

## Sesi3n 2

### Introducci3n

#### 3.1. Cadena de text (string)

```
aws2-08@HPi5-28:~$ python
Python 2.7.12 (default, Jul 1 2016, 15:12:24)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> cadena1= "Hola"
>>> cadena_multilinea= """
... Aix3 es una cadena
... de dues l3nies
... """
>>>
```

#### 3.2. Llista

```
>>> llista_buida=[]
>>> llista1 = ['cadena de text', 15, 2.8, 'altres dades', 25]
>>> llista1[1]
15
>>> llista1[0]
'cadena de text'
>>>
```

```
>>> llista1[2]= 3.8
>>> llista1[1:3]
[15, 3.8]
>>>
```

```
>>> list("Good morning Bietnam")
['G', 'o', 'o', 'd', ' ', 'm', 'o', 'r', 'n', 'i', 'n', 'g', ' ', 'B', 'i', 'e',
't', 'n', 'a', 'm']
>>> list=list("Python")
>>> list
['P', 'y', 't', 'h', 'o', 'n']
>>> list[3]
'h'
>>>
```

list=listilla("Python")

#### 4.1. Diccionari

```
>>> diccionari1 = {'clau1': 12, 'clau2': 23, 'clau6': 34}
>>> diccionari1['clau2']
23
```

Darle un nuevo valor:

```
>>> diccionari1['clau1'] = 12345
>>> diccionari1['clau1']
12345
```

Crear un diccionario:

```
>>> dict (['Jordan', 23], ['Messi', 10])
{'Messi': 10, 'Jordan': 23}
>>>
```

```
1  #!/usr/bin/python
2
3  a=2
4  b=3
5  if a==b:
6      print"son iguals"
7  else:
8      print"son diferents"
9
```

```
aws2-08@HPi5-28:~/Escriptori$ python condicional1.py
son diferents
aws2-08@HPi5-28:~/Escriptori$ python condicional1.py
son diferents
aws2-08@HPi5-28:~/Escriptori$
```

## 2.2- Estructura condicional

- Bucle if

```
1  #!/usr/bin/python
2  #versiol
3
4  edat_usuari=int(raw_input("edat?"))
5  print"tens "
6  if edat_usuari==1:
7      print "un any"
8  else:
9      if edat_usuari==2:
10         print "dos anys"
11     else:
12         if edat_usuari==3:
13             print "tres anys"
14         else:
15             if edat_usuari==4:
16                 print "quatre anys"
17             else:
18                 print "mes de 4 anys"
19
```

```
aws2-08@HPi5-28:~/Escriptori$ python if.py
edat?3
tens
tres anys
aws2-08@HPi5-28:~/Escriptori$
```

- Bucle while:

```
#!/usr/bin/python

numero=1
sortir = False

while sortir==False:
    if numero%15==0 and numero%16==0 and numero%26==0:
        sortir=True
    else:
        numero=numero+1
        if numero>10000:
            sortir=True
    print "Un numero divisible entre 15, 16 i 26: "
if numero>10000:
    print "No l'he trobat"
else:
    print numero

aws2-08@HPi5-28:~/Escriptori$ python buclewhile.py
Un numero divisible entre 15, 16 i 26:
3120
aws2-08@HPi5-28:~/Escriptori$
```

- Sacar todos los divisibles:

```
1  #!/usr/bin/python
2
3  ##Todos los números divisibles de los siguientes numeros
4
5  numero=1
6  sortir = False
7
8  while sortir==False:
9      if numero%15==0 and numero%16==0 and numero%26==0:
10         print numero
11         #sortir=True
12     else:
13
14         if numero>10000:
15             sortir=True
16
17         numero=numero+1
18     print "Un numero divisible entre 15, 16 i 26: "
19 if numero>10000:
20     print "No l'he trobat"
21 else:
22     print numero

aws2-08@HPi5-28:~/Escriptori$ python buclewhile1.py
3120
6240
9360
Un numero divisible entre 15, 16 i 26:
No l'he trobat
aws2-08@HPi5-28:~/Escriptori$
```

## - Bucle For:

El `\r` te lo hace en varias líneas, si solo dejamos el `\` te lo hace en la misma línea

```
#!/usr/bin/python
for i in ["/", "-", "|", "\\", "|"]:
    print "%s \r" % i
    raw_input()
```

```
SyntaxError: EOL while scanning string literal
aws2-08@HPI5-28:~/Escriptori$ python buclefor.py
/
-
|
\
|
```

Para que se genere en una misma línea y en el mismo punto como aquél ejercicio en C, se hace mediante un `system cls`

## - Funciones:

```
#!/usr/bin/python
def imprimeix_taula(numero):
    for i in range(10):
        print "%d*%d=%d" % (numero, i, numero*i)

    imprimeix_taula(3)
    print "\n"
    imprimeix_taula(9)
    print "\n"
```

```
aws2-08@HPI5-28: ~/Escriptori
aws2-08@HPI5-28:~/Escriptori$ python funcion.py
3*1=0
3*1=3
3*1=6
3*1=9
3*1=12
3*1=15
3*1=18
3*1=21
3*1=24
3*1=27

9*1=0
9*1=9
9*1=18
9*1=27
9*1=36
9*1=45
9*1=54
9*1=63
9*1=72
9*1=81
```

compile "buclewhile1.py" (en el directorio: /home/info.dom/aws2)

## Ex1 – Funciones

```
def cheese_and_crackers (cheese_count, boxes_of_crackers):  
    print "You have %d cheeses!" % cheese_count  
    print "You have %d boxes of crackers!" % boxes_of_crackers  
    print "Man that's enough for a party!"  
    print "Get a blanket. \n"  
  
    print "We can just give the function numbers directly:"  
    cheese_and_crackers(20, 30)  
  
    print "Or, we can use variables from our script:"  
    amount_of_cheese=10  
    amount_of_crackers=50  
  
    cheese_and_crackers(amount_of_cheese, amount_of_crackers)  
  
    print "We can even do math inside too:"  
    cheese_and_crackers(10+20, 5+6)  
  
    print "And we can combine the two, variables and math:"  
    cheese_and_crackers(amount_of_cheese+100, amount_of_crackers+1000)
```

---

```
sergio@sergio-desktop:~/Escritorio$ python ex1.py
```

```
We can just give the function numbers directly:
```

```
You have 20 cheeses!  
You have 30 boxes of crackers!  
Man that's enough for a party!  
Get a blanket.
```

```
Or, we can use variables from our script:
```

```
You have 10 cheeses!  
You have 50 boxes of crackers!  
Man that's enough for a party!  
Get a blanket.
```

```
We can even do math inside too:
```

```
You have 30 cheeses!  
You have 11 boxes of crackers!  
Man that's enough for a party!  
Get a blanket.
```

```
And we can combine the two, variables and math:
```

```
You have 110 cheeses!  
You have 1050 boxes of crackers!  
Man that's enough for a party!  
Get a blanket.
```

---

### Executar-lo. Explicar d'on surten en cada cas els valors que es passen a la funció.

- Lo primero que debemos saber es que el código de “def” es una función que podremos ir aprovechando. Lo primero que veremos por pantalla será el primer print.
- Nos imprime la primera frase *We can just give...* y a continuación se llama a la función `cheese_and_crackers` pasándole por parámetro los valores 20 y 30 para cada variable de la función. `cheese_count` valdrá 20 y `boxes_crackers` valdrá 30. Y proseguirá con los print que se siguen en la función.
- De la misma forma funciona la siguiente función. Tienes unas variables propias y luego por parámetros le pasas los valores a la función.
- En la siguiente forma, es más de lo mismo, sólo que por parámetro le pasas una suma que calculará y te la imprimirá ya hecha.
- La siguiente igual, tomas las variables propias anteriores y le sumas un número

Programar una variació del programa en què es demani a l'usuari els nombres de formatges i aperitius. Utilitzar la funció `int()` per transformar a enter el valor retornat per `raw_input()`.

```
def cheese_and_crackers (cheese_count, boxes_of_crackers):  
    print "You have %d cheeses!" % cheese_count  
    print "You have %d boxes of crackers!" % boxes_of_crackers  
    print "Man that's enough for a party!"  
    print "Get a blanket. \n"  
  
print("Introduce el num de cheese y el num de crackers")  
amount_of_cheese=int(raw_input())  
amount_of_crackers=int(raw_input())  
cheese_and_crackers(amount_of_cheese, amount_of_crackers)  
  
amount_of_cheese=int(raw_input("Introduce amount of cheese"))  
amount_of_crackers=int(raw_input("Introduce amount of crackers"))  
cheese_and_crackers(amount_of_cheese, amount_of_crackers)
```

```
sergio@sergio-desktop:~/Escritorio$ python ex2.py  
Introduce el num de cheese y el num de crackers  
26  
68  
You have 26 cheeses!  
You have 68 boxes of crackers!  
Man that's enough for a party!  
Get a blanket.  
  
Introduce amount of cheese44  
Introduce amount of crackers55  
You have 44 cheeses!  
You have 55 boxes of crackers!  
Man that's enough for a party!  
Get a blanket.
```

## Ex2 – Funciones con retorno

```
def add(a,b):
    print "ADDING %d + %d" % (a,b)
    return a+b

def subtract(a,b):
    print "SUBTRACTING %d - %d" % (a,b)
    return a-b

def multiply(a,b):
    print "MULTIPLYING %d * %d" % (a,b)
    return a*b

def divide(a,b):
    print "DIVIDING %d / %d" % (a,b)
    return a/b

print "Let's do some math with just functions!"

age= add(30,5)
height=subtract(78,4)
weight = multiply(90,2)
iq = divide(100,2)

print "Age: %d, Height: %d, Weight: %d, IQ: %d" % (age, height, weight,iq)

#A puzzle for the extra credit, type it in anyway
print "Here is a puzzle"

what=add(age,subtract(height,multiply(weight,divide(iq,2))))
```

Executar-lo. Explicar com es calcula el valor que acaba en la variable *what*.

```
sergio@sergio-desktop:~/Escritorio$ python ex3.py
Let's do some math with just functions!
ADDING 30 + 5
SUBTRACTING 78 - 4
MULTIPLYING 90 * 2
DIVIDING 100 / 2
Age: 35, Height: 74, Weight: 180, IQ: 50
Here is a puzzle
DIVIDING 50 / 2
MULTIPLYING 180 * 25
SUBTRACTING 74 - 4500
ADDING 35 + -4426
That becomes: -4391 Can you do it by hand?
sergio@sergio-desktop:~/Escritorio$
```

El valor de la variable *what* se calcula mediante una combinación de varias funciones que hay en el inicio del programa y que van haciendo cálculos. *What* ve añadidas todas estas funciones mediante la función “add” y obtiene un resultado

Programar una variació del programa en què es demani a l'usuari els valors (nombres reals) per fer les operacions. Utilitzar la funció `float()` per transformar en nombre real el valor retornat per `raw_input()`.

```
print "Let's do some math with just functions!"

age1 = float(raw_input("Introduce una edad"))
age2 = float(raw_input("Introduce una segunda edad"))

height1=float(raw_input("Introduce una altura"))
height2=float(raw_input("Introduce una segunda altura"))

weight1=float(raw_input("Introduceun peso"))
weight2=float(raw_input("Introduce un segundo peso"))

iq1=float(raw_input("Introduce un valor para IQ"))
iq2=float(raw_input("Introduce un segundo palor para IQ"))

age= add(age1,age2)
height=subtract(height1,height2)
weight = multiply(weight1,weight2)
iq = divide(iq1,iq2)

print "Age: %d, Height: %d, Weight: %d, IQ: %d" % (age, height, weight,iq)

#A puzzle for the extra credit, type it in anyway
print "Here is a puzzle"
```

```
sergio@sergio-desktop:~/Escritorio$ python ex3.1.py
Let's do some math with just functions!
Introduce una edad23
Introduce una segunda edad24
Introduce una altura1.74
Introduce una segunda altura1.69
Introduceun peso67
Introduce un segundo peso76
Introduce un valor para IQ100
Introduce un segundo palor para IQ2
ADDING 23 + 24
SUBTRACTING 1 - 1
MULTIPLYNG 67 * 76
DIVIDING 100 / 2
Age: 47, Height: 0, Weight: 5092, IQ: 50
Here is a puzzle
DIVIDING 50 / 2
MULTIPLYNG 5092 * 25
SUBTRACTING 0 - 127300
ADDING 47 + -127299
That becomes: -127252.95 Can you do it by hand?
sergio@sergio-desktop:~/Escritorio$
```



### Ex3 - Funcions amb retorn.

```
print "Let's practice everything"
print 'You\'d need to know \'bout escapes with \\ that do \n newlines and \t tabs'

poem="""
\tTe lovely world
with logic so firmly plantes
cannot discern \n the needs of love
nor comprehend passion from intuition
and requires and explanation
\n\t\twhere there is none
"""

print "-----"
print poem
print "-----"

five = 10-2+3-6
print "This should be five: %s" % five

def secret_formula(started):
    jelly_beans = started*500
    jars = jelly_beans/1000
    crates=jars/100
    return jelly_beans,jars,crates

start_point = 10000
beans,jars,crates = secret_formula(start_point)

print "With a starting point of: %d" % start_point
print "We'd have %d beans, %d jars, and %d crates" % (beans, jars, crates)

start_point = start_point /10

print "We can also do that this way:"
print "We'd have %d beans, %d jars, and %d crates" % secret_formula (start_point)
```

#### Executar-lo. Explicar com funciona el retorn de múltiples valors.

Secret formula es una variable que se ha llamado fuera de ella. Dentro de la función se han realizado una serie de calculos que les son retornados a los prints. A diferencia de los ejercicios anteriores, estos valores devueltos, en el print se deben colocar entre paréntesis

```
sergio@sergio-desktop:~/Escritorio$ python ex4.py
Let's practice everything
You'd need to know 'bout escapes with \ that do
newlines and    tabs
-----

        Te lovely world
with logic so firmly plantes
cannot discern
the needs of love
nor comprehend passion from intuition
and requires and explanation

                where there is none

-----
This should be five: 5
With a starting point of: 10000
We'd have 5000000 beans, 5000 jars, and 50 crates
We can also do that this way:
We'd have 500000 beans, 500 jars, and 5 crates
sergio@sergio-desktop:~/Escritorio$
```

#### Explicar quantes variables anomenades jars i crates hi ha i la seva visibilitat.

Hay dos “jar” y dos “crates”, cada uno de ellos es visible en una parte del código, es decir, un jar y un crates es visible dentro de la función y otro siempre lo será, al estar fuera de la función.

#### Ex4 - Funcions amb retorn que criden altres funcions

```
def break_words(stuff):
    """This function will break up words for us"""
    words = stuff.split(' ')
    return words

def sort_words(words):
    """Sorts the words"""
    return sorted(words)

def print_first_word(words):
    """Prints the first word after popping it off"""
    word = words.pop(0)
    print word

def print_last_word(words):
    """Prints the last word after popping it off"""
    word = words.pop(-1)
    print word

def sort_sentence(sentence):
    """Takes in a full sentence and returns the sorted words"""
    words = break_words(sentence)
    return sort_words(words)

def print_first_and_last(sentence):
    """Prints the first and last words of the sentence"""
    words = break_words(sentence)
    print_first_word(words)
    print_last_word(words)
```

Provisionalment, afegir codi que cridi a les funcions, i executar-lo, per assegurar que no hi ha cap error sintàctic. Els paràmetres anomenats statement han de rebre una frase com a argument. Quan estigui tot correcte, deixar només les línies que es mostren a dalt.

```
phrases = break_words('Hello world')
print phrases

phrases_sort = sort_words(phrases)
print phrases_sort
print phrases
print_first_word(phrases)
print phrases
print_last_word(phrases)
print phrases
phrase2 = sort_sentence('Good morning Vietnam')
print phrase2
print_first_and_last('Good morning Vietnam')
print_first_and_last_sorted('Good morning Vietnam')
```

```
sergio@sergio-desktop:~/Escritorio$ python ex4.1.py
['Hello', 'world']
['Hello', 'world']
['Hello', 'world']
Hello
['world']
world
[]
['Good', 'Vietnam', 'morning']
Good
Vietnam
Good
morning
sergio@sergio-desktop:~/Escritorio$
```

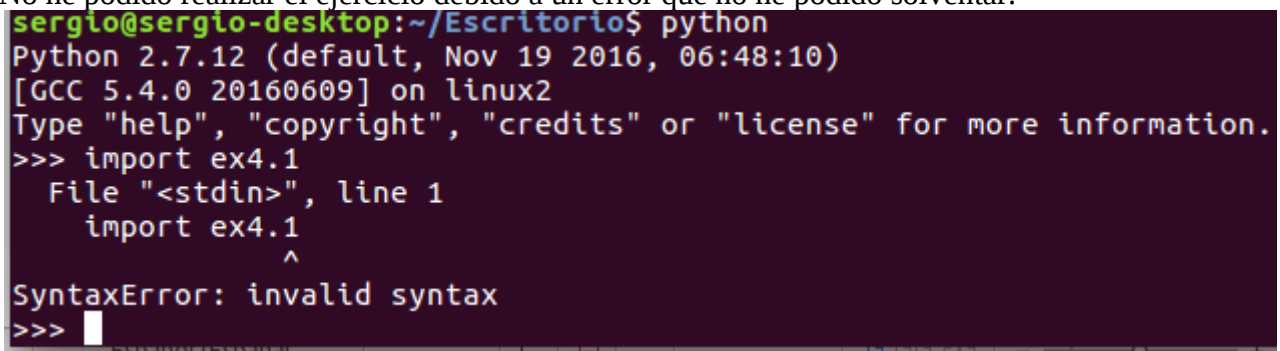
**Fer un esquema, gràficament o amb text, que mostri les cadenes de crides de funcions. Comprendre i explicar què fa cadascuna de les funcions, les codificades a l'script i les predefinides `sorted`, `pop` i `split`.**

- `break_words`: hace una lista con las palabras de la frase introducida.
- `phrases_sort`: ordena las palabras de la lista
- `print_first_word`: Elimina la primera palabra de la lista introducida y le asigna una nueva variable
- `print_last_word`: elimina la última palabra de la lista y le otorga una variable.
- `sort_sentence`: La frase introducida se convierte en una lista, luego `sort_sentence` ordena esta lista de elementos
- `print_first_and_last`: Aquí la frase introducida se convierte en una lista que además, coge la primera y la última palabra.
- `print_first_and_last_sorted`: La frase introducida se convierte en una lista, que luego se ordena y coge la primera y la última palabra de la lista.
- `sorted`: simplemente, ordena la frase
- `pop`: devuelve el elemento eliminado de la lista
- `split`: divide las frases y luego las podremos incluir en una lista

### **Ex5 – Funciones con retorno que llaman a otras funciones**

Utilitzar l'script creat en l'exercici anterior i executar-lo interactivament, des de l'interpret de Python.

No he podido realizar el ejercicio debido a un error que no he podido solventar:



```
sergio@sergio-desktop:~/Escritorio$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import ex4.1
      File "<stdin>", line 1
        import ex4.1
            ^
SyntaxError: invalid syntax
>>>
```

## Ex6 – Control de flux

```
print "You enter a dark room with two doors. Do you go through door #º or door @?"
door = raw_input("> ")
if door == "1":
    print "There's a giant bear here eating a cheese cake. What do you do?"
    print ".1. take the cake"
    print "2. Scream at the bear"

    bear= raw_input("> ")

    if bear == "1":
        print "The bear eats your face off. Good job!"
    elif bear == "2":
        print "The bear eats your legs off. Good job!"
    else:
elif door=="2":
    print "You stare into the endless abyss at Thulu's retina"
    print "1 blueberries"
    print "2 Yellow jacket clothespins"
    print "3 Undertanding revolvers yelling melodies"

    insanity = raw_input("> ")

    if insanity == "1" or insanity == "2":
        print "Your body survives powered by a mind of jello. Good job!"
    else:
        print "The insanity rots your eyes into a pool of muck. Good job!"
```

Executar i comprendre el programa.

Modificar el programa de manera que el joc es repeteixi indefinidament fins que l'usuari digui que no vol continuar.

```
salir = False
while salir == False:

    print "You enter a dark room with two doors. Do you go through door #º or door @?"
    door = raw_input("> ")

    if door == "1":
        print "There's a giant bear here eating a cheese cake. What do you do?"
        print ".1. take the cake"
        print "2. Scream at the bear"

        bear= raw_input("> ")

        if bear == "1":
            print "The bear eats your face off. Good job!"
        elif bear == "2":
            print "The bear eats your legs off. Good job!"
        else:
    elif door=="2":
        print "You stare into the endless abyss at Thulu's retina"
        print "1 blueberries"
        print "2 Yellow jacket clothespins"
        print "3 Undertanding revolvers yelling melodies"

        insanity = raw_input("> ")

        if insanity == "1" or insanity == "2":
            print "Your body survives powered by a mind of jello. Good job!"
        else:
            print "The insanity rots your eyes into a pool of muck. Good job!"
    else:
        print "You stumble around and fall on a knife and die. Good job!"

    salir = raw_input("Desea salir del programa?. Escriba S o N")

    if salir == "S":
        salir=True
    else:
        salir=False
```

## Ex7 – Bucle for

```
the_count = [1,2,3,4,5]
fruits = ['apples','oranges','pears','apricots']
change = [1,'pennies',2,'dimes',3,'quartets']

for number in the_count:
    print "this is count %d" % number

for fruit in fruits:
    print "A fruit of type: %s" % fruit

for i in change:
    print "I got %r" %i

elements = []

for i in range(0,6):
    print "Adding %d to the list" %i
    elements.append(i)

for i in elements:
    print "Element was: %d" %i
```

```
sergio@sergio-desktop:~/Escritorio$ python ex7.py
this is count 1
this is count 2
this is count 3
this is count 4
this is count 5
A fruit of type: apples
A fruit of type: oranges
A fruit of type: pears
A fruit of type: apricots
I got 1
I got 'pennies'
I got 2
I got 'dimes'
I got 3
I got 'quartets'
Adding 0 to the list
Adding 1 to the list
Adding 2 to the list
Adding 3 to the list
Adding 4 to the list
Adding 5 to the list
Element was: 0
Element was: 1
```

**Executar-lo.** Explicar el funcionament de cadascun dels bucles for que hi ha.  
**Explicar l'efecte de la funció (en realitat, és un mètode) append**

· El primer bucle for imprime la lista de números. Number es un índice. El segundo bucle imprime la lista de frutas que tiene entre corchetes. En el tercer bucle imprime la lista change, para ello usa el %r, ya que es una lista llena de strings e ints. Se crea una lista vacía llamada elements que se llenará en el cuarto bucle mediante la función append(). Añade elementos y estos se van añadiendo consecutivamente al final de la lista. En el quinto bucle se recorre la lista y se printan los números que esta lista contiene.

## Ex8 – Bucle while

```
i=0
numbers = []

while i<6:
    print "At the top i is %d" %i
    numbers.append(i)

    i=i+1
    print "Numbers now:", numbers
    print "At the bottom i is %d" %i

print "The numbers: "

for num in numbers:
    print num
```

```
sergio@sergio-desktop:~/Escritorio$ python ex8.py
At the top i is 0
Numbers now: [0]
At the bottom i is 1
At the top i is 1
Numbers now: [0, 1]
At the bottom i is 2
At the top i is 2
Numbers now: [0, 1, 2]
At the bottom i is 3
At the top i is 3
Numbers now: [0, 1, 2, 3]
At the bottom i is 4
At the top i is 4
Numbers now: [0, 1, 2, 3, 4]
At the bottom i is 5
At the top i is 5
Numbers now: [0, 1, 2, 3, 4, 5]
At the bottom i is 6
The numbers:
0
1
2
3
```

### Executar-lo. Explicar el funcionament de cadascun dels bucles que hi ha.

Se crea una lista vacía llamada numbers. El bucle while se mantendrá mientras i sea menor de 6. Se irán printando numeros del 0 al 6 y que luego se irán añadiendo a la lista numbers. El bucle se va autoincrementando. Se printa in índice (posición) y los números. Finalmente, en el bucle final se printan los números de la lista uno a uno