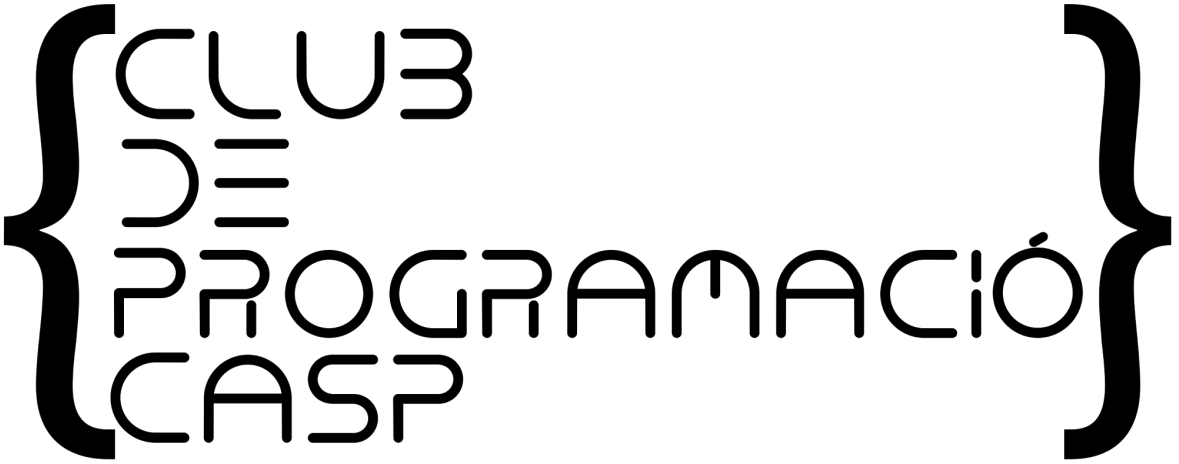


Introducció a Python

Sergi Gilabert Sempere

Índex de continguts

Introduint aquest document

En aquest document trobaràs tot allò bàsic per aprendre Python, des d'allò més simple com és enunciar una variable fins a cridar llibreries i utilitzar-les en el codi. En aquest cas no hem entrat en la programació orientada a objectes introduint conceptes importants com són ara les classes, ja que per fer programes bàsics trobem que és suficient.

La idea d'aquesta guia és que es combini amb aprenentatge autodidacta per part del lector, com ara el cas d'utilitzar altres llibreries: nosaltres hem introduït dues llibreries que trobem importants, com ho són ara Tkinter i random, però el que estaria bé és que servis d'exemple per fer servir altres llibreries, adaptant el contingut a cada llibreria concreta. Les funcions d'aquestes, però, les haurà de cercar el lector per internet. Nosaltres hem après en gran part per internet, i això és el que volem transmetre.

Recomanem utilitzar la documentació de Python i la de les llibreries que l'usuari d'aquesta guia vagi a fer servir com a complementació d'aquesta. També, a l'hora de resoldre dubtes també recomanem que us registreu a Stack Overflow, un fòrum de programació, i busqueu si la vostra pregunta ja ha estat feta. En cas negatiu, us recomanem que la publiqueu.

En el cas dels problemes, molta part són problemes model que podem trobar a qualsevol web o llibre d'aprenentatge del llenguatge, però hi han problemes concrets extrets d'edicions anteriors de l'HP Codewars per exemple. Hem utilitzat els d'aquesta competició ja que els trobem diferents a tots els que hem fet, tenen un punt de complexitat més enllà del simple output del valor. També s'ha de dir que la gran majoria dels problemes els hem 'fet' nosaltres a partir de la nostra experiència com a aprenents del llenguatge o inventats a l'hora d'escriure la teoria per buscar un exemple clar de com s'ha d'executar cada cosa.

Per segons quines coses recomanem utilitzar l'IDE PyCharm degut a la quantitat de característiques que porta, la suggerència de variables, classes i funcions a l'hora d'escriure… A més, per treballar amb llibreries com ara Tkinter trobem que és més còmode i al poder personalitzar l'interfaç del programa més profundament que l'IDE de Python oficial permet que el programador se'l personalitzi al seu gust i li 'atragui' més utilitzar el programa, però això és purament estètic.

Res més a dir, esperem que aquesta guia del llenguatge serveixi per aprendre a programar. Ànims i a programar!

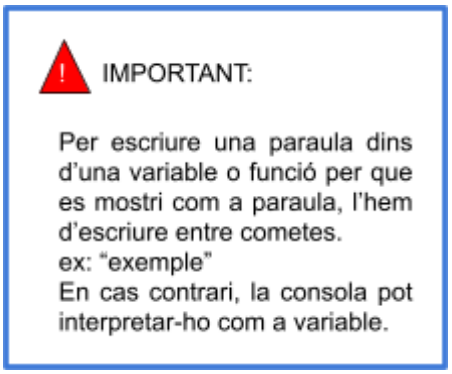
1. Introducció a Python

Python és un llenguatge de programació interpretat, és a dir, que necessita un programa per poder interpretar/executar el codi. És per això que necessitem instalar el programari de Python des del seu web (www.python.org). Quan haguem instal·lat el programa, haurem de obrir el IDLE, que és un processador de text destinat a Python que porta inclòs una consola amb la capacitat d'interpretar el codi. En aquest cas, no tindrem interfaç gràfica per ara, sino que els nostres programes correran en consola. Hi ha una forma de tenir UI mitjançant una llibreria anomenada Tkinter, però per ara ens centrarem en dominar Python per aprofundir en llibreries després.

Primer de tot, començarem amb l'unitat més bàsica, que és la **variable.** Una variable permet guardar un valor per fer-lo servir més endavant durant tot el codi. Es defineix mitjançant la sentència:

```
nomVariable = "python"
```

nomVariable és el nom que li hem donat a aquesta. El nom de la variable no pot tenir espais ja que llavors ens donarà error. Per conveni, quan escrivim dues paraules juntes, es sol escriure la primera lletra de la primera paraula en minúscula i la de la segona en majúscula per diferenciar les dues paraules. A la dreta de l'igual trobem el valor que li hem assignat a la variable, en aquest cas és la paraula *python.*



Ara bé, de que em serveix només tenir una paraula que pugui enmagatzemar alguna cosa si després no puc fer-ho servir. Què puc fer amb això?. Bé, ara us ensenyarem una funció per fer que aparegui alguna cosa a la consola, una frase per exemple o una operació matemàtica senzilla. Aquesta funció s'anomena **print:**

```
print("exemple")
```

El funcionament de print és similar al de les funcions matemàtiques, on f seria la funció print i la x dins del parèntesi allò que volem que surti 'imprès' a la pantalla. Teclejant exactament el codi que trobem aqui a sobre a la pantalla de la consola sortirà això:

```
exemple
```

Però, també podem mostrar altres coses que no fa falta que escribim nosaltres dins de la funció, ara és quan les variables comencen a tenir més sentit. Si en canvi de escriure "exemple" dins del parèntesi escribim el nom de la variable que hem definit abans, nomVariable, d'aquesta forma:

```
print(nomVariable)
```

En canvi d'imprimir `nomVariable`, imprimirà el valor que li hem donat prèviament a la variable, en aquest cas `python.` És important que ho escrivim sense cometes ja que si ho escribim així ens 'imprimirà' `nomVariable`, ja que en el món de la programació quan escribim alguna cosa entre cometes és perque li estem dient que és una paraula/frase i no forma part de cap funció ni variable. També podem complementar text predefinit per nosaltres i variables de la següent forma:

```
print("El nom " + nomVariable + " és un nom d'origen àrab")
```

Hem d'utilitzar el signe de la suma per unir sentències i variables. Hem de deixar un espai al final de la primera sentència i un al principi de la segona sentència per que no surti així:

`El nomMohammedés un nom d'origen àrab` - sense espais a principi i final.
`El nom Mohammed és un nom d'origen àrab` - amb espais a principi i final.

Ara que ja sabem com imprimir una variable, anem una mica mes enllà, com podria fer que jo o l'usuari que està utilitzant el programa determini el valor de la variable? És possible?. Sí que ho és, i en aquest cas utilitzarem la funció **input().** input() permet que l'usuari entri dades que després poden ser aplicades al programa a través de variables. Una variable que permeti guardar el que l'usuari li digui quedaria així:

```
nomVariable2 = input()
```

Un detall a recordar és que ho hem d'escriure per ordre. No podem escriure que primer imprimeixi la sentència amb la variable sense determinar i després et pregunti el valor.

Exercici per practicar:
Fes un programa que pregunti el nom, l'usuari pugui escriure'l i li doni la benvinguda dient el seu nom (ex: Bona tarda Joan!)

2. Veritat, fals, i res en absolut

Què és un boolean? Un boolean és un valor que li donem a una variable per enunciar un estat. Aquest valor pot ser True, False, None… Els booleans més bàsics són True i False, que en electrònica True = 1 i False = 0. És per quan una variable ha de guardar un estat i no un valor. Podríem fer el símil d'un botó: True significaria que el botó està encès, osigui que dóna electricitat i False que està apagat.

```
entradaPermesa = False
```

Hi han booleans per dir que una variable no té cap valor, com podria ser ara aquest cas:

```
valorProva = " "
```

3. Operadors, la base de la programació

Els operadors és una de les coses més elementals de la programació. Són els que en permeten manipular i comparar variables i valors. Hi han diferents tipus d'operadors, entre els que destaquem aquests:

Aritmètics	+, -, *, /, %
Comparatius	==, >, <, <=, >=, !=
Lògics	`and, or, not`
D'assignació	=, +=, -=, *=, /=

+	Per sumar valors o strings
-	Per restar valors
*	Per multiplicar valors o strings
/	Per dividir valors
==	Per dir que una variable és igual (osigui, que té el mateix valor) que una altra
<	Per dir que la variable escrita a l'esquerra és més petita que la escrita a la dreta

>	Per dir que la variable escrita a l'esquerra és més gran que la escrita a la dreta
+=	Per sumar un valor a una variable, estalviant-nos codi
!=	Per dir que una variable no és igual a una altra
and	Nexe conjuntiu utilitzat a les sentències dels condicionals per dir que s'ha de complir més d'una sentència
or	Nexe utilitzat també a les sentències dels condicionals per dir que s'ha de complir una o l'altra sentència escrita al condicional.
not	Serveix per revertir el valor dels and i or.

4. Enter, amb decimals o no és numèric?

Dins les variables, trobem diferents tipus, les variables int() (integer) que són variables amb contingut numèric sense decimals, float() que són variables amb contingut numèric amb decimals i str() (string) que són variables amb contingut de text i similars. Quan fem print, no podem fer que imprimeixi una variable float o integer, per això ho hem de passar a string d'aquesta forma:

```
edat = int(input())
edat += 10
print(edat)
```

No li podriem haver sumat 10 a edat com a valor numèric si no haguéssim especificat que és un valor numèric.

5. Introducció als condicionals

Comencem amb el fonament de la programació, els condicionals. Els condicionals a Python s'escriuen de forma molt intuitiva i molt semblant a com ho escriuriem en anglès. Els condicionals estan composats per:

```
if sentència:
      #codi que s'executarà
```

Sempre comencem per if juntament amb una sentència. Aquesta sentència és la que especifica que s'ha de complir (ha de retornar True) perquè s'executi el codi següent, que en aquest cas seria tornar la variable entradaPermesa a False (False és un boolean, serà el següent que tractem). En aquest cas edatEntrada seria una variable que mitjançant una dada anterior ja sigui amb un input o escrita directa al codi. Aquest codi no tindria per ara cap canvi que es mostrés a la consola però dins del if podem introduir inputs, variables, prints…
Ara bé, i si volem que es compleixin dues sentències o una de dues sentències en canvi d'una? llavors utilitzarem els connector and i or. And el farem servir per connectar dues sentències i deixar clar que s'han de complir els dos enunciats, com per exemple:

```
if edat < 18 or drunk == True:
      entradaPermesa = False
```

En cas de voler afegir una altra sentència però donant-li un codi diferent, osigui, que faci una altra cosa utilitzarem elif:

```
if edat < 18 or drunk == True:
      entradaPermesa = False
elif edat >= 18 and teDNI == True:
      entradaPermesa = True
```

Pots posar tans elif com vulguis. Però, si per altra banda, volem executar un codi per la resta, és a dir, els que estan exclosos de les darreres sentències, utilitzarem else (no li fa falta cap sentència)

```
if edat < 18 or drunk == True:
      entradaPermesa = False
else:
      entradaPermesa = True
```

6. Fins l'infinit i més enllà

Ja sigui per necessitat pròpia del programa o per divertir-te executant codi infinitament algun cop necessitaràs que el programa s'executi un nombre determinat de cops o infinitament. Per això fem servir els loops. En aquest cas introduirem els while, que té aquesta estructura:

```
while True:
      print("test")
```

En aquest cas estarà repetint infinitament però també podem fer que ho repeteixi un nombre determinat de vegades o mentre una sentència es cumpleixi

```
vegadesExecutat = 0
while vegadesExecutat < 18:
      vegadesExecutat += 1
```

Dins dels condicionals hi ha una paraula que farà que es deixi d'executar el loop. Aquesta paraula és break. break és exclusiva dels loops (és a dir, no podem utilitzar-la fora d'aquests) que 'trencarà' amb la rutina que hem especificat. Un exemple seria aquest:

```
while True:
    print("Siusplau, no escriguis 'hola':")
    n = input()
    if n == "hola":
        break
```

Aquí especifiquem que es repeteixi infinitament el programa, i que aquest li demani que no entri la paraula hello, si l'usuari escriu la paraula, el loop deixarà d'executar-se.

Pre-7. Repàs del temari

Per processar tot el que hem après fins ara, fem un petit codi que ens mostra com aplicar tot el que hem vist fins aquest punt en un mateix codi:

```
while True: #un while serveix per repetir un codi segons una sentència
      print("Benvinguts a la tercera sessió") #els prints serveixen per
      poder
      print("Vas gaudir de la darrera sessió?")  #mostrar text a la
      consola
      resposta = input()   #els inputs són necesaris per que l'usuari
      entri text
      if resposta == "si": #els condicionals serveixen per executar codi
      segons una condició
            print("Ens alegrem, gaudeix d'aquesta")
      elif resposta == "no":
            print("Esperem que t'ho passis millor avui, ja que venim
            carregadets de temari")
      else:
            print("No estic programat per entendre el que m'has escrit)
```

També, no ens oblidem, tenim els integers, els floats i les strings que serveixen per donar-li una propietat a una variable. Els integers són nombres no decimals, els floats són nombres decimals i les strings són cadenes de text i altres.

També, si volem saber la llargada d'una string o d'una llista fem servir la funció len(var) on var és la llista o la variable.

Bé, ara que ja hem repassat, seguint donant-li '*canya*' al codi.

7. Variable = llista

Ara que ja coneixem prou bé el concepte de variable, anem a introduir un tipus d'aquestes. Hi ha tipus de variables que depenent de la complexitat amb la que vulguis guardar dades (valors dins de valors, com en el cas dels diccionaris, o múltiples valors). La que presentem avui es diu **Array**. Un array és una variable que permet guardar moltes dades dins d'una sola variable. Aquestes dades, per poder accedir a elles, s'ha de tenir en compte l'ordre en el que estan guardades. L'ordre es comença a contar des de zero. Un exemple:

```
albumsQueen = ("Queen", "Queen II", "Sheer Heart Attack, "A Night at the Opera", "A Day at the Races", "News Of The World", "Jazz", "The Game", "Hot Space", "The Works", "A Kind of Magic", "The Miracle", "Innuendo", "Made in Heaven")
print(albumsQueen[3])
```

El programa si executem aquest codi imprimirà a la consola el següent:

```
A Night at the Opera
```

També, si volem convertir un input que ens faci l'usuari a un array, osigui que el usuari ens doni uns valors separats per espais utilitzarem la funció str.split(), que s'utilitza de la següent forma:

```
puntuacionsTrimestre = input()
dadesSeparades = str.split(puntuacionsTrimestre)
print(dadesSeparades[2])
```

Si l'usuari li dona aquest input:

```
7 8 5 1 10
```

El programa retornarà això:

```
5
```

O si en canvi desitgem separar una paraula o més en una array on cada valor sigui una lletra no fa falta separar-ho, un for amb aquesta forma buscarà cada lletra:

```
print("Introdueix una paraula i et diré si té la lletra e")
paraula = input()
for char in paraula:
      if char == "e":
            print("Conté e")
            break
```

8. For(s), més loops que una muntanya russa

En aquest tema introduïm els for, que són un altre tipus de loops, però aquest s'executa un nombre de vegades que li determinem nosaltres, no com el while que s'executa fins que la condició sigui falsa. Els for tenen la següent estructura:

```
for letter in "python":
      print letter #aquí li estem dient que per cada lletra de la paraula
#python, imprimeixi la lletra. Això ho executarà fins que no quedin
#lletres per imprimir
```

A l'executar aquest codi, ens sortirà el següent output:

```
P
Y
T
H
O
N
```

En aquest cas, letter és una variable itinerant, segons la informació de la segona variable/dada, la posterior a in.

Una de les formes més conegudes dels for és la següent:

```
for i in range(num):
      print("3x"+i+"="3*i)
```

En aquest cas, i és una variable que no hem enunciat abans (pot ser qualsevol, no fa falta i) i num ha de ser els cops que ho volem repetir (si posem una variable aquesta sí que ha d'estar enunciada). La variable i tindrà un valor diferent cada cop que s'executi el for (des de 0 fins al que li haguem posat a la funció range), aquesta variable l'anomenem variable itinerant.

Vídeo explicatiu: Diferències entre For i While

9. Funcions, estalviem codi.

Entrem a una part bastant important d'un llenguatge de programació, les funcions. Una funció és un element dins del llenguatge que et permet executar un seguit de codi mitjançant una paraula determinada amb la següent forma: `nomFuncio()`. Dins dels parèntesi podem afegir incògnites que poden ser utilitzades dins del codi, ara ho explicarem més a fons.
Durant aquests dies hem trobat un munt de funcions, com ara la funció `print()` que ens permet mostrar un seguit de caràcters a la consola. Però no totes les funcions estan determinades pel llenguatge, nosaltres també podem determinar-ne amb la següent sentència:

```
def nomFuncio():
      print("Prova de Funció")
```

D'aquesta forma hem determinat que quan nomFuncio() s'executi, imprimirà "Prova de Funció" a la consola. Però això no és tot, ja que per ara només l'hem definida, ara hem d'executar-la de la següent forma:

```
def nomFuncio():
      print("Prova de Funció")
nomFuncio()
```

```
Prova de Funció
```

D'aquesta forma hem definit i executat la funció. Aquesta és la forma més bàsica. Com hem dit abans, també podem afegir incògnites a dins del parèntesi per fer-ho servir dins el codi a partir de variables existents o d'un text. Un exemple:

```
def moltesFelicitats(x):
      print("Moltes felicitats, " + x + "!")
print("De qui és l'aniversari avui?")
nom = input()
moltesFelicitats(nom)
```

Com hem vist a l'anterior exemple definim una incògnita qualsevol, en aquest cas l'hem anomenat x però podriem haver-la anomenat d'una altra forma. La incògnita x només ens servirà per marcar on anirà la variable que introduirem entre els parèntesis. Podem fer que hi hagin més variables disponibles per fer servir mitjançant una coma, d'aquesta forma

```
def moltesFelicitats(x, y):
```

I a l'executar-la hauríem d'introduir les incògnites de la següent forma:

```
moltesFelicitats(nom1, nom2)
```

On nom1 seria x i nom2 seria y dins de la definició.
Les funcions ens seran molt útils per estalviar-nos codi, així no ens farà falta repetir sempre el codi, només cridant a la funció.

En canvi de print() dins d'una funció podem fer servir l'estructura return d'aquesta forma:

```
def autoPrint():
      var1 = input()
      return var1
```

10. Per què no em corre el codi?

Durant les darreres classes us haureu trobat que el vostre codi no us funciona, normalment sol ser per errors tipogràfics però altres cops Python ens mostra una frase amb el que li passa al nostre codi. Ara és el torn d'aprendre a desxifrar els errors.

`ArithmeticError`	Error a l'hora de calcular una operació
`OverflowError`	Quan un càlcul matemàtic excedeix el nombre màxim de números que es pot processar
`ZeroDivisionError`	Quan es divideix entre zero
`NameError`	Quan no es troba el nom d'una variable
`SyntaxError`	Quan hi ha un error a la sintaxi de Python, osigui, falta dos punts, comas, cometes,...
`KeyError`	Error a l'hora de determinar la posició d'un diccionari
`ValueError`	Error a l'hora d'operar dos tipus diferents de valors (str entre int…)
`IOError`	Quan hi ha hagut un error amb un input o algun output (com podria ser un print)

Normalment aquest codi va acompanyat del número de línia on trobem l'error o la línia impresa. Aquí us deixem uns exemples:

```
while True print("Hello world")
          ^
SyntaxError: invalid syntax → falta dos punts abans del print

x = 1/0

ZeroDivisionError → s'ha intentat dividir un número entre zero
```

Hi ha una estructura que si sabem que ens pot arribar a donar error una peça de codi, podem utilitzar-la per que si ens dóna un error exacte, executi una altra peça de codi en canvi. L'estructura és la següent:

```
try:
      #aqui aniria el codi
except nomError:
      #aqui aniria el codi en cas de que sortís l'error especificat
```

exemple (si num = 0):

```
try:
      rel = 5/num
      print(rel)
except ZeroDivisionError:
      print("El número introduït no pot ser 0")
```

11. Diccionaris

Bàsicament els diccionaris són com les arrays però en aquests cas els valors en canvi d'accedir al valor mitjançant la posició en la que es trobava ho farem amb un altre valor que definirem abans de cada valor.

```
notesMates = {"Oscar": 7, "Pere": 3, "Gerard": 10, "Francesc": 9, "Arnau":
5, "Sergi": 5}
print(notesMates["Gerard"])
```

```
10
```

Dins el diccionari, en el cas anterior "Oscar" l'anomenarem la clau (key, en anglès) i el 7 l'anomenarem valor. Sobre la marxa, després d'haver creat el diccionari podem afegir valors de la següent manera:

```
notesMates["Josep"] = 2
```

En aquest cas hem creat la clau "Josep" i li hem assignat el valor 5. Ara el diccionari serà així:

```
notesMates = {"Oscar": 7, "Pere": 3, "Gerard": 10, "Francesc": 9, "Arnau":
5, "Sergi": 5, "Josep": 2}
```

També podem actualitzar el valor d'una clau, per exemple, en aquest context si "Pere" ha recuperat Matemàtiques, fariem així per actualitzar la seva nota:

```
notesMates["Pere"] = 5
```

I, en conseqüència, el diccionari quedaria de la següent forma:

```
notesMates = {"Oscar": 7, "Pere": 5, "Gerard": 10, "Francesc": 9, "Arnau":
5, "Sergi": 5, "Josep": 2}
```

I, si en canvi volem esborrar la clau "Oscar", farem el següent:

```
del notesMates["Oscar"]
```

O volem esborrar tot el contingut:

```
notesMates.clear()
```

Posa't a prova!

Ara posarem a prova els nostres coneixements de Python. Dividirem la classe en dos grups que es dedicaran a fer un projecte: un encriptador. Un encriptador de text codifica un text per que un tercer no pugui llegir-lo. El que caldrà fer es, per exemple, canviar les lletres per altres. La meitat haurà de fer l'encriptador i l'altre meitat el desencriptador. Per això haureu de posar-vos d'acord en quin patró de codificació segueix per que l'encriptador i el desencriptador siguin compatibles. A cada meitat es faran parelles que hauran de posar-se d'acord amb una altra parella de l'altre meitat per realitzar uns l'encriptador i els altres el desencriptador. Cada grup (dos parelles) hauran d'inventar-se el seu propi llenguatge per encriptar. Quant més original, millor!

12. Fitxers, interacció fora del codi.

Primer de tot hem d'assignar una variable a la funció principal per obrir l'objecte. Bàsicament li assignem perquè sigui més fàcil de cridar posteriorment. Per obrir un arxiu li assignarem a una variable la funció open(valor1, valor2) on valor1 serà el nom del arxiu que volem obrir dins de la mateixa carpeta on guardem el programa i valor2 el mode en que volem obrir l'arxiu. Si volem obrir l'arxiu donant permisos al programa perquè pugui editar-lo, li assignem la lletra "w", si volem obrir l'arxiu donant-li permisos de lectura exclusivament li assignarem la lletra "r". El codi quedaria així si volem obrir l'arxiu.

```
f = open("nomfitxer.txt", "w")
```

Molt bé, ara tenim l'arxiu obert, en aquest cas en mode escriptura ja que hem especificat la lletra w (write). Ara per escriure, a sota haurem de cridar la variable de l'arxiu amb la funció write(), per exemple així:

```
f.write("El text aquí o una variable")
```

Recorda que si fas servir aquesta funció es sobreescriurà el que hi hagi escrit dins de l'arxiu.
També podem, com hem mencionat abans, llegir el contingut de l'arxiu i utilitzar-lo dins del programa. Per exemple, si volem que el text de l'arxiu estigui dins d'una variable, per posteriorment imprimir-la, utilitzariem la següent funció:

```
contingutText = f.read()
```

13. Interfaç gràfica a Python

PER FER AQUESTA PART FARÀ FALTA UTILITZAR L'IDE PYCHARM
Ens desviem una mica del tema central del curs per aprendre una cosa necessària per ampliar el poder d'un llenguatge de programació. Això són les llibreries. Per això prescindirem del IDE que hem estat utilitzant fins ara per utilitzar un que ens dóna més possibilitats i ens permetrà treballar de forma més còmode amb llibreries, aquest és PyCharm.

Per posar un exemple de com es fan servir les llibreries us mostrarem com funciona Tkinter, una llibreria que ens permet crear una interfaç gràfica pels nostres programes. Primer de tot començem en el codi essencial.

Python, abans de tot, necesita que cridis a les llibreries que et son necessàries per executar el codi. Per exemple, per executar Tkinter utilitzarem aquest codi al inici de tot:

```
import tkinter
```

D'aquesta manera estas dient que importi tot el contingut de la llibreria. Una altra forma seria la següent:

```
from tkinter import *
```

En aquest cas li diem el mateix. Amb l'asterisc li estem dient que importi tot el contingut, però podriem dir-li que només importés una part de la llibreria. La pregunta és: quins avantatges tenen cadascuna de les opcions? La primera opció ens permet importar més d'una llibreria a la vegada però no ens permet especificar quina part de la llibreria vols importar, sino que la importa sencera.

Ara, seguint amb Tkinter: Primer de tot hem de crear la finestra on executar el codi, li direm la finestra root. La crearem de la següent forma:

```
root = Tk()
root.mainloop() #necessari perque la finestra es mantingui oberta
```

D'aquesta forma hem creat la finestra, per ara no hi hem posat res a dins. Hi han moltes funcions que ens permeten posar coses a dins, a continuació hi trobarem un seguit d'exemples:

```
text = Label(root, text="Hello World")
imatge = PhotoImage(file="lloc on és el fitxer")
Botó = Button(master, text="text boto", command="") #A command hi posarem
#una funció que faci el que volem que passi quan cliquem el botó
```

```
caixaText = Entry(master) #i per recuperar el text# var = caixaText.get()
```

I per organitzar-ho una mica, usarem el mètode:

```
variableObjecte.pack()
```

Ara, en el cas dels botons haurem de crear una funció on especifiquem què volem que faci el programa quan es cliqui el botó. Un cop definida la funció haurem de escriure-ho a l'apartat command dins la funció Button.

14. Diguem un número a l'atzar

Anteriorment us em parlat de les llibreries (com ara Tkinter, per crear interfaç gràfica). En aquest cas us parlarem d'una que probablement la utilitzareu sovint i que no podem acabar el curs sense que la conegueu. Aquesta llibreria s'anomena random i, com el nom indica, ens servirà per aconseguir valors aleatoris per els nostres programes. Com tota llibreria la cridarem de la següent forma:

```
import random
```

IMPORTANT: NO ANOMENEU L'ARXIU EN EL QUE ESTEU PROGRAMANT NI CAP DINS LA MATEIXA CARPETA COM LA LLIBRERIA, LLAVORS US DONARÀ ERROR A L'HORA DE CORRE EL PROGRAMA.
Dins d'aquesta llibreria trobem les següents funcions.

```
random.random()#ens dona un número a l'atzar
random.randrange(x, y, z)#número a l'atzar entre x i y sense contar z
random.sample(range(x,y), n)#per que ens doni n números entre x i y
random.randint(x, y)#número a l'atzar entre x i y
random.choice(llista)#escollirà un valor a l'atzar d'una llista
random.shuffle(llista)#desordena els valors dins d'una llista
```

15. Programació funcional I: Funcions lambda

Les funcions lambda formen part d'una part de Python anomenada programació funcional. És una forma d'expressar el codi de forma similar a una funció matemàtica.
Les funcions lambda concretament són funcions que poden ser escrites en una línia. Són molt semblants a les funcions matemàtiques, com podem observar en el següent exemple:

```
f = lambda x: x*2
f(2)
```

```
4
```

Aquest darrer exemple matemàticament s'escriuria de la següent manera:

$$f(x) = x * 2$$
$$f(2) = 2 * 2$$
$$f(2) = 4$$

Com podem veure és una forma d'escriure una funció bastant simple, podria ser capaç de ser entesa per qualsevol persona amb coneixements matemàtics però sense coneixements de programació. Aquesta funció a més ja porta implicitament un return. Com podem veure en el primer exemple, lambda porta incorporat un return, és a dir, retorna automàticament el resultat de la funció.

Les funcions lambda a més no els fa falta estar assignades a una variable, poden ser utilitzades de la següent manera:

```
print((lambda x: x**2 + 5*x + 4) (2))
```

```
18
```

16. Programació funcional II: funcions map i filter

La funció map ens pot ser molt útil per programar de forma més eficient. En una sola línia de codi podem executar una funció sobre una llista de la següent manera:

```
def func_ex():
      return x + 1
llista = [2, 4, 6, 8]
resultat = list(map(func_ex, llista))
print(resultat)

[3, 5, 7, 9]
```

Cal dir que hem utilitzat la funció list, que l'únic que ha fet és agrupar tots els valors resultants dins d'una llista. La funció map el que ha fet és executar la funció *func_ex* amb els valors de *llista*. Ja que parlem de programació funcional, podriem simplificar el codi de la següent forma:

```
llista = [2, 4, 6, 8]
resultat = list(filter(lambda x: x + 1, llista))
print(resultat)
```

Una altra funció útil és filter. Aquesta funció, que està estructurada de la mateixa forma que la funció map, filtra a partir d'una condició en forma de funció uns valors assignats per nosaltres. Funciona de la següent forma:

```
llista = [2, 3, 4, 5]
resultat = list(filtter(lambda x: x%2==0, llista))
print(resultat)

[2, 4]
```

Com podem veure en aquest darrer exemple, hem creat una funció lambda que comprovava si un número dins de *llista* era parell, i si era parell el guardava dins de la llista *resultat*.

17. Python orientat a objectes

PROJECTES I EXERCICIS PER PRACTICAR

PROJECTE: TRADUCCIÓ DEL CODI GENÈTIC

En aquesta sessió no donarem més teoria sinó que farem un petit projecte entre tots. Utilitzarem tots els coneixements apresos durant aquest curs de Python per crear un programa a partir d'unes dades donades (taula de proteïnes) i un input.

El codi genètic és una seqüència de nucleòtids a l'ARN que són traduïts en proteïnes utilitzat pel nostre cos per determinar quines proteïnes són sintetitzades per a les diferents funcions del nostre cos. Aquest codi consisteix en quatre bases nitrogenades que es combinen en grups de tres: Citosina (C), Guanina (G), Timina (T) i Adenina (A), i en el cas de l'ARN: Adenina (A), Uracil (U), Guanina (G) i Citosina (C). L'equivalència és:

ADN	ARN
A	U
T	A
C	G
G	C

El projecte consisteix en crear un programa que consisteixi en traduir una seqüència d'ADN en ARN i dir en quins aminoàcids es tradueix l'ARN. Aquest programa ha de ser capaç de detectar si la cadena introduïda és una cadena d'ARN o d'ADN. A partir de la següent taula i de tot el que hem après a classe heu de poder fer un programa que compleixi les següents característiques.

Com a apartat opcional, us proposem que li doneu interfaç gràfica al programa mitjançant la llibreria Tkinter que hem pogut veure les darreres sessions.

És recomanable que definiu funcions amb el propòsit de estalviar codi i en cas de fer l'apartat opcional, seria més fàcil d'adaptar el codi.

RECOMANACIONS TEÒRIQUES:

`array.append("algo que vulgui afegir")` → afegir alguna cosa a una array

`[var[i:i+n] for i in range(0, len(var),3)]` → on var és la variable que vols dividir i n el número de caràcters

```
try:
      codi            això ens permet executar un codi, i que si hi ha un error
except codiError:     especific executi un altre codi
      codi
```

1. Fes un programa en el que demanis el nom d'una persona per després donar-li la benvinguda.

2. Fes un programa que et permeti calcular l'any de naixement d'una persona a partir de l'edat que cumpleix a l'actualitat.

3. Fes un programa en el que el usuari et doni dues edats i el programa determini quin dels dos usuaris és més gran. Només ha de retornar l'edat del més gran.

4. Escriu un programa que a partir de l'alçada i el pes et calculi l'IMC d'una persona. El programa ha de retornar l'IMC. (IMC = pes/alçada(en metres)^2). Si es vol aprofundir, el programa també pot retornar un veredicte per l'IMC (si està obès, té sobrepès…) amb les dades d'aquesta taula:

	homes	dones
Falta de peso	por debajo de 20	por debajo de 19
Peso normal	20-25	19-24
Sobrepeso	26-30	25-30
Obesidad	31-40	31-40
Fuerte obesidad	mayor de 40	mayor de 40

5. Fes un programa que et permeti resoldre equacions de segon grau. L'únic input de l'usuari és l'equació

6. Escriu un programa que et permeti calcular la quantitat de dies i d'anys de traspàs a partir de dos anys proporcionats per l'usuari

7. Fes un programa que et calculi la taula de multiplicacions de qualsevol número que entri l'usuari. El programa només ha de donar un màxim de 11 resultats..

8. Fes un programa que et permeti calcular la lletra d'un DNI a partir dels números proporcionats per l'usuari i que et retorni el DNI complet (amb lletra) sabent això:

   DNI%23 → 0=T; 1=R; 2=W; 3=A; 4=G; 5=M; 6=Y; 7=F; 8=P; 9=D; 10=X; 11=B; 12=N; 13=J; 14=Z; 15=S; 16=Q; 17=V; 18=H; 19=L; 20=C; 21=K; 22=E

9. Determina una funció que et permeti passar un número numèric a números romans.

10. [Codewars Barcelona 2015 - nivell fàcil] El meu veí té un chihuahua molt molest que no para de molestar. M'agradaria saber quina és la seva edat humana sabent quants anys té realment. Si sabem que els dos primers anys de la seva vida conten com a

10 anys humans cadascun, i que durant la resta de la seva vida cada any que cumpleixi equivaldrà a quatre anys, si el chihuahua té 5 anys, quant equivaldria en edat humana?

11. Fes un programa que a partir d'un input amb diferents números separats per espais et determini quins dels nombres entrats són primers, sabent que un nombre primer és un número que la seva divisió dona exacte només entre si mateix i entre 1.

12. [Codewars Barcelona 2015 - nivell fàcil] El banc del teu poble, sabent que saps programar i volent-se aprofitar d'això, et demanen que els facis un programa per executar al seu caixer automàtic. La pega és que normalment només disposen de bitllets de cinc. També has de fer que per cada transacció, el banc es queda 50 cèntims adicionals. Si una transacció no és possible, no se'n fa cap. El primer input ha de ser la quantitat de diners disponibles al compte i la resta transaccions i l'output ha de ser la quantitat de diners que s'extreuen i el balanç final.

13. [Github] Fes un programa per demostrar la propietat distributiva de la suma. Han d'haver tres inputs i a l'output s'ha de veure el procés de demostració

14. En el mercat de Capicapó tenen normes d'importació molt estrictes. Una d'elles és que no permeten la importació de productes que al seu nom portin la lletra i o la lletra o ambdós. Crea un programa que et permeti saber quins productes entraran i quins no. L'output ha de ser el següent: `nomProducte és acceptat`

15. Fes una funció que et faci la mitjana de dos números introduïts fora la funció (que hagis d'escriure les variables dins el parèntesis)

16. [Github] Escriu una funció que imprimeixi el número més gran a partir de dos números introduïts fora la funció. Ara fes un altre amb tres números.

17. Defineix una funció que sumi els números introduïts i imprimeixi el resultat.

18. Fes una funció que a partir d'una array et retorni els números parells

19. A partir d'un número donat per l'usuari fes un programa que crei un diccionari on es mostri el quadrat de cada valor clau fins el valor donat. Ex:
    Input = 8 → output = {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}

20. [Codewars Barcelona 2015 - nivell fàcil] Al petit poble de Dry Gulch, a Califòrnia, trobem una reserva d'aigua sota terra amb 10.000 litres d'aigua. Donat el nombre de litres que consumeix el poble per setmana, calcula quantes setmanes podran aprofitar la reserva.
    Ex: input → seguit de números amb, com a darrer número, un 0
    2343
    2135
    5678

0

output → 2433 litres duraran x setmanes
2135 litres duraran x setmanes
5678 litres duraran x setmanes

21. Aquest programa consisteix en trobar els nombres divisibles entre 7 però no siguin divisibles per 5 entre dos números.
Input → dos números separats per espais
Output → números que compleixin les condicions entre els números introduits anteriorment en una llista.
Ex. input → 5 98 ; output → ['7', '14', '21', '28', '42', '49', '56', '63', '77', '84', '91']

22. Fes un programa que segons un text introduït et faci un recompte de quants números i quantes lletres hi han en el text.
Input → "En un lugar de La Mancha de cuyo…" (per exemple)
Output → Lletres: x - Números: y

23. Dins de la companyia del Taller de Teatre "Els Il·luminats", com a comiat en el seu darrer assaig, volen fer-li una broma al seu monitor. La broma consisteix en intercanviar-se de personalitat, és a dir, que cadascú serà una persona diferent del grup. Crea un programa que a partir dels noms assigni a cadascú una persona a la que imitar
llistaNoms = [Mireia G, Mireia A, Ainoa, Laura, Maria, Clara, Alex, Gerard, Sergi]

24. Fes un petit joc en consola que pensi un número aleatori i hagis d'intentar adivinar-lo. El usuari introduirà possibles noms i la consola ha de tornar si s'apropa, si es queda curt o es queda llarg.

25. Fes un programa que funcioni com un dau, és a dir, que hi hagi un sol input que sigui el nombre de cares del dau i que el programa retorni quin número ha sortit al tirar ficticiament el dau.

Bibliografia

La teoria ha estat escrita i estructurada per Arnau Soler i Sergi Gilabert a partir de la seva experiència amb el llenguatge i ha estat contrastada amb la documentació oficial de Python disponible a https://docs.python.org/3/search.html.

Els problemes són problemes d'exemple que els autors hem hagut de resoldre al llarg del nostre aprenentatge del llenguatge, però hi ha una selecció de problemes extreta d'altres col·leccions que estan marcats amb el lloc d'on provenen.
Si estan marcats amb el prefix [Codewars localització any - dificultat], vol dir que els hem extret del quadern d'exercicis de l'edició de l'any mencionat de la Codewars del lloc especificat. Aquests quaderns d'exercicis es poden trobar al lloc oficial de HP Codewars (http://www.hpcodewars.org/index.php?page=pastevents).
En cas de que estigui marcat com [Github], vol dir que els hem extret d'una col·lecció d'exercicis d'un usuari de Github anomenat Zhiwehu. Aquests problemes es poden trobar a https://github.com/zhiwehu/Python-programming-exercises/