02 - Introducció a la programació

September 21, 2015

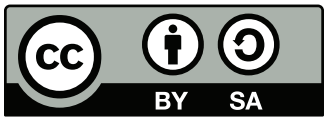

Figure 1: BY-SA

Autors : *Sonia Estradé*
José M. Gómez
Ricardo Graciani
Frank Güell
Manuel López
Xavier Luri
Josep Sabater

1 Introducció a la programació

La programació és el procés d'escriure instruccions per tal de crear software (programes) per un usuari final. Per sobre de tot, és una activitat creativa que preten sol·lucionar, mitjançant software, un determinat problema.

Actualment, la programació forma part de les habilitats bàsiques que ha de proporcionar qualsevol grau d'enginyeria o de base científica.

Les instruccions de un programa s'escriuen fent servir un **llenguatge de programació**. Per tant, un **llenguatge de programació** és el llenguatge formal amb el que comuniquem a una màquina les instruccions que ha d'executar (programes o algorismes). Un **llenguatge de programació** consisteix en una sèrie de regles sintàctiques (forma) i semàntiques (significat).

Tots els llenguatges de programació tenen elements bàsics per a la descripció de les dades i dels processos o transformacions que se'ls hi poden aplicar (exemples d'aquests processos són la suma de dos números o la sel·lecció d'un element concret d'una col·lecció). Aquests elements bàsics es defineixen per una sèrie de regles sintàctiques i semàntiques, que descriuen la seva estructura i funcionalitat respectivament.

La sintàxi es refereix a la forma en que un llenguatge està escrit. La majoria dels llenguatges de programació són textuals. Això vol dir que fan servir seqüències de text, paraules, números i puntuació com la majoria de llenguatges naturals que existeixen. Hi ha, però, d'altres llenguatges de programació que treballen de forma gràfica, fent servir relacions visuals entre símbols per definir i crear el programa final. La sintàxi d'un llenguatge descriu les possibles combinacions de símbols que formen un programa sintàcticament correcte.

La semàntica es refereix al contingut, al significat. Existeixen una sèrie de restriccions en l'estructura de un programa vàlid. En el cas dels llenguatges compilats, la semàntica del llenguatge inclou aquelles regles que s'han de cumplir en el procés de compilació del programa. Per exemple, necessitem comprovar que tot identificador ha estat declarat abans d'utilitzar-se (en aquells llenguatges que requereixen de la declaració d'aquests identificadors). Veiem el següent exemple:

```
print (x)
x = 3
```

i

```
x = 3
print (x)
```

Tot i que tots dos exemples són sintàcticament correctes, són semànticament diferents ja que s'ha de tenir en compte l'ordre d'execució. En funció de l'ordre en que fem servir les instruccions d'un programa el significat d'aquest pot variar. Això té a veure amb l'ordre d'execució, la precedència dels operadors, etc.

Depenent del nivell d'abstracció que el llenguatge presenta respete als detalls físics de l'ordinador que s'està fent servir (tipus de processador, memòria, accés a registres,...), els llenguatges es classifiquen en **llenguatges d'alt nivell** i **llenguatges de baix nivell**. Els llenguatges de més baix nivell són aquells que treballen en llenguatge màquina o ensamblador, atacant dirèctament a registres generals i específics del processador, adreçant la memòria, etc.

1.1 Llenguatges de programació d'alt nivell

Tal i com hem comentat en l'apartat anterior, els llenguatges de programació d'alt nivell són aquells que presenten un alt nivell d'abstracció respecte als detalls del hardware del PC on estem treballant. Fan servir elements del llenguatge natural (anglès) i són normalment més senzills d'utilitzar, amagant al programador elements importants dels ordinadors com és el cas de la gestió de la memòria.

En la dècada dels 60, els llenguatges de programació d'alt nivell feien servir els compiladors per "traduïr" aquests llenguatges formals en seqüències de codi màquina. Aquests llenguatges eren anomenats **autocodes**. Exemples d'autocodes són COBOL i Fortram. Fortram va ser el primer llenguatge d'alt nivell àmpliament utilitzat (de fet, encara hi ha gent avui dia que el fa servir...) proporcionant un veritable sistema de desenvolupament independent de la màquina, en aquella època, als primers sistemes IBM. Un dels grans avantatges d'aquests programes era que **el mateix programa es podia fer servir en diferents màquines**.

En lloc de treballar amb registres, adreços de memòria i call stacks, els llenguatges d'alt nivell feien servir variables, arrays, objectes, complexes expressions aritmètiques o booleanes, subrutines i funcions, bucles, fils, locks i d'altres conceptes de la ciència de la programació abstracta, que es focalitzava en la (re)usabilitat del codi en lloc de buscar la eficiència del programa.

Mentre que els programes d'alt nivell tracten de fer més senzilla la programació, els llenguatges de baix nivell sovint generen codi més eficient (en quant a velocitat d'execució i espai en memòria). Ara bé, amb l'evolució de les modernes i complexes arquitectures de processadors actuals, el desenvolupament de compiladors ha evolucionat en concordància a aquestes arquitectures, generant codi comparable en eficiència i mida a aquell que poden generar la majoria de programadors de baix nivell.

Depenent del mode d'execució, els llenguatges de programació d'alt nivell es poden subdividir en: **compilats** i **interpretats**. En el primer cas, el codi és transformat en un fitxer executable abans de fer-lo còrrer. Això es fa produïnt directament "codi màquina" a partir del programa d'alt nivell, o, més recentment, produïnt una representació optimitzada intermitja com és el "byte code", que es pot guardar per futurs processos sense tenir que accedir al codi original. Aquesta representació intermitja requereix ser posteriorment compilada o interpretada per tal de poder-se executar.

1.2 Llenguatges interpretats

Quan el llenguatge és interpretat, la seva sintàxi és llegida i executada directament, sense necessitat de ser compilat. Hi ha un programa anomenat **intèrpret** que llegeix cada sentència del programa, i seguint la seva seqüència natural, l'executa. Un híbrid d'un intèrpret i un compilador compilarà la sentència en codi màquina i l'executarà. Un cop executat, el codi màquina es descartarà, per interpretar la següent sentència de programa.

El pas de compilació usualment introdueix una comprobació extensiva de la sintàxis de tot el codi, asegurant amb un alt nivell de precissió, la consistència del codi abans de produïr l'executable, al mateix

temps s'introdueixen procediments automàtics que optimitzen l'ús de recursos. Quan el codi és intepretat, aquesta comprobació només es pot fer en aquella sentència que s'està executant.

Nota: Els llenguatges no són estrictament o "interpretats" o "compilats", més aviat implementacions d'aquests llenguatges fan servir la compilació o l'interpretació.

1.3 Python, IPython and Notebook

Python és un llenguatge de programació d'alt nivell, multiplataforma, de propòsit general i interpretat. La seva filosofia rau en la re-utilització de codi, amb una sintàxi que permet als programadors expressar conceptes en menys línies de codi que les que es farien servir en altres llenguatges com C++ o Java.

El intèrprets de Python estan disponibles per l'instal·lació en la majoria de sistemes operatius, permetent l'ús de Python en una àmplia varietat de sistemes.

En lloc de introduïr tota la funcionalitat requerida en el nucli del llenguatge, Python incorpora un conjunt de llibreries que poden ser importades en cas de ser necessàries. Python pot ser incrustat en aplicacions ja existents que necessiten una interfície programable. Aquest disseny d'un nucli d'instruccions relativament petit amb una gran quantitat de llibreries estàndars i un senzill i extensible intèrpret forma part de la filosofia de Python des dels seus orígens.

Es tracta de un llenguatge de programació multiparadigma, ja que permet la programació orientada a objectes (OOP), la programació imperativa i la programació funcional. Com ja s'ha comentat és un llenguatge interpretat, que fa servir tipus de dades dinàmic i és multiplataforma. Una important característica de Python és la resolució dinàmica de noms, associant noms de variables i funcions durant l'execució del programa.

Un neologisme prou comú en la comunitat de Python és ***pythonic***. Diem que un codi és pythonic perque fa servir correctament el llenguatge, és senzill de llegir, és reutilitzable i segueix la filosofia de Python. Al contrari, quan veiem un codi que és difícil de seguir o llegir, o sembla una complicada traducció d'un altre llenguatge de programació i que és difícilment reutilitzable, diem que és ***unpythonic***.

Python pot ser empaquetat en programes executables per alguns dels sistemes operatius més utilitzats fent servir eines d'altres distribuidors o fabricants (i per tant de pagament), permetent la distribució de software basat en Python per l'ús d'aquests programes sense requerir la instal·lació d'un intèrpret de Python.

1.3.1 IPython

IPyhton (python interactivo) és un projecte de software lliure que proporciona:

- Una consola de python millorada
- Un model de comunicació de dos processos desacoblat, que permet a diversos clients connectar-se al kernel de computació, fent servir l'eina notebook
- Una arquitectura que permet la computació interactiva paral·lela

IPython té característiques com: terminació de sentències fent servir el tabulador, introspecció d'objectes, accés a la consola de sistema, historial de comandaments de recuperació entre sessions, i el seu propi sistema de comandes per afegir funcionalitat quan es treballa de forma interactiva. Proporciona també un entorn de treball força eficient pel desenvolupament de codi Python i per sol·lucionar problemes fent servir objectes de Python.

Les quatre comandes més útils de IPython es mostren a l'inici de l'intèrpret:

```
?         -> Introducció o visió en conjunt de les característiques de IPython
%quickref -> Quick reference
help      -> Sistema de ajuda de Python
object?   -> Característiques d'un determinat objecte. Podem fer servir 'object??' per més informació
```

Altres característiques interessants són:

- Terminació de sentències fent servir el tabulador, especialment per atributs, és interessant per explorar l'estructura i característiques de qualsevol objecte, simplement escrivint `object_name.<TAB>`. També funciona amb noms de fitxers i directoris.

- Escrivint `object_name?` imprimirà les característiques i detalls sobre qualsevol objecte.
- Enmagatzema les comandes introduïdes i el resultat que generen. Es pot anar fàcilment a les comandes prèvies simplement clicant les fletxes adalt i abaix, o podem accedir a l'històric de forma més sofisticada.
- Permet correr qualsevol comanda al terminal de sistema, simplement amb el prefix `!`

1.3.2 IPython Notebook

IPython Notebook millora la proposta de programació interactiva proporcionada per la consola o terminal, explicat en el punt anterior, introduïnt un entorn de programació més agradable fent servir el propi navegador. Aquesta aplicació és adecuada per treballar el procés de software global: desenvolupament, documentació i codi executable, així com un entorn per preparar i documentar els resultats.

Les principals característiques són:

- Editor de codi al navegador, amb identificació de comandes de forma automàtica, identació, terminació de sentències fent servir el tabulador, i introspecció
- Execució de codi des del navegador, amb el resultat de l'execució annexada al codi que l'ha generat
- Mostra el resultat de l'execució del programa utilitzant representació enriquida, com és HTML, LaTeX, PNG, SVG, etc
- Editor integrat al navegador per text enriquit fent servir l'opció **Markdown**, que proporciona comentaris de text pel codi, no limitat a text pla.
- L'opció d'incloure equacions matemàtiques i afegir notació matemàtica a les cel·les definides com markdown, utilitzant LaTeX, proporcionat de forma nativa per MathJax.

1.4 El primer programa

Normalment el primer codi que es genera quan s'apren un llenguatge de programació consisteix en imprimir una sentència. El codi bàsicament imprimeix una cadena de text, fent servir la consola o sortida estàndar. La sentència que imprimim és "Hola Mon!".

```
In [1]: print("Hola Mon!")
```

```
Hola Mon!
```

Nota <TAB>+<Intro> executa el codi que hi ha a la cel·la

1.4.1 Comentant el codi

Els comentaris són essencials per ajudar a altres usuaris (o a nosaltres mateixos passat un temps) a entendre el codi. Farem servir tres formes de comentar el codi

- Inline comment (#)
- Multi-line comment(""")
- Markdown

```
In [2]: # Aquest és el primer programa Hola Mon
        print("Hola Mon!")
        """ Les comes defineixen un
            comentari de múltiples
            línies i es pot fer servir
            per comentaris més llargs.
            Ara imprimim "Adeu!".
        """
        print("Adeu!")
```

```
Hola Mon!
Adeu!
```

1.5 Markdown

Markdown és una eina que permet la conversió de text a HTML. Markdown permet escriure de forma senzilla text pla i convertir-lo a un format estructurat HTML.

El principal avantatge de fer servir Markdown és que la lectura i escriptura del seu codi font és molt senzill. La idea és que un document amb format Markdown es pot presentar tal com és, com text pla, sense semblar que ha estat marcat amb etiquetes o instruccions de format.

La millor manera de veure com funciona Markdown i la seva sintaxi és símplement donar una ullada a un document amb format Markdown.

IPython Notebook permet combinar cel·les de codi (que fan servir l'intèrpret de IPython) amb cel·les de text (fent servir l'intèrpret de Markdown).

1.5.1 Negreta i Itàliques

Es poden generar a partir de la següent sintaxi:

- `**Negreta**`: **Negreta** o `__Negreta__`: **Negreta**
- `*Itàlica*`: *Itàlica* o `_Itàlica_`: *Itàlica*

1.5.2 Enllaços externs

Es poden crear enllaços a pàgines web:

```
[Universitat de Barcelona](http://www.ub.edu)
```

Universitat de Barcelona
i també incloure imatges

```
![UB](http://www.ub.edu/web/ub/galeries/imatges/logo_home_nou.png)
```

Figure 2: UB

1.5.3 Estructura de text

El text pot ser estructurat en seccions i subseccions

```
#Títol de primer nivell
Text de la secció principal


##Títol de segon nivell
Text de la subsecció


###Títol de la sub-subsecció
Text de la sub-subsecció
```

2 Títol de primer nivell

Text de la secció principal

2.1 Títol de segon nivell

Text de la subsecció

2.1.1 Títol de la sub-subsecció

Text de la sub-subsecció

2.1.2 Format de citació

Per fer cites es pot usar el format de citació: > Això és una cita literal: Això és un exemple o com a text d'amplada fixa (freqüentment usat per codi):

```
indicar
print("x + y = " + str(x + ))
```

2.1.3 Llistes

Es poden crear llistes iniciant el texte amb `-` o amb `*` :

```
* Item a
- Item a
```

- Item a
- Item a

Aquesta llista conté una sub-llista

```
 * sub-item 1
 * sub-item 2
```

- sub-item 1
- sub-item 2
- Item b
- Item c

I llistes numerades:

1. Item 1
2. Item 2
3. Item 3

2.1.4 Equacions matemàtiques

També es poden escriure fórmules si fem servir

```
$\vec{F} = -G \frac{mM}{R^2}\hat{r}$‘‘‘
obtenim
$\vec{F} = -G \frac{mM}{R^2}\hat{r}$

Fent servir ‘‘‘$$ equació$$‘‘‘ introdueix un canvi de línia

Per exemple
```

executem ara això

$$\vec{F} = -G\frac{mM}{R^2}\hat{r}$$

“‘ executem ara això

$$\vec{F} = -G\frac{mM}{R^2}\hat{r}$$