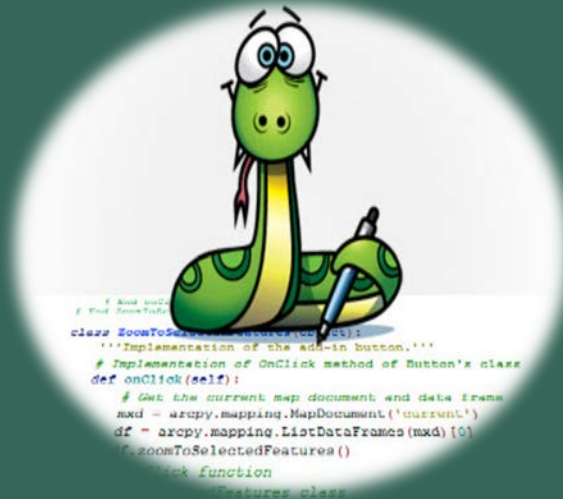


FUNCIONS: ÀMBIT DE LES VARIABLES

VARIABLES LOCALS, GLOBALS I NONLOCAL



M^a Belén Tortosa Pedrón



ÀMBIT DE LES VARIABLES

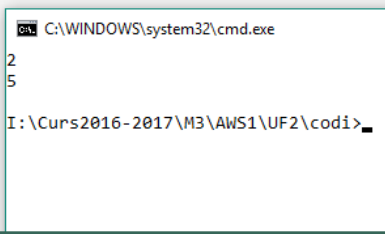
- Python distingeix tres tipus de variables: les variables locals i dos tipus de variables lliures (globals i no locals):
 - **Variables locals**: les que pertanyen a l'àmbit de la subrutina (i que poden ser accessibles a nivells inferiors)
 - **Variables globals**: les que pertanyen a l'àmbit del programa principal.
 - **Variables no locals**: les que pertanyen a un àmbit superior al de la subrutina, però que no són globals.
- Si el programa conté només funcions que no contenen funcions, totes les variables lliures són **variables globals**.
- Però si el programa conté una funció que conté altra/es funcions, les variables lliures d'aquestes "subfuncions" poden ser **globals** (si pertanyen al programa principal) o **no locals** (si pertanyen a la funció).
- Per identificar explícitament les variables globals i no locals s'utilitzen les paraules reservades **global** i **nonlocal (nonlocal)**.



VARIABLES LOCALS

- Variables locals: Existeixen en la pròpia funció, fins i tot quan en el programa hi hagi una variable amb el mateix nom, com mostra el següent exemple:

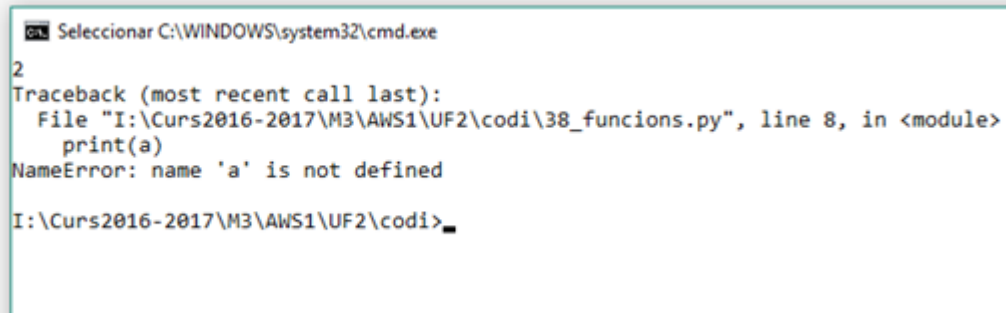
```
def funcio():  
    a = 2  
    print(a)  
  
a = 5  
funcio()  
print(a)
```



```
C:\WINDOWS\system32\cmd.exe  
2  
5  
I:\Curs2016-2017\M3\AWS1\UF2\codi>
```

- Les variables locals només existeixen en la pròpia funció i no són accessibles des de nivells superiors, com es pot veure en el següent exemple:

```
def funcio():  
    a = 2  
    print(a)  
  
funcio()  
print(a)
```



```
Selecció C:\WINDOWS\system32\cmd.exe  
2  
Traceback (most recent call last):  
  File "I:\Curs2016-2017\M3\AWS1\UF2\codi\38_funcions.py", line 8, in <module>  
    print(a)  
NameError: name 'a' is not defined  
I:\Curs2016-2017\M3\AWS1\UF2\codi>
```



VARIABLES LOCALS

- Si a l'interior d'una funció s'assigna valor a una variable que no s'ha declarat com global o no local, aquesta variable és local a tots els efectes. Per això el següent programa dóna error:

```
def funcio():  
    print(a)  
    a = 2  
    print(a)
```

```
a = 5  
funcio()  
print(a)
```

```
C:\WINDOWS\system32\cmd.exe  
Traceback (most recent call last):  
  File "I:\Curs2016-2017\M3\AWS1\UF2\codi\39_funcions.py", line 8, in <module>  
    funcio()  
  File "I:\Curs2016-2017\M3\AWS1\UF2\codi\39_funcions.py", line 2, in funcio  
    print(a)  
UnboundLocalError: local variable 'a' referenced before assignment  
I:\Curs2016-2017\M3\AWS1\UF2\codi>_
```



VARIABLES GLOBALS

- Si a una variable no se li assigna valor a una funció, Python la considera lliure i busca el seu valor en els nivells superiors d'aquesta funció, començant pel immediatament superior i continuant fins al programa principal. Si a la variable se li assigna valor en algun nivell intermedi la variable es considera no local i si se li assigna en el programa principal la variable es considera global, com mostren els següents exemples:
- En l'exemple següent, la variable lliure "a" de la funció subrutina () es considera global perquè obté el seu valor del programa principal:

```
def subrutina():  
    print(a)  
  
a = 5  
subrutina()  
print(a)
```



VARIABLES GLOBALS

- En l'exemple següent, la variable lliure "a" de la funció() es considera no local perquè obté el seu valor d'una funció intermèdia.

```
def subrutina():  
    def sub_subrutina():  
        print(a)  
  
    a = 3  
    sub_subrutina()  
    print(a)  
  
a = 4  
subrutina()  
print(a)
```

C:\WINDOWS\system32\cmd.exe

3
3
4

I:\Curs2016-2017\M3\AWS1\UF2\codi>



VARIABLES GLOBALS

- Si a una variable que Python considera lliure (perquè no se li assigna valor a la funció) tampoc se li assigna valor a nivells superiors, Python donarà un missatge d'error, com mostra el programa següent:

```
def subrutina():  
    print(a)
```

```
subrutina()  
print(a)
```

C:\WINDOWS\system32\cmd.exe

Traceback (most recent call last):

File "I:\Curs2016-2017\M3\AWS1\UF2\codi\42_funcions.py", line 6, in <module>
 subrutina()

File "I:\Curs2016-2017\M3\AWS1\UF2\codi\42_funcions.py", line 3, in subrutina
 print(a)

NameError: name 'a' is not defined

I:\Curs2016-2017\M3\AWS1\UF2\codi>_



VARIABLES DECLARADES global O nonlocal

- Si volem assignar valor a una variable en una subrutina, però no volem que Python la consideri local, hem de declarar-la en la funció com global o **nonlocal**, com mostren els exemples següents:

```
def subrutina():  
    global a  
    print(a)  
    a = 1
```

```
a = 5  
subrutina()  
print(a)
```

```
C:\WINDOWS\system32\cmd.exe  
5  
1  
I:\Curs2016-2017\M3\AWS1\UF2\codi>
```




VARIABLES DECLARADES **global** O **nonlocal**

- En l'exemple següent la variable es declara com nonlocal, perquè el seu valor sigui el de la funció intermèdia:

```
def subrutina():  
    def sub_subrutina():  
        nonlocal a  
        print(a)  
        a = 1  
  
    a = 3  
    sub_subrutina()  
    print(a)  
  
a = 4  
subrutina()  
print(a)
```

```
C:\WINDOWS\system32\cmd.exe  
3  
1  
4  
  
I:\Curs2016-2017\M3\AWS1\UF2\codi>
```



VARIABLES DECLARADES **global** O **nonlocal**

- Si a una variable declarada global o nonlocal en una funció no se li assigna valor en el nivell superior corresponent, Python donarà un error de sintaxi, com mostra el programa següent:

```
def subrutina():  
    def sub_subrutina():  
        nonlocal a  
        print(a)  
        a = 1
```

```
    sub_subrutina()  
    print(a)
```

```
a = 4  
subrutina()  
print(a)
```

C:\WINDOWS\system32\cmd.exe

File "I:\Curs2016-2017\M3\AWS1\UF2\codi\45_funcions.py", line 3
nonlocal a
SyntaxError: no binding for nonlocal 'a' found

I:\Curs2016-2017\M3\AWS1\UF2\codi>



nonlocal

Exemple 1

```
def f1():  
    def f2():  
        nivell2 = 2  
        print(nivell0, nivell1, nivell2)  
  
    nivell1 = 1  
    f2()  
    print(nivell0, nivell1)
```

```
nivell0 = 0  
f1()  
print(nivell0)
```

C:\WINDOWS\system32\cmd.exe

```
0 1 2  
0 1  
0
```

- Observa que la definició de la funció **f1()** inclou la definició de la funció **f2()**.
- Hem definit tres variables en àmbits diferents: *nivell0* a nivell de mòdul, *nivell1* dins de la funció **f1()** i *nivell2* dins **f2()**, que al seu torn està imbricada dins de **f1()**.
- En **f2()**, s'hi crea la variable local *nivell2* i s'imprimeix el seu valor, al costat del de les variables *nivell0* i *nivell1*, que pertanyen, respectivament, al mòdul i a l'àmbit de **f1()**.



VISIBILITAT

- Una funció pot accedir com a consulta a totes les variables dels àmbits en què està continguda la seva definició. `f2()` està continguda en `f1()`, que al seu torn ho està en el mòdul principal.
- La funció `f1()`, a més de definir `f2()`, crea la variable local `nivell1`, invoca `f2()` i imprimeix les variables `nivell0` i `nivell1`.
- Finalment, el mòdul principal defineix `f1()`, crea `nivell0`, invoca `f1()` i imprimeix a continuació el valor de `nivell0`.
- Des del mòdul principal, qualsevol intent d'accedir a `nivell1` o a `nivell2` resultaria un error, de la mateixa manera que des `f1()` tampoc podríem accedir a `nivell2`.
- Per contra, en sentit contrari, anant de menys a més si podem accedir a les variables, però només en **mode consulta**.



EXEMPLE 1

- Si des de **f1()** voldríem no només **consultar**, sinó a més **modificar** el valor de la variable a nivell de mòdul, *nivell0*. Hem d'utilitzar el qualificatiu global, exemple:

```
# Exemple 1

def f1():

    def f2():
        nivell2 = 2
        print(nivell0, nivell1, nivell2)

    nivell1 = 1
    global nivell0
    nivell0 = 5
    f2()
    print(nivell0, nivell1)

nivell0 = 0
f1()
print(nivell0)
```

```
cmd C:\WINDOWS\system32\cmd.exe
```

```
5 1 2
5 1
5
```



EXEMPLE 2

- Des de la funció més interna, **f2()**, també podríem definir la variable de mòdul *nivell0* usant el modificador **global**:

```
# Ejemplo 2

def f1():

    def f2():
        nivell2 = 2
        global nivell0
        nivell0 = 6
        print(nivell0, nivell1, nivell2)

    nivell1 = 1
    f2()
    print(nivell0, nivell1)

nivell0 = 0
f1()
print(nivell0)
```

```
C:\WINDOWS\system32\cmd.exe
6 1 2
6 1
6
```



EXEMPLE 3

- Si volem definir la variable *nivell1* des de dins de *f2()*? La resposta és **NONLOCAL**. Exemple:

Exemple 3

```
def f1():
```

```
    def f2():
```

```
        global nivell1
```

```
        nivell1 = 7
```

```
        nivell2 = 2
```

```
        print(nivell0, nivell1, nivell2)
```

```
    nivell1 = 1
```

```
    f2()
```

```
    print(nivell0, nivell1)
```

```
nivell0 = 0
```

```
f1()
```

```
print(nivell0)
```

```
C:\WINDOWS\system32\cmd.exe
```

```
0 7 2
```

```
0 1
```

```
0
```

La primera línia correspon a la impressió de *f2()*. Com es veu, s'ha imprès el valor 7, corresponent a *nivell1*. No obstant això, la següent línia, que correspon a la impressió pròpia de *f1()*, un instant després, ens demostra que el valor de *nivell1* no ha estat canviat, mantenint el seu valor 1 dins de *f1*.



EXEMPLE 4: **nonlocal**

- La qüestió és que el qualificatiu global només pot usar-se per a variables globals a nivell de mòdul. Per poder definir la variable a nivell de funció nivell1, necessitem un nou qualificador: **nonlocal**.

Exemple 4

```
def f1():
```

```
    def f2():
```

```
        nonlocal nivell1
```

```
        nivell1 = 7
```

```
        nivell2 = 2
```

```
        print(nivell0, nivell1, nivell2)
```

```
    nivell1 = 1
```

```
    f2()
```

```
    print(nivell0, nivell1)
```

```
nivell0 = 0
```

```
f1()
```

```
print(nivell0)
```

```
C:\WINDOWS\system32\cmd.exe
```

```
0 7 2
```

```
0 7
```

```
0
```

Aquesta vegada sí: hem aconseguit modificar *nivell1* des de dins de *f2()*.



global / nonlocal

- Ara modifiquem, des de dins de *f2()* tant nivell0 com a nivell1.

```
# Exemple 5

def f1():
    def f2():
        global nivell0
        nonlocal nivell1
        nivell0 = 8
        nivell1 = 9
        nivell2 = 2
        print(nivell0, nivell1, nivell2)

    nivell1 = 1
    f2()
    print(nivell0, nivell1)

nivell0 = 0
f1()
print(nivell0)
```

```
C:\WINDOWS\system32\cmd.exe
```

```
8 9 2
8 9
8
```



RESUM

- Una variable creada a l'interior d'una funció és **local** i només existeix dins d'ella.
- Les variables creades a nivell de mòdul són **globals** en el sentit que són accessibles, a més de pel mateix mòdul, per totes les funcions definides dins d'ell. No obstant això, només són modificables pel propi mòdul; les funcions només poden consultar el seu valor.
- Per a modificar una variable **global** des de dins d'una funció cal que aparegui declarada com global en el cos de la funció.
- Utilitza **global** per poder modificar una variable creada a nivell de mòdul des de dins d'una funció definida en aquest mòdul, encara que estigui imbricada dins d'una altra funció; utilitza **nonlocal** per modificar una variable creada a nivell de funció des d'una altra funció definida dins d'aquella.