Bloc D: Programació en Python

pre-alpha 10/11/22 by XFF

Bibliografia

- Sweigart, A. (2015). *Automate the boring stuff with python: practical programming for total beginners.* San Francisco, No Starch Press.
- Matthes, E. (2015) *Python Crash Course: A Hands-On, Project-Based Introduction to Programming* (1st. ed.). No Starch Press, USA.
- Zelle, J. (2009) *Python Programming: An Introduction to Computer Science.* (2nd ed.) Franklin, Beedle & Associates INC. USA
- van Rossum, G. et al. (2018) *Python Tutorial*
- Guru99 Python tutorial (*https://www.guru99.com/python-tutorials.html*)
- W3 Schools Python tutorial (https://www.w3schools.com/python/default.asp)

Ampliació i reptes de programació pels més agosarats:

- https://projecteuler.net/
- https://stackoverflow.com/questions/2573135/python-progression-path-from-apprentice-to-guru

Metodologia de classe

Classes de 2h dividides en dues parts:

- Primera part: Part extensiva de teoria on es presenta el material de manera teorico-pràctica. i.e: mentre es presenten ints, llistes, for loops, etc s'ensenya al IDE o a VSCode el seu comportament. L'alumne s'espera que provi els exemples de classe o que prengui apunts de la teoria. La classe està pensada per a fer servir un híbrid entre diapositives/pissarra i demostracions en viu amb IDE.
- Sergona part: Part de treball en grup amb exercicis guiats a classe. Es pot dedicar part de la segona hora a explicar nous conceptes però aquests sempre aniran acompanyats d'exercicis. Al final de la classe se'ls donarà el projecte que tenen que fer per a la setmana vinent (a entregar online el dia que convingui)

Filosofia del bloc D

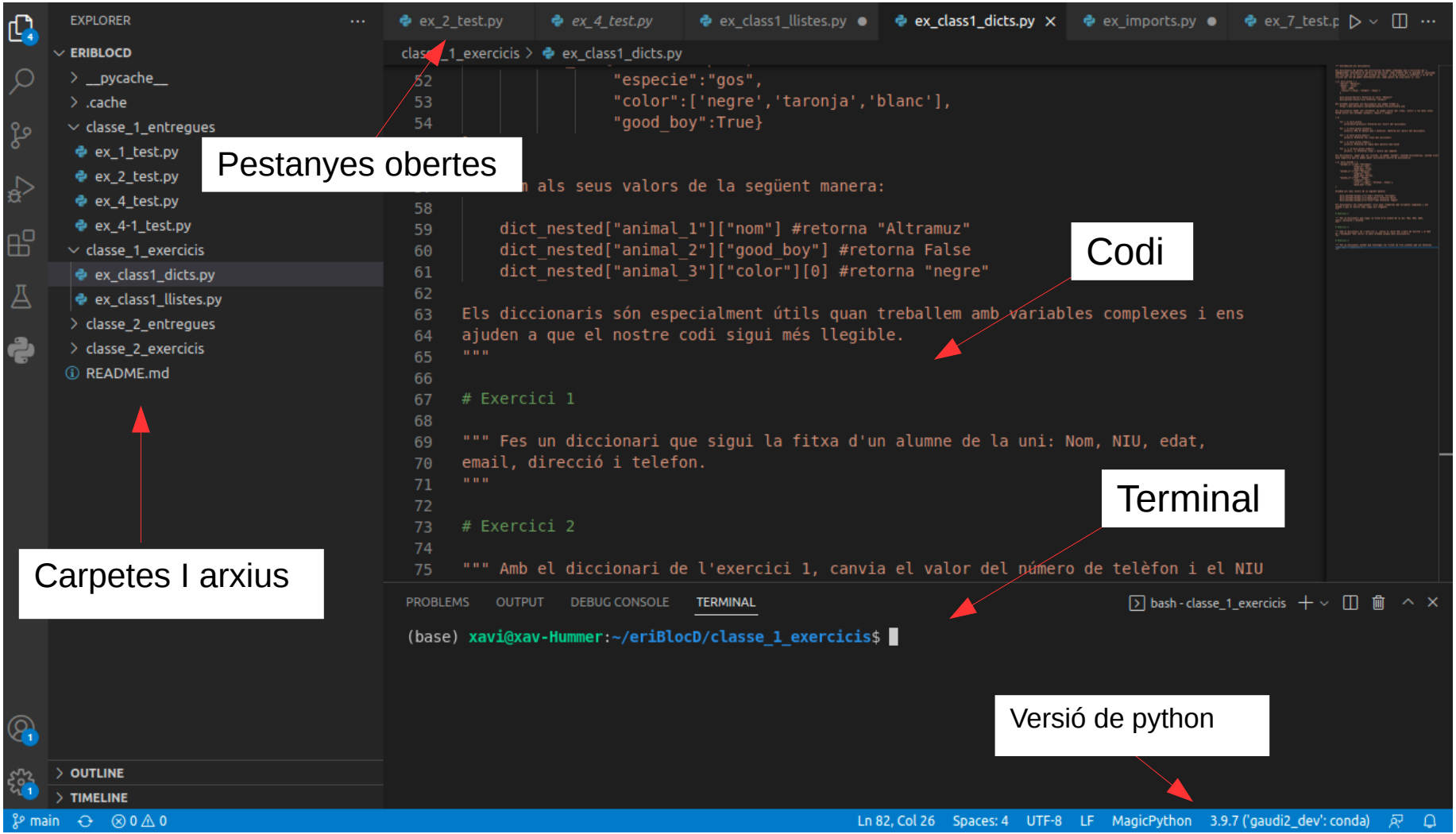
La programació és una disciplina que s'aprèn sobretot de manera autònoma i dedicant-li temps i interessant-se per un/a mateix/a. Tenint una quantitat limitada de classes (5 classes de 2h i escaix) i molt temari per a poder oferir una visió complerta de la programació, es defineix el bloc D com un curs **intensiu**, assumint zero coneixement per part de l'alumne. Els coneixements explicats a classe estan pensats per a ser treballats durant la segona hora i fora de classe per mitjà d'exercicis i projectes entregables. Tantmateix, els mètodes d'avaluació canvien respecte els altres blocs i s'incentiva el treball en grup per a fer més amena la càrrega lectiva del bloc.

Índex per classe (Part 1: 1h teoria. Part 2: 1h teoria/problemes)

- **Classe 0, 1:**
  - Part 1: Introducció a la programació. IDE de python, VSCode, conceptes fonamentals sobre programació (problema real -> formulació algorítmica -> traducció a llenguatge de programació). Bones praxis en programació: escriure pseudocodi -> programar. Hello World. StackOverflow.
  - Part 2: Presentació de les estructures de dades: variables i tipus, llistes i diccionaris. Operadors. Control flow: for, while, if, else, elif, break i continue. Definició de funció I la seva sintaxi. Fer grups de treball i assignar primera entrega.
- **Classe 2:**
  - Part 1: Introducció a la programació modular. Importar mòduls, utilitzar mètodes i funcions de tercers. Presentació del paquet std de python. Operacions sobre llistes, diccionaris (pop, append, key, etc...). Wrappers i casting de variables.
  - Part 2: Funcions en detall: sintaxi, invocació i estructura. Control d'errors: estratègies de debugging, print(), type(), entendre els outputs d'error (SyntaxError, ValueError, NameError, etc). Documentació online de mòduls. Assignar segona entrega.
- **Classe 3:**
  - Part 1: Mòduls de python en l'àmbit científic: NumPy, Matplotlib, SciPy (fsolve per exercicis de PH). STDIN, STDOUT i escriure resultats a fitxer. Carregar arxius csv amb NumPy i ús d'estructures de dades amb funcions (programació funcional). Generar gràfiques a partir de dades reals.
  - Part 2: Introducció a les Classes de Python i al OOP. Estructura de classe, inicialització, objectes, invocar mètodes, atributs, herència. Assignar tercera entrega.
- **Classe 4:**
  - Part 1: Classes i OOP en detall. RDKit i usos en química computacional. Classes aplicades a la química.
  - Part 2: RDkit en profunditat: consolidació dels conceptes de classes i OOP amb la API de RDkit i exercicis guiats a classe. ***A final de classe****: Tria projecte final: Chatbot Twitter, ML app, Joc (snake, tic-tac-toe, game of life*). TO BE DETERMINED!
- **Classe 5:**
  - Part 1: Presentació projecte final: explicació codi (50%), demostració funcionalitat (30%) estructura del codi (20%). Treball final 70% i entregues 30%.
  - Part 2: Contingut addicional per a l'ampliació i comiat. Recollir crítiques/comentaris dels alumnes.

Classe 0: Instal·lació de VSCode (o Spyder) en environment de Conda. Què és un IDE?

- https://code.visualstudio.com/download (idealment ja hauria d'estat tot instal·lat a l'aula)
- Un IDE (Integrated Development Environment) és una interfase on podem crear els nostres programes i ens facilita les tasques relacionades amb el formateig del codi, la compilació, el debugging i d'altres.
- Es pot programar des d'un bloc de notes, però els IDEs estan pensats per a facilitar-nos la vida I tenen multitud d'opcions per a ser personalitzats I treure'n el màxim profit.
- Van des del IDE de python més senzill fins a IDEs on tens total control sobre totes les variables relacionades amb la programació.


EXPLORER
ERIBLOCD
__pycache__
.cache
classe_1_entregues
ex_1_test.py
ex_2_test.py
ex_4_test.py
ex_4-1_test.py
classe_1_exercicis
ex_class1_dicts.py
ex_class1_llistes.py
classe_2_entregues
classe_2_exercicis
README.md
OUTLINE
TIMELINE
ex_2_test.py
ex_4_test.py
ex_class1_llistes.py
ex_class1_dicts.py
ex_imports.py
ex_7_test.p
classe_1_exercicis > ex_class1_dicts.py
"especie":"gos",
"color":['negre','taronja','blanc'],
"good_boy":True}
als seus valors de la següent manera:
dict_nested["animal_1"]["nom"] #retorna "Altramuz"
dict_nested["animal_2"]["good_boy"] #retorna False
dict_nested["animal_3"]["color"][0] #retorna "negre"
Els diccionaris són especialment útils quan treballem amb variables complexes i ens ajuden a que el nostre codi sigui més llegible.
"""
Exercici 1
""" Fes un diccionari que sigui la fitxa d'un alumne de la uni: Nom, NIU, edat, email, direcció i telefon.
"""
Exercici 2
""" Amb el diccionari de l'exercici 1, canvia el valor del número de telèfon i el NIU
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
bash - classe_1_exercicis
(base) xavi@xav-Hummer:~/eriBlocD/classe_1_exercicis$
Pestanyes obertes
Codi
Terminal
Carpetes I arxius
Versió de python
main
Ln 82, Col 26
Spaces: 4
UTF-8
LF
MagicPython
3.9.7 ('gaudi2_dev': conda)

Si en algun moment esteu perduts o perdudes, no enteneu alguna cosa o teniu un bug que no sabeu resoldre...

https://stackoverflow.com/

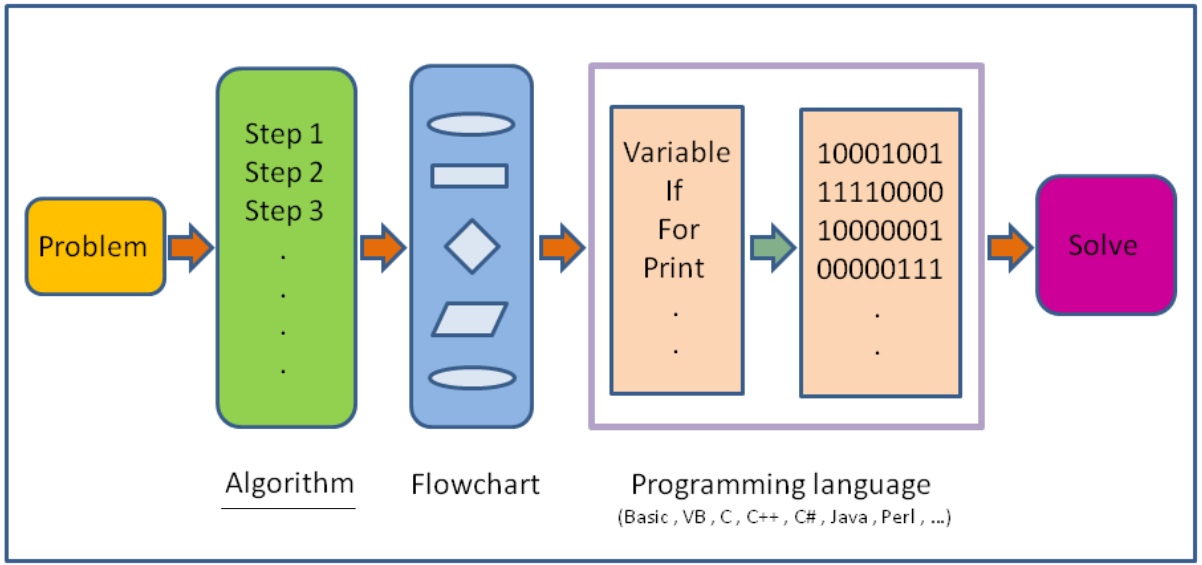
Un ~99% del codi que s'escriu a tot arreu s'agafa d'aquí. És el millor lloc per aprendre a programar I demanar dubtes o resoldre problemes que vagin sorgint.

Classe 1: Introducció a la programació. Plantejament del problema

Problema real → **Adaptació algorítmica (passos concrets)** → Control de flux (primer A, després B, …) → Traducció a llenguatge de programació → END

I.e: Fer una truita:

1) Comprar ingredients
2) Treure estris (paella, bol, espàtula, etc)
3) Pelar patates I ceba
4) Tallar patates I ceba
5) Posar oli a paella
6) Cuinar ceba
7) Apartar la ceba
8) Cuinar patata
9) Batre ous
10) Posar sal
11) ...

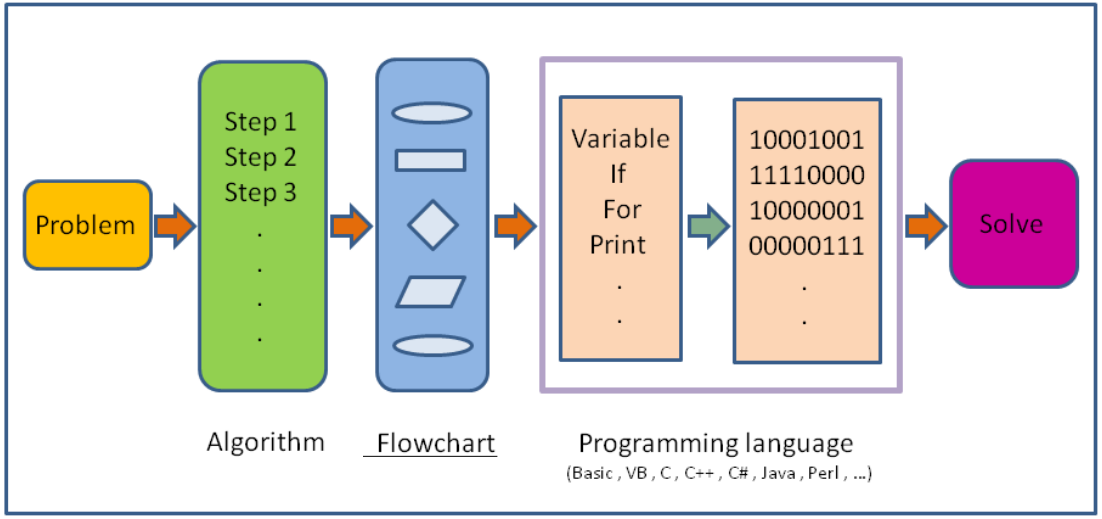


Classe 1: Introducció a la programació. Pseudocodi

Problema real → Adaptació algorítmica (passos concrets) → **Control de flux (primer A, després B, …)** → Traducció a llenguatge de programació → END

I.e: Fer una truita:

1) Comprar ingredients
2) Treure estris (**si l'estri és necessari**: paella, bol, espàtula, etc)
3) Pre-tractament ingredients
   -**Per cada ingredient**: rentar, pelar (si cal), tallar (si cal)
4) Cuinar ingredients
5) Mesclar ingredients
6) Cuinar truita



Classe 1: Introducció a la programació. Adaptació

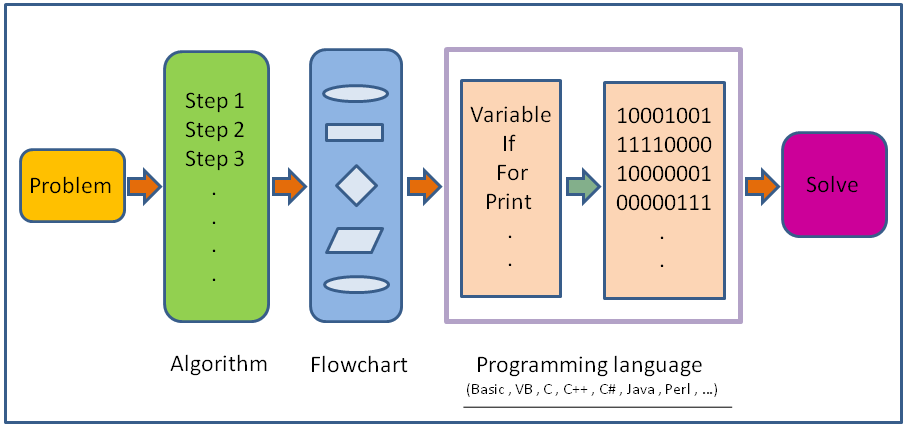
Problema real → Adaptació algorítmica (passos concrets) →Control de flux (primer A, després B, …) → **Traducció a llenguatge de programació** → END

I.e: Fer una truita:

```
1) def buy_ingredients():
       ingredients=[“potato”, “potato”, “onion”, “egg”, “egg”, “egg”, “egg”]
       return ingredients

2) tools = [“pan”, “oil”, “salt”]

3) def cut(ingredients):
       peeled = ingredients.copy()
       for i in range( len(ingredients)):
              peeled[i] = “peeled “ + ingredients[i]
       return peeled
```



Classe 1: Introducció a la programació. Consells

A l'hora de començar a fer un programa complex, feu un primer esbós en paper sobre l'estructura que li voleu donar, les variables que tindrà I les operacions algorítmiques que fareu per a transformar el input A en el resultat que voleu B.

També és recomanable que les variables tinguin un nom que s'entengui I que aquest sigui representatiu del que són. Tots tenim pressa, però passats dos mesos, un d'aquests dos programes no sabreu què fa:

```
def cut(ingredients):
    peeled = ingredients.copy()
    for i in range( len(ingredients)):
        peeled[i] = "peeled " +
ingredients[i]
    return peeled
```

```
def f(x):
    tall = x.copy()
    for i in range( len(x)):
        tall[i] = "peeled " + x[i]
    return tall
```

Classe 1: Introducció a la programació. Hello World!

- **Primer programa**: Hello world!
- Programa que imprimeix per pantalla les paraules “Hello World!”

```
>>> print("Hello World!")
Hello World!
>>>
```

print() → operació de STDOUT (standard output)

- Funció que accepta un argument
- Accepta strings, nombres, etc...

“Hello World!” → String

- A python, les strings son variables que es fan servir per a representar paraules i frases. Similars a les llistes.

Classe 1: Introducció a la programació: Hello World!

- Print() és una **funció** que admet un argument entre parèntesis I imprimeix per pantalla el que li passem
- Podem emmagatzemar un valor donat en una **variable** I imprimir per pantalla la variable. Les variables s'escriuen sempre en minúscules I acostumem a escriure els espais amb barra baixa. I.e: my_var, atomic_distance_matrix, name, etc… Les majúscules les reservem per un altre tipus d'objectes.
- Per a crear una variable, li fiquem un nom I li assignem un valor. I.e:

```
>>> print("Hello World!")
Hello World!
>>>
```

```
var = "Howdy, World?"
print(var)

>>> 'Howdy, World?'
```

Classe 1: Introducció a la programació:Paraules reservades

- Python es reserva certes paraules que nosaltres no podem fer servir com a variables. Generalment, ho veurem molt clar quan una paraula està reservada perquè si intentem fer-la servir com a variable, el nostre IDE l'escriurà amb un determinat color.

- Si intentem definir una variable amb una paraula reservada, ens donarà error. Algunes paraules reservades són: list, string, int, float, ...

False	def	if	raise
None	del	import	return
True	elif	in	try
and	else	is	while
as	except	lambda	with
assert	finally	nonlocal	yield
break	for	not	
class	from	or	
continue	global	pass	

```
my_string = "nom de variable acceptable, en blau"

string = "les paraules reservades surten en verd al VSCode"
```

Classe 1: Introducció a la programació: Tipus de variables

- Els tipus de variables més comuns que farem servir són números o lletres.
- **Int:** variables de tipus enter. Nombres enters, incloent el zero (1, 2, 7262, -6746, 0, etc…)
- **Float:** variables de “punt flotant”. Representen nombres decimals (1.2, -5.664334, 0.6, etc…)
- **String:** variables per a representar paraules, frases o caracters. S’escriuen entre cometes I es poden indexar (“Foo”, “a”, “Foo Bar”, etc…) Per a indexar strings, hem de donar-lis un nom. I.e:
  - my_string = “ABCDEFG”
  - my_string[0] >>> ‘A’
  - my_string[2] >>> ‘C’
- **Bool:** Variables booleanes. Són True i False. Es fan servir amb els operadors per a veure si una afirmació és certa o falsa.
  - Print(9>6) >>> True
  - Print(8>66) >>> False

Classe 1: Introducció a la programació: Operadors

- Els operadors són eines que ens permeten treballar amb variables I obtenir-ne de noves o bé comprovar relacions entre elles.
- **Operadors d'assignació I**
  - **(+):** Suma dos o més nombres o iterables (com llistes o strings)
  - **(-):** Resta dos nombres. No funciona amb lletres o llistes.
  - **(*)**: Multiplica un nombre per un altre o un nombre per un iterable. (Proveu a fer paraula = "hola" I després feu paraula * 4).
  - **(/):** Divideix dos nombres. No funciona amb iterables.
  - **(//):** Divisió entera. I.e: 7 // 3 >>> 2
  - **(%)**: Operació mòdul. Ens retorna el mòdul d'un nombre
  - **(**):** Exponencial. Eleva un nombre a l'exponent que sigui.

Classe 1: Introducció a la programació: Operadors d'assignació

- Els operadors d'assignació ens permeten donar un valor a una variable o bé modificar el seu valor existent.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Classe 1: Introducció a la programació: Operadors de comparació, lògics, d'identitat I de memebresia.

- Els operadors de comparació ens permeten comparar dos valors I ens retornen True o False depenent de si la comparació és certa o no.

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Classe 1: Introducció a la programació: Operadors de comparació, lògics, d'identitat I de memebresia.

- Els operadors lògics ens permeten crear enunciats complexos en els que demanem que certes variables compleixin tot un seguit de relacions concretes.
- Acostumem a fer-los servir dins de bloc condicionals.

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Classe 1: Introducció a la programació: Operadors de comparació, lògics, d'identitat I de memebresia.

- Els operadors d'identitat es fan servir per a comparar directament dues variables

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

- Els operadors de membresia serveixen per a saber si una variable en conté una altra o no

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Classe 1: Introducció a la programació: Llistes

- Les llistes són estructures de dades indexades (comencen al 0) que ens permeten emmagatzemar variables en posicions concretes I permeten valors duplicats. Súper útils a l'hora de programar. Es poden fer servir per a guardar valors que volguem conservar, com a vectors I en general per a guardar qualsevol cosa I modificar-la mentre corre el programa. Es defineixen entre claudators I els valors se separen amb comes. Es poden anidar!

```
my_list_1 = [1,2,[3,4,5],6]

my_list_2 = [“hola”, “bon”, ”dia”]

my_list_3 = [1.33334, True, “Foo”, 42]

my_list_1[2] >>>  [3,4,5]
my_list_2[2] >>> ‘dia’
my_list_3[0] >>> 1.33334
```

Classe 1: Introducció a la programació: Llistes

- Les llistes també accepten tot un seguit de funcions que existeixen per defecte a python I ens ajuden a manipular-les més fàcilment. El llenguatge de python ens permet utilitzar el punt **(.)** per a aplicar una funció a una llista. Veurem els funcions més endavant, però convé que ens familiaritzem amb la sintaxi I que veiem com les apliquem. Per exemple, la funció append() afegeix un valor concret a una llsita. La funció pop() el·limina l'últim valor d'una llista.

```
my_list_1 = [1,2,[3,4,5],6]

my_list_2 = [“hola”, “bon”, ”dia”]


my_list_1.append(300) >>> [1,2,[3,4,5],6, 300]
my_list_2.pop() >>> [‘hola’, ‘bon’]
```

Classe 1: Introducció a la programació: Diccionaris

- Els diccionaris a Python són estructures de dades més ordenades que les llistes. Consten de parells de claus:valors (key:value). A diferència de les llistes, cada parell de valors clau:valor és **únic!** Els diccionaris també es poden indexar, però per claus en comptes de per índex. També es poden anidar. Es defineixen entre claus I els valors es separen amb comes. Els diccionaris, com les llistes, també accepten funcions que veurem més endavant.

```
my_dict = {
    "marca": "Renault",
    "model": "Clio",
     "any": 1996,
     "colors":['blau','vermell','blanc']
    }
```

```
dict_nested = {
    "animal_1":{"nom":"Altramuz",
                "especie":"Gos"},
    "animal_2":{"nom":"Petunia",
                "especie":"Oca"}
}
```

```
my_dict[“marca”] >>> ‘Renault’
dict_nested["animal_2"]["nom"] >>> ‘Petunia’
```

Classe 1: Introducció a la programació: Control Flow

- Un cop tenim controlades les estructures de dades que podem fer servir per a emmagatzemar informació I els operadors bàsics ara hem d'aconseguir transformar un conjunt de variables en un altre. Per això podem fer servir els **condicionals** I els **loops.**

- **Condicionals**:

  - **If / else:** condicional més bàsic. El bloc de codi que escrivim després d'un **if** només s'executarà si la condició que li donem és **certa.** Sinó, l'ordinador ignorarà el bloc I seguirà. Hem d'afegir indentació sota del "if". **else** serveix per a executar el bloc següent en cas de que el "if" sigui fals. I.e:

```
def funcio(x, y):
    if x > y:
        print(x)
    else:
        print(y)
```

```
i=23
if i%2==0:
    print("This is the if block")
    print("i is an even number")
else:
    print("This is the else block")
    print("i is an odd number")
```

```
This is the else block
i is an odd number
```

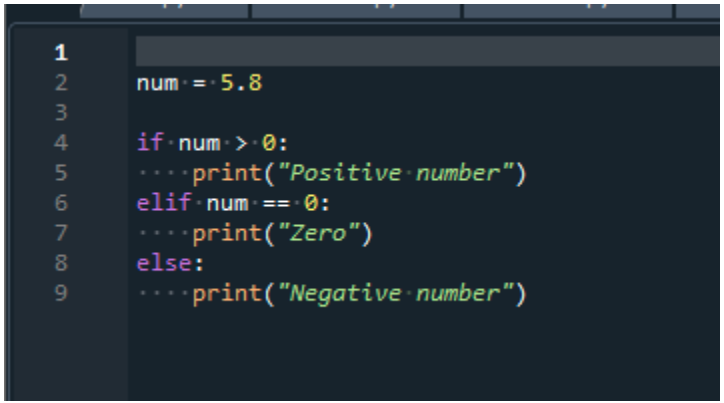
Classe 1: Introducció a la programació: Control Flow

- Un cop tenim controlades les estructures de dades que podem fer servir per a emmagatzemar informació I els operadors bàsics ara hem d'aconseguir transformar un conjunt de variables en un altre. Per això podem fer servir els **condicionals** I els **loops.**

- **Condicionals**:

  - **elif:** condicional complementari al **if**. Si el primer bloc "if" és fals, abans de passar al "else" final, si hi ha **elif,** el programa evaluarà totes les condicions intermitges per a veure si són certes.

```
def funcio(x, y):
    if x > y:
        print(x)
    elif x == y:
        print("Els dos valors són iguals!)
    else:
        print(y)
```



Classe 1: Introducció a la programació: Control Flow

- Un cop tenim controlades les estructures de dades que podem fer servir per a emmagatzemar informació I els operadors bàsics ara hem d'aconseguir transformar un conjunt de variables en un altre. Per això podem fer servir els **condicionals** I els **loops.**

- **Loop "for"**:

  - La seva sintaxi s'assembla molt a l llenguatge humà. Si volguessim iterar sobre una llista diriem: "per cada element d'aquesta llista...". En python es tradueix a: "for element in my_list", on my_list pot ser una variable iterable qualsevol.

  - Generalment, sempre que volguem modificar un seguit de valors dins una llista o diccionari, iterarem sobre ella mitjançant un bucle **for.**

Classe 1: Introducció a la programació: Control Flow

- Un cop tenim controlades les estructures de dades que podem fer servir per a emmagatzemar informació I els operadors bàsics ara hem d'aconseguir transformar un conjunt de variables en un altre. Per això podem fer servir els **condicionals** I els **loops.**

- **Loop for**:
  - El loop més utilitzat. Pot iterar sobre una llista o una string, sobre un diccionari, etc. Es pot fer servir conjuntament amb la funció range() per a que ens faci X iteracions. I.e:

```
my_list = [1,2,3]

for element in my_list:
        element +=1

print(my_list) >>> [2,3,4]
```

Això ens permet modificar llistes o diccionaris!

```
my_list = [1,2,3]

for _ in range(0,5):
        print(my_list)
```

```
>>>[1, 2, 3]
>>>[1, 2, 3]
>>>[1, 2, 3]
>>>[1, 2, 3]
>>>[1, 2, 3]
```

Classe 1: Introducció a la programació: Control Flow

- Un cop tenim controlades les estructures de dades que podem fer servir per a emmagatzemar informació I els operadors bàsics ara hem d'aconseguir transformar un conjunt de variables en un altre. Per això podem fer servir els **condicionals** I els **loops.**

- **Loop while**:

  - El loop **while** és un bucle que es va repetint fins que la condició que li donem de partida s'avalua com a False (fins que quelcom deixa de ser cert).

```
my_num = 0

while my_num < 5:
    print("not yet")
    my_num +=1
print("now!")
```

Resultat:

```
>>>not yet
>>>not yet
>>>not yet
>>>not yet
>>>not yet
>>>now!
```

Classe 1: Introducció a la programació: Control Flow

- Un cop tenim controlades les estructures de dades que podem fer servir per a emmagatzemar informació I els operadors bàsics ara hem d'aconseguir transformar un conjunt de variables en un altre. Per això podem fer servir els **condicionals** I els **loops.**

- **Loop while**:
  - **PRECAUCIÓ:** Els "while" poden arribar a generar bucles infinits! (si la condició sempre és certa, mai es sortirà del loop).

```
while 1:
    print("ad nauseam")
```

Trivia: el valor 0 sempre s'avalua com a False,i el 1 com a True

Resultat:

```
>>>ad nauseam
>>>ad nauseam
>>>ad nauseam
>>>ad nauseam
>>>ad nauseam
>>>...
```

Classe 1: Introducció a la programació: Introducció a les funcions

- Les funcions són una manera convenient d'organitzar codi per a fer-lo més fàcil d'entendre I importar-lo entre programes. Un cop hem fet una funció, la podem reutilitzar sempre que volguem.

- Per a fer una funció, utilitzem la paraula clau **def**, li donem un nom I posem entre parèntesis quines variables agafa (cap, una, dues, etc…). Seguidament, en un bloc indentat, escrivim el que volem que faci I finalment amb la paraula reservada **return** li diem el que volem que ens torni. I.e:

def per a definir-la. Li posem de nom *cut* I accepta la llista *ingredients* com a única variable

```
def cut(ingredients):
    peeled = ingredients.copy()
    for i in range( len(ingredients)):
        peeled[i] = "peeled " + ingredients[i]
    return peeled
```

Usem un **for** loop per iterar sobre els elements de la llista

La paraula reservada **return** ens especifica que la funció ens retorna la llista **peeled**