For, Listas y Tuplas

Programación
http://progra.usm.cl

UTFSM

Departamento de Informática
Universidad Técnica Federico Santa María

Desviación estándar

Datos: $[5{,}8 \quad 3{,}1 \quad 2{,}5]$

Promedio:

$$\bar{x} = \frac{1}{3}(5{,}8 + 3{,}1 + 2{,}5) = 3{,}8$$

Desviación estándar:

$$\sigma_x = \sqrt{\frac{1}{2}\Big((5{,}8 - 3{,}8)^2 + (3{,}1 - 3{,}8)^2 + (2{,}5 - 3{,}8)^2\Big)}$$

$$\approx 1{,}76$$

Ejercicio

Escriba un programa que pregunte
cuántos datos se ingresará, pida los datos,
y entregue la desviación estándar:

```
Cuantos datos? 5
Dato 1: 2.4
Dato 2: 7.1
Dato 3: 3.8
Dato 4: 3.3
Dato 5: 4.9
La desviacion estandar es 1.80693109996
```

Solución

```
n = int(raw_input('Cuantos datos? '))

datos = []
cont= 0
while cont < n:
    x = float(raw_input('Dato ' + str(i) + ': '))
    datos.append(x)
    cont =  cont + 1

suma = 0.0
i = 0
while i < n:
    suma = suma + datos[i]
    i = i + 1
promedio = suma / n

suma_dif_cuad = 0.0
i = 0
while i < n:
    suma_dif_cuad = suma_dif_cuad + (datos[i] - promedio) ** 2
    i = i + 1
desviacion = (suma_dif_cuad / (n - 1)) ** 0.5

print 'La desviacion estandar es', desviacion
```

Solución con funciones

```
def leer_datos(cantidad):
    datos = []
    cont = 1
    while cont < cantidad + 1:
        x = float(raw_input('Dato ' + str(cont) + ': '))
        datos.append(x)
        cont += 1
    return datos

def promedio(valores):
    suma = 0.0
    i = 0
    while i < len(valores):
        suma += valores[i]
        i += 1
    return suma / len(valores)

def desviacion(valores):
    p = promedio(valores)
    n = len(valores)
    suma = 0.0
    i = 0
    while i < n:
        suma += (valores[i] - p) ** 2
        i += 1
    return (suma / (n - 1)) ** 0.5

n = int(raw_input('Cuantos datos? '))
datos = leer_datos(n)
print 'La desviacion estandar es', desviacion(datos)
```

Agregar cosas a una lista

```
>>> valores = []
>>> valores.append(5)
>>> valores.append(1)
>>> valores.append(6)
>>> valores
[5, 1, 6]
>>> valores.append(-4)
>>> valores
[5, 1, 6, -4]
```

Crear lista con range

```
>>> valores = range(4)
>>> valores
[0, 1, 2, 3]
>>> range(2, 8)
[2, 3, 4, 5, 6, 7]
>>> range(2, 8, 2)
[2, 4, 6]
>>> range(5, 1)
[]
>>> range(5, 1, -1)
[5, 4, 3, 2]
```

Índices

```
>>> colores = ['azul', 'rojo', 'verde',
...            'amarillo']
>>> colores[2]
'verde'
>>> colores[1:3]
['rojo', 'verde']
>>> colores[-1]
'amarillo'
>>> colores.index('rojo')
1
>>> len(colores)
4
```

Modificar elementos de una lista

```
>>> colores = ['azul', 'rojo', 'verde',
...            'amarillo']
>>> colores[0] = 'fucsia'
>>> del colores[2]
>>> colores.insert(1, 'gris')
>>> colores
['fucsia', 'gris', 'rojo', 'amarillo']
```

Operaciones sobre listas

```
>>> x = [6, 1, 4, 7, 8]
>>> len(x)
5
>>> sum(x)
26
>>> x.sort()
>>> x
[1, 4, 6, 7, 8]
>>> x.reverse()
>>> x
[8, 7, 6, 4, 1]
>>> 7 in x
True
```

Iterar sobre listas

```python
ramos = ['Progra', 'Mate', 'Fisica']
i = 0
while i < len(ramos):
    print 'Tengo clases de', ramos[i]
    i += 1
```

```python
ramos = ['Progra', 'Mate', 'Fisica']
horas = ['8:00', '10:00', '12:00']
i = 0
while i < len(ramos):
    print 'Tengo', ramos[i], 'a las', horas[i]
    i += 1
```

Iterar con For sobre listas

```python
ramos = ['Progra', 'Mate', 'Fisica']
for ramo in ramos:
    print 'Tengo clases de', ramo
```

```python
ramos = ['Progra', 'Mate', 'Fisica']
horas = ['8:00', '10:00', '12:00']
for i in range(len(ramos)):
    print 'Tengo', ramos[i], 'a las', horas[i]
```

Copiado de listas (¡cuidado!)

```
>>> a = [5, 1, 4]
>>> b = a
>>> b[0] = 1000
>>> a
[1000, 1, 4]
```

```
>>> a = [5, 1, 4]
>>> b = list(a)
>>> b[0] = 1000
>>> a
[5, 1, 4]
```

Tuplas

```
posicion_alfil = (7, 6)

alumno = ('Fulano', 'De Tal', '201199001-1')

carta = (5, 'corazones')

fecha = (2011, 4, 12)

triangulo = ((5, 1), (2, 4), (-2, 0))

personaje = ('Arturo Prat',
             (1848, 4, 3),
             (1879, 5, 21))
```

Desempaquetado de tuplas

```
>>> punto = (6.6, -2.4, 3.7)
>>> x, y, z = punto
>>> print x
6.6

>>> personaje = ("Bernardo O'Higgins",
...              (1778, 8, 20),
...              (1842, 10, 24))
>>> nombre, nacimiento, defuncion = personaje
>>> an, mn, dn = nacimiento
>>> ad, md, dd = defuncion
>>> ad - an
64
```

Comparación de tuplas

```
>>> (4, 5, 7) < (7, 0, 0)
True
>>> (4, 5, 7) < (4, 2, 1)
False
>>> (4, 5, 7) < (4, 5, 7, -10)
True
```

```
>>> (2010, 5, 12) >= (2009, 12, 25)
True
```

Listas de tuplas

```
ramos = [
    ('Progra',     'IWI-131', 3),
    ('Mate',       'MAT-021', 5),
    ('Fisica',     'FIS-100', 3),
    ('Ed.Fisica', 'DEW-100', 1),
    ('Inmanente', 'HRW-102', 2),
]

for nombre, sigla, c in ramos:
    print nombre, 'tiene', c, 'creditos'
```

Ejercicio: distancias

Escriba una función `distancia(p1, p2)` que retorne la distancia entre los puntos `p1` y `p2`:

```
>>> a = (5.1, 7.3, 2.4)
>>> b = (1.1, 2.8, 4.9)
>>> distancia(a, b)
6.519202405202649
```

Solución

```
def distancia(p1, p2):
    x1, y1, z1 = p1
    x2, y2, z2 = p2
    dx = x2 - x1
    dy = y2 - y1
    dz = z2 - z1
    return (dx ** 2 + dy ** 2 + dz ** 2) ** .5
```

Ejercicio

Un polígono está determinado por la lista de sus vértices. Escriba una función `perimetro(vertices)` que entregue el perímetro del polígono definido por la lista `vertices`:

```
>>> p = [(4, 1), (7, 2), (7, 4), (5, 9)]
>>> perimetro(p)
18.609700215601432
```

Solución

```
def distancia(p1, p2):
    (x1, y1), (x2, y2) = p1, p2
    dx, dy = x2 - x1, y2 - y1
    return (dx ** 2 + dy ** 2) ** .5

def perimetro(vertices):
    n = len(vertices)
    suma = 0.0
    for i in range(n):
        a = vertices[i]
        b = vertices[(i + 1) % n]
        suma += distancia(a, b)
    return suma
```

Solución 2

```
def distancia(p1, p2):
    (x1, y1), (x2, y2) = p1, p2
    dx, dy = x2 - x1, y2 - y1
    return (dx ** 2 + dy ** 2) ** .5

def perimetro(vertices):
    n = len(vertices)
    actual = vertices[0:n]
    sgte = vertices[1:n] + [vertices[0]]
    distancias = []
    for i in range(n):
        d = distancia(actual[i], sgte[i])
        distancias.append(d)
    return sum(distancias)
```