entrega_3

July 18, 2018

1 Programación para la Bioinformática

1.1 Unidad 3: Conceptos avanzados de Python

1.2 Ejercicios y preguntas teóricas

A continuación, encontraréis la parte que tenéis que completar en este modulo y las preguntas teóricas a contestar.

1.2.1 Ejercicio 1

Completa el código necesario para:

- Calcular el número de palabras
- Calcular el número de espacios de un texto
- Calcula el número de letras 'r' o 'R' que aparecen

Ten en cuenta lo siguiente:

- **El número de espacios no tiene que ser necesariamente función del número de palabras**
- **Entre una línea y la siguiente de texto, justo después del carácter " hay un gran número de espacios**
- **Hay que contar el número de letras 'i', indistintamente de si son mayúsculas o minúsculas**

```
In [1]: texto1 = "There was no reason to believe that a young black boy at this time, \
            in this place, could in any way alter history. After all, South Africa was \
            then less than a decade removed from full British control. Already, laws \
            were being codified to implement racial segregation and subjugation, \
            the network of laws that would be known as apartheid"

        texto2 = "Repugnant is a creature who would squander the ability to lift \
                    an eye to heaven, conscious of his fleeting time here."

        def contar_palabras(texto):
            # Cuenta las palabras contenidas en el string texto
            num_palabras = 0

            # Código a completar
```

```
        return num_palabras


    def contar_espacios(texto):
        # Cuenta los espacios contenidos en el string texto
        num_espacios = 0

        # Código a completar

        return num_espacios


    def contar_letras(texto, letra='r'):
        # Cuenta de letras 'letra' contenidos en texto
        num_letras = 0

        # Código a completar

        return num_letras


    # Utilizamos la funciones para el texto1:
    num_palabras = contar_palabras(texto1)
    num_espacios = contar_espacios(texto1)
    num_letras = contar_letras(texto1)
    print("El número de palabras es de %d" % num_palabras)
    print("El número de espacios es de %d" % num_espacios)
    print("El número de letras 'r' es de %d" % num_letras)

    print()

    # Y también para el texto2:
    num_palabras = contar_palabras(texto2)
    num_espacios = contar_espacios(texto2)
    num_letras = contar_letras(texto2)
    print("El número de palabras es de %d" % num_palabras)
    print("El número de espacios es de %d" % num_espacios)
    print("El número de letras 'r' es de %d" % num_letras)
```

```
El número de palabras es de 0
El número de espacios es de 0
El número de letras 'r' es de 0

El número de palabras es de 0
El número de espacios es de 0
El número de letras 'r' es de 0
```

Pista Echad un vistazo a la documentación de Python para la función *count()*: https://docs.python.org/3/library/string.html

1.2.2 Ejercicio 2

Completad las siguientes funciones matemáticas y documentad el código también de cada función. Por último, escribid algún ejemplo de uso de cada una de las funciones:

```
In [2]: # Completa las siguientes funciones matemáticas.
        # ¡Atención al uso de números decimales!
        import math

        # Un ejemplo:
        def area_rectangulo(base, altura):
            return base * altura


        def area_circulo(radio):
            # A completar
            return 0.


        def longitud_circulo(radio):
            # A completar
            return 0.


        def hipotenusa(cateto1, cateto2):
            # A completar
            return 0.


        def area_cuadrado(lado):
            # A completar
            return 0.


        def volumen_esfera(radio):
            # A completar
            return 0.


        def volumen_cubo(lado):
            # A completar
            return 0.


        print(area_rectangulo(2.0, 4.0))
```

```
        # Escribe aquí algunos ejemplos utilizando estas funciones, por ejemplo:
        # print 'El volumen del cubo de lado 3.5 es %f' % volumen_cubo(3.5)
```

```
8.0
```

1.2.3 Ejercicio 3

El siguiente ejercicio consiste en pasar un número en base 2 (binario, 0/1) a base 10 (decimal).

Dado un string que representa un número en binario, por ejemplo, 1011, devolver el número natural correspondiente, en este caso, 11.

```
In [3]: # A completar: DEFINID UNA FUNCIÓN y escribid algunos casos de
        # uso de esa función
```

1.2.4 Ejercicio 4

Dada una molécula representada por un string del estilo C9-H8-O4 calcula su masa atómica. Por ejemplo, para una molécula C4-H3, la masa atómica sería de 4*12.01 + 3*1.007825.

Haced una solución general accediendo al diccionario mediante la clave, que en este caso será el tipo de átomo. Por ejemplo, para la molécula C5-H3 deberíamos seguir estos pasos: * Separar la molecula por los guiones (podemos hacerlo con la función split, por ejemplo). * Para cada una de las partes, C5 y H3, encontrar el tipo de átomo: C y H. (Necesitaremos un bucle de algún tipo aquí) * Acceder al diccionario de masas y para la clave que se corresponde con el tipo de átomo, obtener la masa. * Una vez encontrada la masa, multiplicarla por el número de átomos encontrados.

Pista: para un string del tipo a = 'C15', a[0] nos proporcionará el tipo de átomo, C. a[1:] nos proporciona el string restante: '15'. **Tened en cuenta que ha de convertirse a número decimal para poder multiplicarse**.

```
In [4]: # Masas atómicas
        masas = {'H': 1.007825, 'C': 12.01, 'O': 15.9994,
                 'N': 14.0067, 'S': 31.972071, 'P': 30.973762}

        def calcula_masa_atomica(molecula):
            """
            Calcula la masa atomica de una molecula
            """
            masa = 0.0

            # Código a completar

            return masa

        print(calcula_masa_atomica('C13-H18-O2'))
        print(calcula_masa_atomica('C8-H10-N4-O2'))
        print(calcula_masa_atomica('C20-H25-N3-O'))
        print(calcula_masa_atomica('C20-H10-O2-P2-S'))
```

```
0.0
0.0
0.0
0.0
```

Nota importante Como se especifica en el enunciado, el diccionario ha de crearse sobre la marcha. **Los siguientes ejemplos son incorrectos**:

```
In [5]: masas = {'H': 1.007825, 'C': 12.01, 'O': 15.9994,
                 'N': 14.0067, 'S': 31.972071, 'P': 30.973762}

        # Este primer ejemplo tiene varios problemas:
        # 1: no tiene en cuenta el parámetro molecula
        # 2: no utiliza ningún bucle, la solución está "harcoded"
        # 3: la función no tiene return
        def calcula_masa_atomica(molecula):
            masa = masas['C']*13 + masas['H']*18 + masas['O']*2

        print(calcula_masa_atomica('C13-H18-O2'))
```

```
None
```

```
In [6]: masas = {'H': 1.007825, 'C': 12.01, 'O': 15.9994,
                 'N': 14.0067, 'S': 31.972071, 'P': 30.973762}

        # Este segundo ejemplo sí que utiliza un bucle,
        # pero no hace uso del diccionario de masas
        def calcula_masa_atomica(molecula):
            masa = 0.0
            grupos = molecula.split('-')
            for grupo in grupos:
                if grupo[0] == 'C':
                    masa += float(grupo[1:]) * 12.01
            return masa

        print(calcula_masa_atomica('C13-H18-O2'))
```

```
156.13
```

1.2.5 Ejercicio 5

Uno de los algoritmos más básicos en criptografía es el cifrado César (https://es.wikipedia.org/wiki/Cifrado_C%C3%A9sar), que fue utilizado por Julio César para comunicarse con sus generales, y que consiste en dado un texto, por cada una de las letras del texto, añadirle un desplazamiento para conseguir una nueva letra diferente de la original. Comprenderemos rápidamente su mecanismo mediante 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

Si aplicamos un desplazamiento de 3, buscaremos cuál es la letra en el abecedario que se corresponde: DEF 456

ABC se ha convertido en DEF porque hemos sumado un desplazamiento de 3. También podríamos aplicar otros tipos de desplazamiento como los negativos. Por ejemplo, para el desplazamiento -1 y el mensaje original ABC tendríamos un mensaje cifrado de: ZAB.

Escribid una función que dado un mensaje original y un desplazamiento, calcule y devuelva el mensaje cifrado:

```
In [7]: def cifrado_cesar(mensaje, desplazamiento=1):
            """
            Cifra el mensaje utilizando el metodo de Cesar
            dado un desplazamiento
            """
            mensaje_cifrado = ""

            # Codigo a completar

            return mensaje_cifrado

        # Aqui podeis añadir mas ejemplos
        print(cifrado_cesar("PROGRAMACION PARA LA BIOINFORMATICA", 1))
```

1.2.6 Ejercicio 6

El formato PDB se utiliza en química computacional para guardar información sobre moléculas en disco. Cada línea que empieza por ATOM representa un átomo de la molécula: ATOM 1 N ARG A 1 0.609 18.920 11.647 1.00 18.79 N ATOM 2 CA ARG A 1 0.149 17.722 10.984 1.00 13.68 C

0.609, 18.920 y 11.647 son las coordenadas x, y, z del átomo 1 (un nitrógeno, 'N', última letra de la línea). El segundo átomo es de tipo C (Carbono) y nombre CA (Carbono Alfa).

Completad el siguiente código que nos cuente el número de átomos encontrado para cada elemento.

Por ejemplo, para las siguientes líneas: ATOM 211 N TYR A 27 4.697 8.290 -3.031 1.00 13.35 N ATOM 212 CA TYR A 27 5.025 8.033 -1.616 0.51 11.29 C ATOM 214 C TYR A 27 4.189 8.932 -0.730 1.00 10.87 C ATOM 215 O TYR A 27 3.774 10.030 -1.101 1.00 12.90 O

Tendríamos dos átomos de tipo C, un átomo de tipo O y otro átomo de tipo N.

IMPORTANTE: Tened en cuenta que a priori no sabéis que tipos de átomo podéis encontrar hasta que no se lee la línea en vuestro código, por lo que no podéis definir una lista o diccionario de átomos encontrados salvo que se defina vacía y vaya actualizándose conforme encontramos tipos de átomos que no habíamos encontrado antes.

El archivo que utilizaremos está en el directorio data/ y el fichero es 1l2y.pdb.

```
In [8]: import os
```

```
        """
        El siguiente código es una pista de cómo leer línea por
        línea los átomos y acceder a su tipo (el último carácter):

        lines = molecula.split(os.linesep)
        for line in lines:
            if line.startswith('ATOM'):
                atomo = line[77].strip()
                print atomo
        """

        def cuenta_atomos(pdb_file_name):

            # Guardaremos en esta variable los átomos encontrados:
            numero_atomos = dict()

            # Abrimos el fichero:
            with open(pdb_file_name) as file_content:

                # Código a completar, borrad 'pass'
                pass

            return numero_atomos


        resultado = cuenta_atomos('data/1l2y.pdb')

        # El resultado debería ser un diccionario con los
        # siguientes valores:
        # resultado = {'N':27, 'C':98, 'O':29, 'H':150}
        print(resultado)
```

```
{}
```

Pista El siguiente código os ayudará a entender cómo crear un diccionario sobre la marcha y programar la solución para este ejercicio:

```
In [9]: import os

        molecula = """
        ATOM    211  N   TYR A  27       4.697   8.290  -3.031  1.00 13.35           N
        ATOM    212  CA  TYR A  27       5.025   8.033  -1.616  0.51 11.29           C
        ATOM    214  C   TYR A  27       4.189   8.932  -0.730  1.00 10.87           C
        ATOM    215  O   TYR A  27       3.774  10.030  -1.101  1.00 12.90           O
        ATOM    216  CB  TYR A  27       6.509   8.214  -1.310  0.51 12.65           C
        ATOM    218  CG  TYR A  27       7.406   7.086  -1.795  0.51 14.00           C
        ATOM    220  CD1 TYR A  27       7.951   6.144  -0.978  0.51 14.16           C
```

```
ATOM    222  CD2 TYR A  27       7.674   6.963  -3.164  0.51 17.10           C
ATOM    224  CE1 TYR A  27       8.752   5.109  -1.405  0.51 14.93           C
ATOM    226  CE2 TYR A  27       8.455   5.964  -3.656  0.51 17.43           C
ATOM    228  CZ  TYR A  27       8.990   5.041  -2.763  0.51 16.44           C
ATOM    230  OH  TYR A  27       9.803   4.026  -3.237  0.51 17.79           O
ATOM    232  N   CYS A  28       3.977   8.402   0.487  1.00 10.53           N
ATOM    233  CA  CYS A  28       3.295   9.146   1.517  1.00 10.04           C
ATOM    234  C   CYS A  28       4.174  10.264   2.053  1.00 10.31           C
ATOM    235  O   CYS A  28       5.378  10.108   2.202  1.00 12.91           O
ATOM    236  CB  CYS A  28       2.912   8.210   2.680  1.00  9.81           C
ATOM    237  SG  CYS A  28       1.804   6.853   2.197  1.00  9.93           S
ATOM    238  N   GLY A  29       3.546  11.391   2.430  1.00 10.76           N
ATOM    239  CA  GLY A  29       4.295  12.472   3.049  1.00 11.93           C
ATOM    240  C   GLY A  29       3.416  13.524   3.596  1.00 12.60           C
ATOM    241  O   GLY A  29       3.985  14.574   4.052  1.00 15.82           O
ATOM    242  OXT GLY A  29       2.168  13.386   3.672  1.00 11.37           O
"""

# Vamos a leer línea por línea cada átomo y guardaremos sus
# coordenadas en un diccionario:
coordenadas = {}

# Separamos el string molecula por los saltos de línea
lines = molecula.split(os.linesep)
# Ahora iteramos por cada una de las línas
for line in lines:
    # Si la línea empieza por ATOM:
    if line.startswith('ATOM'):
        # Vamos a quedarnos con el número de átomo (211, 212, 214, etc.):
        campos = line.split()
        # El número de átomo se corresponde con el segundo campo
        # cuando separamos por espacios la línea. Como los índices
        # empiezan en 0, el segundo se corresponde con el índice 1:
        num_atom = campos[1]
        print(num_atom)

        # Ahora obtenemos las coordenadas x, y, z del átomo de
        # la misma forma:
        x = float(campos[6])
        y = float(campos[7])
        z = float(campos[8])

        # Por último, guardamos en el diccionario para la clave
        # del número de átomo las coordenadas:
        coordenadas[num_atom] = [x, y, z]

# Muestra el diccionario:
print(coordenadas)
```

```
211
212
214
215
216
218
220
222
224
226
228
230
232
233
234
235
236
237
238
239
240
241
242
{'211': [4.697, 8.29, -3.031], '212': [5.025, 8.033, -1.616], '214': [4.189, 8.932, -0.73], '2
```

1.2.7 Ejercicio 7

Para la siguiente lista de números, escribe por pantalla todos aquellos que sean impares:

```
numeros = [386, 462, 47, 418, 907, 344, 236, 375, 823, 566,
           597, 978, 328, 615, 953, 345, 399, 162, 758, 219,
           918, 237, 412, 566, 826, 248, 866, 950, 626, 949,
           687, 217, 815, 67, 104, 58, 512, 24, 892, 894, 767,
           553, 81, 379, 843, 831, 445, 742, 717, 958, 743,
           527, 345, 221, 200, 456]
```

```
In [10]: # Tu respuesta
```

1.2.8 Ejercicio 8

Escribe un programa que calcule la distancia euclidiana entre dos puntos en el espacio 2D. Estos puntos son (x1, y1) y (x2, y2):

```
In [11]: def distance(x1, y1, x2, y2):
             d = 0.0

             # A completar
```

```
            return d


        print("La distancia entre (1,2) y (2,4) es: ", distance(1.0, 2.0, 2.0, 4.0))
        print("La distancia entre (3,2) y (7,-3) es: ", distance(3.0, 2.0, 7.0, -3.0))
        print("La distancia entre (5,1) y (-2,5) es: ", distance(5.0, 1.0, -2.0, 5.0))
```

```
La distancia entre (1,2) y (2,4) es:  0.0
La distancia entre (3,2) y (7,-3) es:  0.0
La distancia entre (5,1) y (-2,5) es:  0.0
```

1.2.9 Pregunta 1

El paradigma de programación orientada a objetos es ampliamente utilizado en gran parte de las librerías que se escriben en Python. Es una forma útil de encapsular información de la que se ocupará el propio objeto donde se ha definido esa información.

Explicad qué son los siguientes conceptos:

- Una clase
- Un objeto
- Un atributo
- Un método
- Un constructor
- Una superclase y una subclase

Poned un ejemplo de definición de una clase en código Python y un ejemplo de uso de esa misma clase.

Puedes basarte en este material: http://life.bsc.es/pid/brian/python/#/7

Respuesta:

Tu respuesta

1.2.10 Pregunta 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.

Poned ejemplos de tipos de excepción en el lenguaje Python y de cómo se capturan.

Respuesta:

Tu respuesta