Sesión 1
Ex1 - Strings
Vamos al «Geany» y escribimos el texto:

```
#!/usr/bin/python
print "Hello World!"
print "Hello Again"
print "I like typing this."
print "This is fun."
print 'Yay! Printing.'
print "I'd much rather you 'not'."
print 'I "said" do not touch this.'

```

En el terminal tan sólo debemos dirigirnos al directorio «cd Escriptori» y ejecutar lo que hay escrito en la captura de pantalla

```
super@HPi5-28:~/Escriptori$ chmod 755 ex1.py
super@HPi5-28:~/Escriptori$ ./ex1.py
Hello World!
Hello Again
I like typing this.
This is fun.
Yay! Printing.
I'd much rather you 'not'.
I "said" do not touch this.
super@HPi5-28:~/Escriptori$
```

Els strings es poden delimitar de dues maneres. Explicar quines són i la combinació que se'n fa a l'exercici.
- Los string se pueden delimitar mediante comillas y comillas simples. " y ‘
- Se combina de formas diferentes. Encontramos que si a principio y a final hay unas comillas "" se printa toda la cadena de carácteres que haya ahí dentro. Ocurre lo mismo si a principio y a final hay las simples, pues imprimirá todo el contenido que haya entre ‘ y ‘

Ex2- Comentarios

```
#!/usr/bin/python
#A comment, this is so you can read your program later.
#Anything after the # is ignored by python.

print "I could have code like this." #and the comment after is ignored

#you can also use a comment to "disable" or comment out a piece of code:
#print "this won't run."

print "this will run."
```

```
super@HPi5-28:~/Escriptori$ chmod 755 ex2.py
super@HPi5-28:~/Escriptori$ ./ex2.py
I could have code like this.
this will run.
super@HPi5-28:~/Escriptori$
```

Explicar com s'escriuen comentaris en Python. Si trobem la instrucció. Quin efecte hi tindrà el caracter #?
- Los comentarios en Python se hacen mediante #. En el caso del ejemplo, se está mostrando por pantalla Hi # there, por lo que el símbolo # tiene un efecto nulo

Ex3- Operaciones aritméticas y comparaciones

```
#!/usr/bin/python

print "I will now count my chickens:"

print "Hens", 25+30/6
print "Roosters", 100-25*3%4

print "Now I will count the eggs:"
print 3+2+1-5+4%2-1/4+6

print "Is it true that 3+2<5-7?"
print 3+2<5-7

print "What is 3+2?", 3+2
print "What is 5 - 7?", 5-7

print "Oh, that's why it's False."
print "How about some more."

print "Is it greater?", 5>-2
print "Is it greater or equal?", 5>=-2
print "Is it less or equal?", 5<= -2
```

```
super@HPi5-28:~/Escriptori$ chmod 755 ex3.py
super@HPi5-28:~/Escriptori$ ./ex3.py
I will now count my chickens:
Hens 30
Roosters 97
Now I will count the eggs:
7
Is it true that 3+2<5-7?
False
What is 3+2? 5
What is 5 - 7? -2
Oh, that's why it's False.
How about some more.
Is it greater? True
Is it greater or equal? True
Is it less or equal? False
super@HPi5-28:~/Escriptori$
```

Explicar perquè s'obtenen els resultats True i False de l'exercici.

- Porque el signo < devuelve un boolean, pues sólo pueden ser true o false, ya que se están comparando operaciones

Ex4- Variables y nombres. Normas de escritura

```
1  #!/usr/bin/python
2
3  cars = 100
4  space_in_a_car = 4.0
5  drivers = 30
6  passengers = 90
7  cars_not_driven = cars - drivers
8  cars_driven = drivers
9  carpool_capacity = cars_driven*space_in_a_car
10 verage_passengers_per_car = passengers / cars_driven
11
12 print "there are", cars, "cars avaible."
13 print "There are only", drivers, "drivers avaible."
14 print "There wil be", cars_not_driven, "empty cars today"
15 print "We can transport", carpool_capacity, "people today"
16 print "We have", passengers, "to carpool today"
17 print "We need to put about", average_passengers_per_car, "in each car"
18
```

Resultado:

```
aws2-08@HPi5-28:~/Escriptori$ python ex4.py
there are 100 cars avaible.
There are only 30 drivers avaible.
There wil be 70 empty cars today
We can transport 120.0 people today
We have 90 to carpool today
We need to put about 3 in each car
aws2-08@HPi5-28:~/Escriptori$
```

Utilitzar la funció type i esbrinar el tipus que conté cada variable al final de l'execució de l'script.

```
18
19 print "\r"
20
21 print type(cars),cars
22 print type(space_in_a_car), space_in_a_car
23 print type(drivers), drivers
24 print type(passengers), passengers
25 print type(cars_not_driven), cars_not_driven
26 print type(carpool_capacity), carpool_capacity
27 print type(average_passengers_per_car), average_passengers_per_car
```

```
aws2-08@HPi5-28:~/Escriptori$ python ex4.py
there are 100 cars avaible.
There are only 30 drivers avaible.
There wil be 70 empty cars today
We can transport 120.0 people today
We have 90 to carpool today
We need to put about 3 in each car

<type 'int'> 100
<type 'float'> 4.0
<type 'int'> 30
<type 'int'> 90
<type 'int'> 70
<type 'float'> 120.0
<type 'int'> 3
```

Ex5- Variables y escritura con formato

```
my_name = 'Zed A. Shaw'
my_age = 35 # not a lie
my_height = 74 # inches
my_weight = 180 # lbs
my_eyes = 'Blue'
my_teeth = 'White'
my_hair = 'Brown'

print "Let's talk about %s." % my_name
print "He's %d inches tall." % my_height
print "He's %d pounds heavy." % my_weight
print "Actually that's not too heavy."
print "He's got %s eyes and %s hair" % (my_eyes, my_hair)
print "His teeth are usually %s depending on the coffee." % my_teeth

# this line is tricky, try to get it exactly right
print "If I add %d, %d, and %d I get %d." % (my_age, my_height, my_weight, my_age + my_height + my_weight)
```

```
aws2-08@HPi5-28:~/Escriptori$ python ex5.py
Let's talk about Zed A. Shaw.
He's 74 inches tall.
He's 180 pounds heavy.
Actually that's not too heavy.
He's got Blue eyes and Brown hair
His teeth are usually White depending on the coffee.
If I add 35, 74, and 180 I get 289.
```

Explicar quina és la funció de %, %s i %d en la sentència print

- La función «%» te imprime la variable, mientras que «%s» imprime una cadena de texto. El signo «%d» imprime un número entero. Funciona como C.

Ex6- Combinación de strings y formatos

```
x= "There are %d types of people." %10
#Se le asigna a la variable "x" un string
#Además, a la vez se le está asignando un número a la string

binary= "binary" #Se le signa a la variable binary el texto "binary"
do_not= "don't" #Lo mismo de antes

y= "Those who know %s and those who %s." % (binary, do_not)
#Se le asigna a la variable "y" un string
#A la vez se le asignan dos variable dentro del string que irán colocadas donde se encuentran los "%s"

print x
print y #Se imprimen las variables x e y

print "I said: %r." %x
print "I also said: %s" %y
#se imprime un string al que se le ha asignado una variable

hilarious= False #será una variable Boolean
joke_evaluation="Isn't that joke so funny?! %r" #se le asigna un string

print joke_evaluation % hilarious #Se imprimen las variables por orden.

w= "This is the left side of... "
e= "a string with a right side"
#Se les asigna un string
#A continuación se imprime
print w+e
```

```
aws2-08@HPi5-28:~/Escriptori$ python ex6.py
There are 10 types of people.
Those who know binary and those who don't.
I said: 'There are 10 types of people.'.
I also said: 'Those who know binary and those who don't.'.
Isn't that joke so funny?! False
This is the left side of...a string with a right side.
```

Posar comentaris al codi, explicant què fa cada sentència.

El format %r dóna el valor “raw”, cru, sense modificar, dels objectes o variables. Moltes vegades el resultat de %d i %r és el mateix, però no sempre. En general %r va bé per veure les dades quan es depuren els programes i els altres formats per mostrar-les a l'usuari.

Exemple on es veu clarament la diferència:

```
>>> import datetime
>>> x = datetime.date.today()
>>> print x
2013-01-11
>>>
>>>
>>> print "Today's date is %s ..." % x
Today's date is 2013-01-11 ...
>>>
>>> print "Today's date is %r ..." % x
Today's date is datetime.date(2013, 1, 11) ...
>>>
```

Ex7 – Impresión de Strings. Concatenación

```
print "Mary had a little lamb."
print "Its fleece was white as %s." % 'snow'
print "And everywhere that Mary went."
print "." * 10 # what'd that do?

end1 = "C"
end2 = "h"
end3 = "e"
end4 = "e"
end5 = "s"
end6 = "e"
end7 = "B"
end8 = "u"
end9 = "r"
end10 = "g"
end11 = "e"
end12 = "r"

# watch that comma at the end. try removing it to see what happens
print end1 + end2 + end3 + end4 + end5 + end6,
print end7 + end8 + end9 + end10 + end11 + end12
```

```
aws2-08@HPi5-28:~/Escriptori$ python ex7.py
Mary had a little lamb.
Its fleece was white as snow.
And everywhere that Mary went.
..........
Cheese Burger
```

Explicar la sentència de la línia 4 i què passa si es treu la coma final de la línia 20. Quina utilitat pot tenir?

- La sentencia de la línia 4 hace que se imprima el punto 10 veces gracias ala multiplicación. Si se le quita la coma, los dos prints se hacen en líneas diferentes y eso sirve para poder tener prints diferentes si se desea en una misma línia o no.

Ex8 – Más impresión de Strings

```
# Here's some new strange stuff, remember type it exactly.

days = "Mon Tue Wed Thu Fri Sat Sun"

months = "Jan\nFeb\nMar\nApr\nMay\nJun\nJul\nAug"

print "Here are the days: ", days

print "Here are the months: ", months

print """

There's something going on here.

With the three double-quotes.

We'll be able to type as much as we like.

Even 4 lines if we want, or 5, or 6.

"""
```

```
aws2-08@HPi5-28:~/Escriptori$ python ex8.py
Here are the days:  Mon Tue Wed Thu Fri Sat Sun
Here are the months:  Jan
Feb
Mar
Apr
May
Jun
Jul
Aug


There's something going on here.

With the three double-quotes.

We'll be able to type as much as we like.

Even 4 lines if we want, or 5, or 6.
```

Executar-lo. Explicar quin és l'efecte de \n i de les tres cometes dobles.

- Cuando se usa \n se realiza un salto de línia, peró cuando se usan las tres comillas dosbles (""") podemos escribir un texto en líneas diferentes.

Ex9- Carácteres escapados con la contrabarra

```
#!/usr/bin/python

tabby_cat = "\t I'm tabbed in"
persian_cat = "I'm split\non a line"
blackslash_cat = "I'm \\ a \\ cat"

fat_cat = """
I'll do a list:
\t* Cat food
\t* Fishies
\t* Catnip\n\t* Grass
"""

print tabby_cat
print persian_cat
print blackslash_cat
print fat_cat
```

```
aws2-08@HPi5-28:~/Escriptori$ python ex9.py
        I'm tabbed in
I'm split
on a line
I'm \ a \ cat

I'll do a list:
        * Cat food
        * Fishies
        * Catnip
        * Grass
```

Executar-lo. Explicar quin és l'efecte de la contrabarra en cadascun dels strings on apareix.
- La \t tabula el texto
- La \ entre medio de una string corta la línia
- La \\ doble, hace que no corte la línia y que además, aparezca el signo allí donde la hemos colocado
Además, \n también corta la línia, es como hacer un «intro»

Llistar les "seqüències d'escape" que hi ha a Python. Buscar a la documentació oficial: docs.python.org.
- No ha sigut possible poder llistar les seqüencies dins de la pàgina, però he trobat una web on estava tot ben explicat: https://123programando.wordpress.com/tag/secuencias-de-escape/

Secuencia de Escape	En pantalla…	O lo que es lo mismo…
\\	\	Una única barra invertida
\'	‘	Escape de comilla simple en una cadena
\"	“	Escape de comilla doble en una cadena
\a	BELL	ASCII bell
\b	BS	ASCII retroceso
\f	FF	ASCII avance de página
\n	LF	ASCII linefeed o salto de línea
\N{name}	Sólo Unicode	Carácter name en la base de datos Unicode
\r	CR	ASCII retorno de carro
\t	TAB	ASCII tabulación horizontal
\uxxxx	Sólo Unicode	Carácter con 16-bits hexadecimal y valor xxxx
\Uxxxxxxxx	Sólo Unicode	Carácter con 32-bits hexadecimal y valor xxxxxxxx
\v	\v	ASCII tabulación vertical (VT)

\ooo	oo	Carácter con valor octal
\xhh	hh	Carácter con valor hexadecimal

Ex10- Leer datos del usuario I

```
#!/usr/bin/python

print "How old are you?"
age = raw_input()
print "How tall are you?"
height = raw_input()
print "How much do you weigh?"
weight = raw_input()

print "So, you're %r old, %r tall and %r heavy" % (age, height, weight)

```

```
How old are you?
26
How tall are you?
1.74
How much do you weigh?
65
So, you're '26' old, '1.74' tall and '65' heavy
```

Explicar la utilitat de les comes finals en les sentències print.
- En el caso de usar no usar comas al final del print, hace que la respuesta del teclado se haga en otra línia, mientras que el uso de las comas al final del print, hace que la respuesta entrada por teclado se haga en la misma línia que el mensaje del print

Si el text escrit al final no és satisfactori, provar de canviar els formats.

Ex11- Leer datos del usuario II

```
How old are you?23
How tall are you?1.74
How much do you weight?45
So, you're '23' old, '1.74' tall and '45' heavy
```

```
#!/usr/bin/python

age = raw_input("How old are you?")
height = raw_input("How tall are you?")
weight = raw_input("How much do you weight?")

print "So, you're %r old, %r tall and %r heavy" % (age, height, weight)
```

Comparar el resultat amb l'obtingut en l'activitat anterior.
- Realmente es lo mismo, però de forma abreviada

Buscar i enganxar les característiques de la funció raw_input() documentació oficial.

Fer-ho també utilitzant la comanda pydoc a la consola o terminal:
$ pydoc raw_input

```
Help on built-in function raw_input in module __builtin__:

raw_input(...)
    raw_input([prompt]) -> string

    Read a string from standard input.  The trailing newline is stripped.
    If the user hits EOF (Unix: Ctl-D, Windows: Ctl-Z+Return), raise EOFError.
    On Unix, GNU readline is used if enabled.  The prompt string, if given,
    is printed without a trailing newline before reading.
~
```

Ex12- Paso de parámetros a un script por línia de comandos

```
#!/usr/bin/python

from sys import argv
script, first, second, third = argv

print "The script is called: ", script
print "Your first variable is: ", first
print "Your second variable is: ", second
print "Your thrid variable is: ", third
```

```
aws2-08@HPi5-28:~/Escriptori$ python ex12.py first 2nd 3rd
The script is called:  ex12.py
Your first variable is:  first
Your second variable is:  2nd
Your thrid variable is:  3rd
```

Llamas al ejercicio y le pasas como parámetro «first 2nd y 3rd. Argv es un sistema de transferencia de datos por parámetro

Provar i explicar què passa si es passen menys valors dels esperats.
- Da error y te remite a la línia de argumentos, en mi caso la línia 4, donde el «argv»

Escriure aquest altre script ex13.py. En aquest cas s'importa tota la llibreria sys i s'accedeix a argv sabent que és una llista. La posició 0 conté el nom de l'script i la 1, 2 etc els corresponents paràmetres:

```
#!/usr/bin/python

import sys
params = sys.argv
nom = params[1]
print "Hola, %s" % (nom)
```

```
aws2-08@HPi5-28:~/Escriptori$ python ex13.py 0 1 2 3
Hola, 0
```

Explicar les diferències que hi ha entre l'script ex12 i l'ex13.
- En el ex12, puedes llamar al script y printas el nombre del script seleccionado. Le has pasado por línia de parámetros en el orden en el que quieres que muestre los datos introducidos
- La manera de importar el agrv y la toma de datos varía y en este caso imprime el primer dato introducido, ya que le has dicho: params[1] (primer dato introducido).