Introducció a la Programació
Grau en Estadística

Sessió 10: Errors, *, **, paràmetres, for en llistes

Vicenç Soler

Índex

1. Introducció

Aquest document intenta ser un compendi d'alguns temes que no s'han vist durant el curs però que són importants tenir-los en compte.

2. Control d'errors

A la Sessió 6 (Variables String, Fitxers i Tractament d'errors) ja es va veure l'estructura try..except per a controlar errors, però només es va mostrar des del punt de vista de fitxers.

Aquest tipus de control es pot aplicar a qualsevol altre error que es pugui produir: conversió de String a enter, divisió per 0, etc. A continuació es mostra com llegir un valor i convertir-lo a enter sense que el programa falli:

```
try:
    n=int(input("Entra un num:"))
    print(f"Correcte: has entrat el valor {n}")
except:
    print("ERROR: No s'ha entrat un enter")
```

Exercici proposat 1: Fer que el programa anterior demani un num. constantment i que no pari fins que entrin un valor enter.

Afegir que es pot col·locar al costat de "except" el tipus d'error que es vol controlar, entre els quals hi ha: ValueError (el tipus d'error que hem volgut controlar a l'exemple anterior) i ZeroDivisionError.

3. Ús de * i **

3.1. Ús de *

Hem de diferenciar el seu ús en 2 parts: a dins del codi i en la capçalera de funcions.

Enmig del codi

Enmig del codi serveix per extreure els elements d'una llista. En l'exemple següent veiem 2 maneres d'afegir elements a una llista:

Fent servir l'operador '+' que concatena 2 llistes:

```
v = [10,20,30]
w = v + [40]
```

Extraient els elements d'una llista amb *:

```
v = [10,20,30]
w = [ *v, 40]
```

Evidentment, també serviria **v.append(40),** ja que l'afegim al final.

Recordem que en el cas de la dreta, no serviria w = [v , 40], ja que aquesta operació crearia una llista 'w' de 2 elements: el primer seria la llista 'v' i el segon seria el valor 40.

Altres expressions que es poden fer servir:

```
primer,segon,*resta_de_la_llista=w   #equivalent a v[0],v[1],v[2:]
primer, *mig, ultim=w    #equivalent a v[0],v[1:-1],v[-1]
w=[ w[-1] , *w[:-1] ]  #rotar-la:últim+tots els valors menys el darrer
w=[ *w[1:] , w[0] ]    #rotar-la:tots els valors menys el primer+primer
```

En capçalera de funcions

En capçalera de funció serveix per a indicar que es poden rebre un nombre indeterminat de paràmetres. Un exemple ja existent és la funció 'print', que pot rebre un nombre indeterminat de paràmetres (separats per coma). Exemple:

```
def f1(*v): #f1 té un num. il.limitat de paràmetres
    return sum(v)

s=f1(1,2,3,4,5,6,7,8,9)
print(s)
```

En aquest exemple, la funció 'f1' pot rebre el nombre de paràmetres que es desitgi. Dins la funció 'f1', la variable 'v' es tracta com una llista (per si volem accedir a elements concrets, comptar quants paràmetres rebem, etc.)

Si volem combinar-ho amb altres paràmetres:

```
def f1(*v,multiplicador=1): #f1 té un num. il.limitat de paràmetres
    return sum(v)*multiplicador

s=f1(1,2,3,4,5,6,7,8,9,multiplicador=10)
print(s)
```

En la crida hem d'especificar quin dels valors és 'multiplicador', i mantener el mateix ordre (és a dir, 'multiplicador' ha de posar-se el darrer).

3.2. Ús de **

Com abans, hem de diferenciar el seu ús en 2 parts: a dins del codi i en la capçalera de funcions. En aquest cas, ** també serveix per a extreure els valors però, a diferència de només *, els valors també tenen associat una clau. Per tant, l'ús de ** l'associarem més a **diccionaris**.

Enmig del codi

Enmig del codi serveix per extreure els elements d'un diccionari. En l'exemple següent veiem 2 maneres d'afegir elements a un diccionari:

Fent servir l'índex directament:

```
d = {"primer":1,"segon":2}
d["tercer"] = 3
print(d)
```

Extraient els elements d'una llista amb **:

```
d = {"primer":1,"segon":2}
d={**d,"tercer":3}
print(d)
```

En capçalera de funcions

En capçalera de funció també serveix per a indicar que es poden rebre un nombre indeterminat de paràmetres, però aquests han de portar un nom de paràmetre associat, ja que el ** empaquetarà aquest seguit de paràmetres en un diccionari, que es treballarà internament la funció. Exemple:

```
def f2(**v):    #converteix a diccionari el v que m'envien
    return v    #retornarem {'primer': 1, 'segon': 2}

a = 1
b = 2
d = f2(primer=a, segon=b)  #enviem 2 paràmetres
print(d)
```

3.3. Conclusió

El símbol *** es fa servir a llistes** i **** a diccionaris**, ja que el primer extreu valors de llistes i el segon no només els valors, sinó també l'índex associat (en llistes no és necessari ja que sabem que els índexs van de 0 a n).

Respecte a crides a funcions, * empaqueta paràmetres en una sola variable, que és tractada com a llista a dins de la funció. ** també empaqueta paràmetres, però amb nom associat per a poder convertir-los en diccionari, en aquest cas.

4. Paràmetres a funcions

En anteriors temes, ja s'havia parlat de paràmetres a funcions. Aquí els comentarem des del punt de vista posicional.

Hi ha 2 tipus de paràmetres:

1. Posicionals (sense posar noms): Estem obligats a posar-los en la posició que ens obliga la funció.
2. Per nom clau (quan posem noms): Podem posar-los en la posició que vulguem, sempre que especifiquem el nom de paràmetre.

Així, un exemple de crida posicional i no posicional seria:

```
def f4(sumand1, sumand2, multiplicador):
    return (sumand1 + sumand2) * multiplicador

#crida mantenint la posició
res = f4(1,2,3)     #crida de manera posicional
print(res)

#crida sense mantenir la posición, però especificant nom
res = f4(multiplicador=3, sumand2=100, sumand1=3)
print(res)
```

En aquest sentit, la capçalera genèrica d'una funció seria:

```
def funcio(posic1,posic2,/,posicionals o nom_clau,*,nom_clau)
```

On els caràcters '/' I '*' separem diversos tipus (zones) de paràmetres:

- **Posic1 i posic2** són 2 paràmetres de tipus posicionals. En aquest sentit, quan es crida a la funció, els 2 primers paràmetres de la crida aniran a les variables 'posic1' i 'posic2'.
- El **separador '/'** ens indica que els paràmetres anteriors són de tipus posicionals, i els de Després no.
- **Posicionals o nom_clau** són els paràmetres normals que estem acostumats a fer servir, els de l'exemple anterior: es poden cridar de manera posicional o bé fent servir el nom,
- El **separador '*'** ens indica que els paràmetres posteriors estan obligats a ésser cridats amb nom de paràmetre (el no clau o keyword).
- **nom_clau**: aquesta zona és dels paràmetres que porten nom associat.

5. For en llistes

Estem acostumats a fer anar llistes amb un 'for' o amb un 'while',quan volem o bé crear els elements d'una llista o treballar amb els seus elements.

Però Python, quan es crea una llista, permet de fer una sintaxi més reduïda. Exemples:

1. Crea una llista amb els enters de 0 a 9

```
v = [x for x in range(10)]  #x és el valor que es posarà a la llista
```

2. Crea una llista amb els enters de 1 a 10

```
v = [x+1 for x in range(10)] #x+1 són els valors de la llista
```

3. Crea una llista de parells

```
v = [x*2 for x in range(10)]  #x*2 són els valors
```

4. Crea una llista de quadrats de valors >3. Fixem-nos en el 'if' darrere del 'for':

```
v = [x**2 for x in range(10) if x>3 ]
```

5. Crea una llista amb quadrats de valors >3 i no del 6. Aquí posem 2 condicions, i es fa posant 2 'if' després del 'for':

```
v = [x**2 for x in range(10) if x>3 if x!=6 ]  # Resultat:[16,25,49,64,81]
```

6. Crea una llista amb quadrats i els que siguin <50 els substitueix per 0. En aquest cas posem un 'if' al davant del 'for' quan volem substituir alguns valors per uns altres, és a dir, que no sempre es posi el quadrat de la 'x' seleccionada al 'for'. En aquest cas, estem obligats posar 'else' ja que hem de dir el valor que substitueix.

```
v = [x**2 if x**2<50 else 0 for x in range(10) if x>3 if x!=6 ]
# El resulatt seria: [16, 25, 49, 0, 0]
```

Per tant, la sintaxi completa seria:

```
llista = [valor_a_posar_a_la_llista
          if condició else valor_substitut (*)
          for x in conjunt_de_dades
              if condició_1 if condició_... if condició_n (**)
         ]
```

(*) Aquest 'if' davant de 'for' serveix per a decidir quin valor posem, ja que posarem un valor.
(**) Aquest 'if' darrere de 'for' serveix per a decidir si posem valor a la llista.

Exercici proposat 2: Posar les expressions anteriors en el format de 'for' habitual, no reduït.

Matrius

En el següent exemple recordem com es crea una matriu. En aquest cas, de 2x3 plena de 0's:

```
matriu=[[0 for col in range(4)] for fila in range(3)]  #matriu 3x3 de 0's
```

6. Solucions als exercicis proposats

Solució Exercici proposat 1:

```
while True:
    try:
        n=int(input("Entra un num:"))
        break
    except:
        print("No has entrat un enter. Torna-ho a intentar.")
```

Solució Exercicis proposats del 2:

```
v = [x for x in range(10)]  #crea una llista amb els enters de 0 a 9
v=[]
for x in range(10):
    v+=[x]       #Equival a v.append(x)


v = [x+1 for x in range(10)]  #crea una llista amb els enters de 1 a 10
v=[]
for x in range(10):
    v+=[x+1]


v = [x*2 for x in range(10)]  #crea una llista amb parells
v=[]
for x in range(10):
    v+=[x*2]


v = [x**2 for x in range(10) if x>3 ]  #crea una llista amb quadrats de
valors >3
v=[]
for x in range(10):
    if x>3:
        v+=[x**2]


v = [x**2 for x in range(10) if x>3 if x!=6 ]  #crea una llista amb quadrats
de valors >3 i no del 6: [16, 25, 49, 64, 81]
v=[]
for x in range(10):
    if x>3 and x!=6:
        v+=[x**2]


v = [x**2 if x**2<50 else 0 for x in range(10) if x>3 if x!=6 ] #Els quadrats
que siguin <50 els substitueix per 0. Obligat posar 'else'
v=[]
for x in range(10):
    if x > 3 and x != 6:
        if x**2<50:
            v += [x ** 2]
        else:
            v+=[0]
```