro del bloque try y si hay una excepción se buscará una instrucción except que capture dicha excepción. En caso de encontrarla se ejecutará el código dentro del bloque except.

In [5]:
```
try:
    print 5/0 # División por cero - genera ZeroDivisionError
except ZeroDivisionError:
    print "¡Cuidado! División por cero."
```

```
¡Cuidado! División por cero.
```

Ejercicio 3

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

Respuesta

In [6]:
```
def contar_vocales_y_consonantes(texto):
    # Cuenta las vocales contenidas en el string texto y también las cons
onantes.
    num_vocales = 0
    num_consonantes = 0

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

    for c in texto.lower(): # Podemos convertir el texto a minúsculas par
a simplificar los cálculos
        if c in vocales:
            num_vocales += 1
        elif c > 'a' and c <= 'z':
            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 número de vocales es de %d" % num_vocales
print "El número de consonantes es de %d" % num_consonantes
```

```
El número de vocales es de 44
El número de consonantes es de 62
```

Si queremos considerar también las vocales acentuadas o caracteres especiales, podemos modificar el código anterior para tenerlo en cuenta:

In [7]:
```
def contar_vocales_y_consonantes(texto):
    # Cuenta las vocales contenidas en el string texto y también las cons
onantes.
    num_vocales = 0
    num_consonantes = 0

    # Convertimos el texto a Unicode
    # En este caso sabemos seguro que la codificación de caracteres es UT
F-8,
    # pero si nuestro código se puediera ejecutar fuera del Notebook podr
íamos
    # incluir la codificación como otro parámetro de la función
    texto = texto.decode('utf-8')

    # Definimos una lista con las vocales en unicode
    vocales = [u'a', u'e', u'i', u'o', u'u', u'à', u'á', u'è', u'é', u'í'
, u'ï', u'ò', u'ó', u'ú', u'ü']

    for c in texto.lower(): # Podemos convertir el texto a minúsculas par
a simplificar los cálculos
        if c in vocales:
            num_vocales += 1
        elif c > 'a' and c <= 'z' or c == u'ç' or c == u'ñ':
            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 número de vocales es de %d" % num_vocales
print "El número de consonantes es de %d" % num_consonantes

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

```
El número de vocales es de 44
El número de consonantes es de 62
El número de vocales es de 5
El número de consonantes es de 1
```