Introducció a Python
Sessió1: Primer contacte

1. Consola i scripts

Python és un llenguatge interpretat i es pot invocar la consola des de la *shell*:

Consola

Iniciem l'intèrpret de Python i anem executant les sentències d'una amb una, tal com les anem escrivint:

```
enric@pepino:~$ python
Python 2.7.3 (default, Apr 20 2012, 22:44:07)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a=1
>>> print a
1
```

Per sortir feu servir CTRL+D o quit()

Nota: La consola de Python es pot millorar afegint-hi la característica "autocompletion" amb <TAB>. Més endavant es detalla com fer-ho.

Script

Si fem un script (nom_arxiu.py) el podem executar de dues maneres:

1. Cridant el fitxer de text (script) amb l'intèrpret:

```
$ python elmeuscript.py
```

2. Executant-lo com un **shell script**: cal insertar una línia al principi del script (shebang) amb la ruta de l'intèrpret i concedir-li permisos d'execució.
Creem l'arxiu:

```
#!/usr/bin/python
i = 7
print i
```

canviem permisos i l'executem:

```
$ chmod 755 elmeuscript.py
$ ./elmeuscript.py
```

Accents i demés: codificació UTF-8 de l'arxiu

Python no ens deixarà posar accents i caràcters especials. Si volem poder treballar amb cadenes en el nostre idioma convé canviar-ho al principi de l'arxiu amb:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
print "mira què bé"
```

Si no hi posem la 2a línia, el script ens donarà un error en executar-lo.

2. Blocs per indentació, ~~res de claus {}~~

Els blocs de sentències funcionen amb **indentació** (és a dir, tabuladors/espais al principi de línia), el que és molt adequat per establir un estil clar i net de programació. Observeu com a exemple el següent fragment de codi:

```
a , b = 1, 2     # en Python podem fer assignacions de diverses variables alhora
if a == b:
   print "son iguals"
   c = True
else:
   print "son diferents"
   c = False
```

Cal tenir en compte que:

- **Els inicis de bloc els marquen els dos punts ":"** de les sentències de control (condicionals, bucles) i de les funcions.
- Tant ens fa posar un tabulador com 3 o 4 espais, la única regla és que **la indentació sigui coherent al llarg del bloc**.
- Si estem copiant exemples de tutorials i similars, i després hi afegim codi, serà convenient ajustar el tipus d'indentació de l'editor per tal que sigui coherent. En el cas de l'editor *geany* (recomanat) es pot canviar a:
  Document -> Tipus de sagnat

3. Variables dinàmiques

En Python, igual que en molts llenguatges interpretats, no cal declarar el tipus de les variables. Es pot crear una variable quan es vulgui i del tipus que es vulgui i es pot canviar al llarg de l'execució. Per exemple:

```
>>> a = 1
>>> type(a), a
(<type 'int'>, 1)
>>> a = "cosa"
>>> type(a), a
(<type 'str'>, 'cosa')
```

4. Entrada de dades per teclat

Tenim bàsicament dues funcions:

- **input()**: s'interpreta l'entrada com a sentència Python
- **raw_input()**: s'interpreta com una cadena de caràcters

En el següent exemple es veu com **raw_input()** manté l'entrada de teclat com un string (cadena de caràcters):

```
>>> xx = raw_input()
3+2                  <- entrada de teclat
>>> xx
'3+2'
```

En canvi amb **input()** s'interpreta el "3+2" i s'efectua la suma.

```
>>> xx = input()
3+2                    <- entrada de teclat
>>> xx
5
```

És interessant veure que també podriem entrar variables:

```
>>> a = 55
>>> b = input()
a                      <- entrada de teclat
>>> b
55
```

En aquest cas, el que entrem pel teclat "a" s'ha interpretat com la variable i per tant "b" valdrà el mateix que la variable "a".

Per entrar contrassenyes

Per entrar passwords sense que es mostri per pantalla, podem utilitzat la funció "getpass" del mòdul del mateix nom. Observeu com s'importen les llibreries:

```
import getpass
password = getpass.getpass("Entra la teva contrassenya: ")
```

5. Activitats de la Sessió 1

1. Strings.

Crear el fitxer de script ex1.py amb el següent contingut:

```
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.'
```

Executar-lo. S'hauria d'obtenir un resultat semblant a:

```
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.
```

Els strings es poden delimitar de dues maneres. Explicar quines són i la combinació que se'n fa a l'exercici.

2. Comentaris.

Crear el fitxer de script ex2.py amb el següent contingut:

```
# 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."
```

Executar-lo. S'hauria d'obtenir un resultat semblant a:

```
$ python ex2.py
I could have code like this.
This will run.
```

Explicar com s'escriuen comentaris en Python. Si trobem la instrucció:

```
print "Hi # there."
```

Quin efecte hi tindrà el caracter #?

3. Operacions aritmètiques i comparacions.

Crear el fitxer de script ex03.py amb el següent contingut:

```
1   print "I will now count my chickens:"
2
3   print "Hens", 25 + 30 / 6
4   print "Roosters", 100 - 25 * 3 % 4
5
6   print "Now I will count the eggs:"
7
8   print 3 + 2 + 1 - 5 + 4 % 2 - 1 / 4 + 6
9
10  print "Is it true that 3 + 2 < 5 - 7?"
11
12  print 3 + 2 < 5 - 7
13
14  print "What is 3 + 2?", 3 + 2
15  print "What is 5 - 7?", 5 - 7
16
17  print "Oh, that's why it's False."
18
19  print "How about some more."
20
21  print "Is it greater?", 5 > -2
22  print "Is it greater or equal?", 5 >= -2
23  print "Is it less or equal?", 5 <= -2
```

Executar-lo. S'hauria d'obtenir un resultat semblant a:

```
$ python 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
```

Els operadors aritmètics són:

+	suma
-	resta
*	multiplicació
/	divisió amb nombres reals
//	división amb nombres enters
%	residu de la divisió entera (mòdul)
**	exponenciació

La prioritat d'execució és: Parèntesis, Exponents, Multiplicació, Divisió, Suma i Resta.

Els operadors relacionals són:

==	Igualtat
!=	Desigualtat
<	més petit que
<=	més petit o igual que
>	més gran que
>=	més gran o igual que

El resultat d'una comparació és un valor booleà, que pot ser True o False.

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

4. Variables i noms. Normes d'escriptura.

Crear el fitxer de script ex4.py amb el següent contingut:

```
cars = 100
space_in_a_car = 4.0
drivers = 30
passengers = 90
cars_not_driven = cars - drivers
cars_driven = drivers
carpool_capacity = cars_driven * space_in_a_car
average_passengers_per_car = passengers / cars_driven


print "There are", cars, "cars available."
print "There are only", drivers, "drivers available."
print "There will be", cars_not_driven, "empty cars today."
print "We can transport", carpool_capacity, "people today."
print "We have", passengers, "to carpool today."
print "We need to put about", average_passengers_per_car, "in each car."
```

Executar-lo. S'hauria d'obtenir un resultat semblant a:

```
$ python ex4.py
There are 100 cars available.
There are only 30 drivers available.
There will 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.
```

Els noms de les variables poden contenir lletres, números i el caràcter _ (guió baix o underscore) però no poden començar amb un número.
Quan el nom d'una variable (o altre objecte del programa) es composa de diverses paraules, s'acostuma a separar-les amb un guió baix.
Les constants en Python són simplement variables que no canvien el seu valor. Es recomana escriure el seu nom en majúscules.
Es recomana escriure espais entre els operadors i els operands.
Els nombres reals s'escriuen amb punt decimal.

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

5. Variables i escriptura amb format.

Crear el fitxer de script ex5.py amb el següent contingut:

```
1   my_name = 'Zed A. Shaw'
2   my_age = 35 # not a lie
3   my_height = 74 # inches
4   my_weight = 180 # lbs
5   my_eyes = 'Blue'
6   my_teeth = 'White'
7   my_hair = 'Brown'
8
9   print "Let's talk about %s." % my_name
10  print "He's %d inches tall." % my_height
11  print "He's %d pounds heavy." % my_weight
12  print "Actually that's not too heavy."
13  print "He's got %s eyes and %s hair." % (my_eyes, my_hair)
14  print "His teeth are usually %s depending on the coffee." % my_teeth
15
16  # this line is tricky, try to get it exactly right
17  print "If I add %d, %d, and %d I get %d." % (
18      my_age, my_height, my_weight, my_age + my_height + my_weight)
```

Executar-lo. S'hauria d'obtenir un resultat semblant a:

```
$ 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

6. Combinació d'strings i formats.

Crear el fitxer de script ex6.py amb el següent contingut:

```
1   x = "There are %d types of people." % 10
2   binary = "binary"
3   do_not = "don't"
4   y = "Those who know %s and those who %s." % (binary, do_not)
5
6   print x
7   print y
8
9   print "I said: %r." % x
10  print "I also said: '%s'." % y
11
12  hilarious = False
13  joke_evaluation = "Isn't that joke so funny?! %r"
14
15  print joke_evaluation % hilarious
16
17  w = "This is the left side of..."
18  e = "a string with a right side."
19
20  print w + e
```

Executar-lo. S'hauria d'obtenir un resultat semblant a:

```
$ 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) ...
>>>
```

7. Impressió d'strings. Concatenació.

Crear el fitxer de script ex7.py amb el següent contingut:

```
1   print "Mary had a little lamb."
2   print "Its fleece was white as %s." % 'snow'
3   print "And everywhere that Mary went."
4   print "." * 10  # what'd that do?
5
6   end1 = "C"
7   end2 = "h"
8   end3 = "e"
9   end4 = "e"
10  end5 = "s"
11  end6 = "e"
12  end7 = "B"
13  end8 = "u"
14  end9 = "r"
15  end10 = "g"
16  end11 = "e"
17  end12 = "r"
18
19  # watch that comma at the end.  try removing it to see what happens
20  print end1 + end2 + end3 + end4 + end5 + end6,
21  print end7 + end8 + end9 + end10 + end11 + end12
```

Executar-lo. S'hauria d'obtenir un resultat semblant a:

```
$ python ex7.py
Mary had a little lamb.
Its fleece was white as snow.
And everywhere that Mary went.
..........
Cheese Burger
```

S'aconsella no escriure línies de codi amb més de 80 caràcters.

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

8. Més impressió d'strings.

Crear el fitxer de script ex8.py amb el següent contingut:

```
# 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.
"""
```

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

9. Caràcters escapats amb la contrabarra o backslash. Documentació.

Crear el fitxer de script ex9.py amb el següent contingut:

```
1   tabby_cat = "\tI'm tabbed in."
2   persian_cat = "I'm split\non a line."
3   backslash_cat = "I'm \\ a \\ cat."
4
5   fat_cat = """
6   I'll do a list:
7   \t* Cat food
8   \t* Fishies
9   \t* Catnip\n\t* Grass
10  """
11
12  print tabby_cat
13  print persian_cat
14  print backslash_cat
15  print fat_cat
```

Executar-lo. Explicar quin és l'efecte de la contrabarra en cadascun dels strings on apareix.

Llistar les "seqüències d'escape" que hi ha a Python. Buscar a la documentació oficial: **docs.python.org**.

10. Llegir dades de l'usuari.

Crear el fitxer de script ex10.py amb el següent contingut:

```
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)
```

Executar-lo.
Explicar la utilitat de les comes finals en les sentències print.
Si el text escrit al final no és satisfactori, provar de canviar els formats.

11. Llegir dades de l'usuari (2). Documentació.

Crear el fitxer de script ex11.py amb el següent contingut:

```
1    age = raw_input("How old are you? ")
2    height = raw_input("How tall are you? ")
3    weight = raw_input("How much do you weigh? ")
4
5    print "So, you're %r old, %r tall and %r heavy." % (
6        age, height, weight)
```

Executar-lo.
Comparar el resultat amb l'obtingut en l'activitat anterior.
Buscar i enganxar les característiques de la funció raw_input() a la documentació oficial.
Fer-ho també utilitzant la comanda pydoc a la consola o terminal:

```
$ pydoc raw_input
```

12. Pas de paràmetres a un script per línia de comandes.

Crear el fitxer de script ex13.py amb el següent contingut:

```
1  from sys import argv
2
3  script, first, second, third = argv
4
5  print "The script is called:", script
6  print "Your first variable is:", first
7  print "Your second variable is:", second
8  print "Your third variable is:", third
```

Importem el mòdul argv per poder utilitzar una llista de valors d'entrada a l'script. En el codi es pot veure que s'esperen quatre valors. Per això, a la crida, s'han de proporcionar aquests valors.

Un exemple de crida a l'script i el seu resultat podria ser aquest:

```
$ python ex13.py first 2nd 3rd
The script is called: ex13.py
Your first variable is: first
Your second variable is: 2nd
Your third variable is: 3rd
```

Provar i explicar què passa si es passen menys valors dels esperats.

Escriure aquest altre script ex13b.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:

```
import sys

params = sys.argv # argv és una llista amb tots els arguments que passem
nom = params[1]
print "Hola, %s." % (nom)
```

Cridarlo de la manera que correspongui.
Explicar les diferències que hi ha entre l'script ex13 i l'ex13b.