Conceptes bàsics de programació i Python, orientat a 1r de Batxillerat

Enric X. Martín Rull
Manel Velasco García
Juny 2022

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Estructura del curs:

PART 1. El llenguatge Python

PART 2. Conceptes d'algorísmia

PART 3. Utilització de llibreries

PART 4. Exercicis tipus

3. Llibreries

PART 3. Utilització de llibreries

1. Utilització de llibreries
2. Creació de llibreries
3. Algunes llibreries d'interès

3.1. Utilització de llibreries

Instal·lació

Per utilitzar una llibreria de Python cal, en primer lloc, que la tinguem instalada. Si instal·lem un paquet estàndard gran com Anaconda ja tindrem disponibles una gran varietat de llibreries. Sinó haurem de cercar "a mà" com es fa la instal·lació.

Per saber quins mòduls o llibreries tenim instal·lats, podem fer des d'una consola de Python:

```
>> help ('modules')
```

I se'ns mostrarà la llista de mòduls instal·lats al nostre sistema.

També des de la consola, o al programa podem escriure:

```
>> import nom_llibreria
```

I comprovar la seva disponibilitat.

3.1. Utilització de llibreries

Instal·lació

La forma estàndard d'instal·lar llibreries a Python és utilitzant **pip**.

Pip cercarà la versió més actualitzada, resoldrà si ho ha de fer des del codi font (*sdist: source distribution*) o de versions precompilades (*wheels*) i ho posarà en el lloc correcte perquè la tinguem disponible.

Normalment cada llibreria tindrà una pàgina web o ajuda indicant el mètode d'instal·lació preferit.

El procés general de funcionament de llibreries és aquí:

https://packaging.python.org/en/latest/tutorials/installing-packages/

3.1. Utilització de llibreries

Utilització

Es farà servir la comanda **import**. A continuació de *import* posarem el nom de la llibreria. Programant en Python veurem que hi ha unes formes habituals de fer servir la importació de llibreries. Anem a veure les quatre més habituals:

import *llibreria*

Estem portant tota la *llibreria* al nostre programa. Aquest serà el nom (*llibreria*) per fer-la servir, invocant les seves definicions o les seves funcions.
Exemple:

```
import numpy      # importem la llibreria numpy per tenir el nombre pi

radi = float ( input (" Entra el radi: ")
perimetre = radi * 2.0 * numpy.pi          # invoquem pi de numpy
print (perimetre)
```

3.1. Utilització de llibreries

Utilització

import llibreria as lib

Portem la llibreria amb un nom diferent, normalment més curt, per estalviar escriptura.

Exemple, el mateix d'abans, anomenem a numpy np:

```
import numpy as np        # importem la llibreria numpy amb nom np

radi = float ( input (" Entra el radi: ")
perimetre = radi * 2.0 * np.pi          # invoquem pi de np (numpy)
print (perimetre)
```

3.1. Utilització de llibreries

Utilització

from llibreria import part

Portem d'una llibreria una part, que pot ser una funció, grup de funcions. Va bé per tenir el codi més clar i no inundar l'espai de noms del nostre programa (si ho importem tot). A canvi si després ens calen més mòduls, caldrà afegir més línies.

Exemple, de la llibreria matplotlib (que és molt extensa) volem usar només les funcions de dibuixar:

from matplotlib import pyplot

from llibreria import part as nom

Importem una part de la llibreria i li donem un nom que ens sigui còmode d'utilitzar.

3.1. Utilització de llibreries

Utilització

Exemple:

```
from matplotlib import pyplot as plt
                    # anomenem plt a les funcions de dibuix
x = [ -3, -2, -1, 0, 1, 2, 3 ]
y = [  5,  3,  4,  6, 3, 3, 2 ]

plt.plot( x, y, 'r' )   # definim punts de l'eix x i y, color vermell
plt.show( )             # pintem
```

3.2. Creació de llibreries

Posarem el codi que volem definir com a llibreria en un fitxer *nom.py* que importarem des d'altres programes.

El nom no pot coincidir amb altres llibreries que volguem utilitzar.

El fitxer de la llibreria ha d'estar accessible (a la mateixa carpeta o una carpeta general) perquè import el pugui trobar.

TADs.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Jun 11 16:10:55 2022
@author: Profe Python
"""

class llista ():

    def __init__(self):
        self.dades=[]
        self.actual=0
        self.num=0
    def seleccionar (self, nombre):
        self.actual = nombre
    def afegir(self, element):
        self.dades.insert(self.actual,element)
        self.num=self.num+1
    def extreure(self):
        self.num=self.num-1
        return self.dades.pop(self.actual)
    def consultar(self):
        return (self.dades[self.actual])
    def buida(self):
        return len(self.dades)==0
    def quants(self):
        return self.num
```

3.2. Creació de llibreries

El programa que usa la llibreria fa servir:

import nom (sense .py)

ProvaTADs.py

```
1   #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3   """
4   Created on Wed Jun 22 14:41:19 2022
5
6   @author: Profe Python
7   """
8   import TADs
9
10
11  l = TADs.llista()
12  l.afegir("Joanna")
13  l.seleccionar(1)
14  l.afegir("Maria")
15  l.seleccionar(2)
16  l.afegir("Moha")
17  l.seleccionar(1)
18  l.afegir("Kevin")
19
20  i=0
21  while i<l.quants():
22      l.seleccionar(i)
23      print (l.consultar())
24      i=i+1
25
26
27
28
```

3.3. Algunes llibreries d'interès

random

La llibreria *random* ens ofereix funcions per generar nombres aleatoris, que seran útils per estadística, inicialització de jocs, etc.

Exemple.

```
import random

valor = random.randint (1, 100)        # generem un valor de 1 a 100

print (" El valor és: ", valor )
```

3.3. Algunes llibreries d'interès

numpy

La llibreria *numpy* ens ofereix multitud d'operacions matemàtiques optimitzades perquè corrin el més ràpid possible en el processador. Per garantir un ús òptim dels recursos hardware i una bona organització a memòria, ens ofereix també estructures de dades (bàsicament *arrays*) on col·locar la nostra informació.

Exemple 1. Calculem el sinus d'una sèrie de 0 a 2*PI

```
import numpy as np
x = np.arange( 0.0, 2.0*np.pi, 0.01)        # creem un array de valors de 0 a 2 pi
                                            # amb pas 0.01
y = np.sin ( x )                            # calcula amb una sola crida el sinus de
                                            # tots els elements de l'array

print (x)
print (y)
```

3.3. Algunes llibreries d'interès

numpy

Exemple 2. Treballem amb matrius

```
import numpy as np

m = [ [1.0,1.3,2.9] , [0.0,2.4,1.5] , [1.2,1.0,0.5] ]
M1= np.array(m)
print (M1)          # declarem les files d'una matriu i les passem a un
                    # array de numpy i el mostrem per pantalla
ID =np.identity(3)
print (ID)          # identity(n) crea una matriu identitat de la dimensió n

M2= np.linalg.inv(M1)  # provem de fer inversa de M1 a M2 amb les funcions
print (M2)          # de linear algebra de numpy

M3=np.dot(M1,M2)     # fem el producte de matrius per provar
print (M3)
```

3.3. Algunes llibreries d'interès

matplotlib

matplotlib és una llibreria multiplataforma que associada amb *numpy* ens permet visualitzar dades i fer tota mena de gràfics matemàtics. Es va crear com una alternativa lliure i gratuïta a eines com Matlab.

Exemple 1. Representem el sinus calculat amb numpy

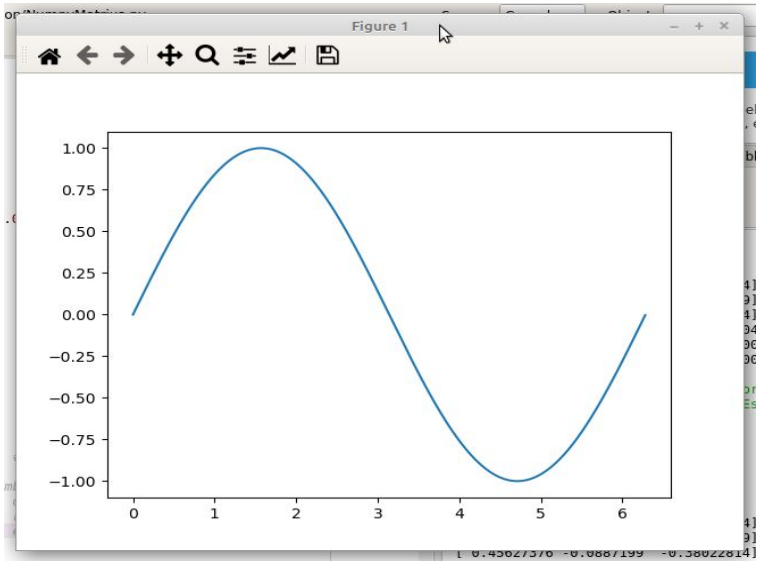
```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange( 0.0, 2.0*np.pi, 0.01)      # creem un array de valors de 0 a 2 pi
                                          # amb pas 0.01
y = np.sin ( x )                          # calcula amb una sola crida el sinus de
                                          # tots els elements de l'array
plt.plot( x, y)                           # presentarem y respecte a les x
plt.show( )                               # fem aparèixer el dibuix
```

3.3. Algunes llibreries d'interès

matplotlib

Exemple 1. Representem el sinus calculat amb numpy

Obtenim:



3.3. Algunes llibreries d'interès

tkinter

Fins ara hem estat veient exemples on el codi feia interacció amb l'usuari a través de la consola de Python. Allà mitjançant les funcions *print* i *input* podem imprimir i demanar dades a l'usuari. La llibreria *tkinter* permet generar programes amb interfícies d'usuari estàndard com botons, caixes de text, etiquetes, etc.

El següent exemple mostra com crear una finestra, posar-hi un botó, una capsa d'entrada d'informació, una capsa per mostrar informació i com associar una funció a un event de que s'ha premut el botó.

Exemple amb diversos elements de tkinter:

3.3. Algunes llibreries d'interès

tkinter

```
import tkinter as tk

def boto_pres (event):          # funció associada al botó. print a pantalla el
                                # nom i al textbox text saluda i compta lletres
    print ("Hola", entry.get() )
    text.insert ("1.0", "Hola "+ entry.get( ) + "\n")
    text.insert ("2.0","El nom té "+ str (len (entry.get() ) ) + " lletres\n")


w = tk.Tk( )                    # declarem una finestra de tkinter w i li
w.title ( "Prova tkinter" )     # donem un nom i una mida
w.geometry ( "300x450" )
```

3.3. Algunes llibreries d'interès

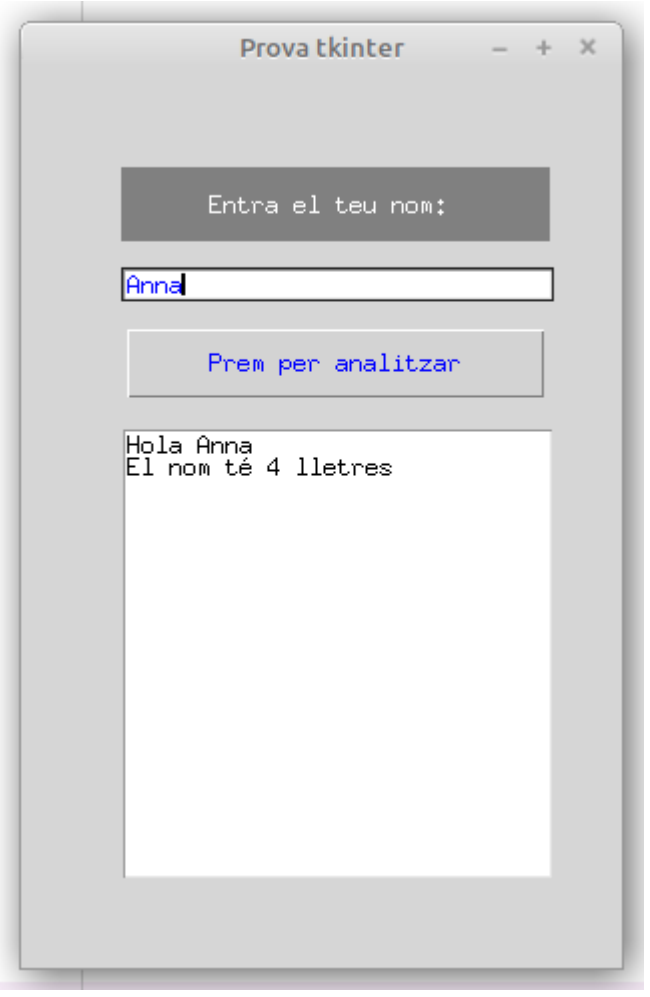
tkinter

```
etiq = tk.Label ( text="Entra el teu nom: ", fg="white", bg="grey",width=30,height=3 )
etiq.place( x=50,y=50 )

entry = tk.Entry ( fg="blue", bg="white", width=30)
entry.place(x=50,y=100)

boto = tk.Button ( text= "Prem per analitzar",width=26, height=2, bg="lightgrey", fg="blue")
boto.bind( "<Button-1>", boto_pres)# associem una funció al botó
boto.place( x=52, y=130)

text = tk.Text ( fg= "black", bg="white", width=30, height=20)
text.place(x=50,y=180)

w.mainloop()          # mainloop activa tot el mecanisme tkinter
```

3.3. Algunes llibreries d'interès

tkinter

Resultat:



3.3. Algunes llibreries d'interès

pygame

La llibreria pygame ens dóna recursos per la gestió de petits jocs. Pinta imatges de fons, pinta objectes en primer pla, controla el temps d'actualització de la pantalla, detecta les tecles premudes, etc.

L'exemple plantejat a continuació mou un avió (imatge .png amb fons transparent) sobre un mapa pregenerat de 4000x2000 píxels.

En cas de que generem una gran quantitat de dibuixos i figures, serà una bona idea crear una carpeta de recursos del joc per tenir-ho tot ordenat.

Com en el cas de tkinter, passarem el control del programa al mòdul pygame i atendrem events com la pulsació de les tecles.

3.3. Algunes llibreries d'interès

pygame

```
import pygame
import pygame.key
from pygame.locals import *

pygame.init()                                   # Creem la finestra pygame de 800x600
clock    = pygame.time.Clock()                  # amb títol i inicialitzem els timers
screen   = pygame.display.set_mode((800,600))
pygame.display.set_caption("Prova Joc")
                                                # Carreguem imatge de fons i l'avió
img_fons = pygame.image.load("Mapa.png")
img_avio = pygame.image.load("avio.png")
img_f_sc = pygame.transform.scale(img_avio,(100,100))
                                                # Avió escalat a 100x100 (opcional)
x=-957
y=-439                                          # Posició original (mourem el fons)
vx=0
vy=0                                            # Velocitat inicial aturat
```

3.3. Algunes llibreries d'interès

pygame

```
done = False
while not done:              # mentre no acabem el joc

for event in pygame.event.get():      # analitzem events: sortir (quit) o tecla
        if event.type == pygame.QUIT:
                done = True

        if event.type == pygame.KEYDOWN:

                if event.key==K_UP:       # segons les tecles donem més o menys
                        vy=vy+1           # velocitat a l'avió amb vx i vy
                if event.key==K_LEFT:
                        vx=vx+1
                if event.key==K_DOWN:
                        vy=vy-1
                if event.key==K_RIGHT:
                        vx=vx-1
```

3.3. Algunes llibreries d'interès

pygame

```
        x=x+vx                      # Actualitzem x i y amb la velocitat
        y=y+vy
        screen.blit(img_fons, (x,y))          # pintem imatge de fons
        screen.blit(img_f_sc, (360,300))      # pintem avió al centre
        pygame.display.flip()             # actualitzem la pantalla amb la imatge
        clock.tick(40)              # nova i esperem 40 ms

pygame.quit()                       # a l'acabar, sortim de pygame
```

Obtenint...

3.3. Algunes llibreries d’interès

pygame