VARIABLES DINS EL TEXT

```
# OLD STYLE - Tal com ho faries a C
# ... cal triar %d, %s, %f segons el tipus de la variable.
# ... quan tens { pel text, va millor que els altres (p.ex. LaTeX).
print("M'he menjat %d cocodrils." % quants)
print("En %s s'ha menjat %d cocodrils." % (qui, quants))

# PYTHON 3.0 - Llarg i poc pràctic
# ... però et permet tenir desordenades les coses si poses keywords.
print("M'he menjat {} cocodrils.".format(quants))
print("En {} s'ha menjat {} cocodrils.".format(qui, quants))
print("En {nom} s'ha menjat {n} cocodrils.".format(n=quants, nom=qui))

# PYTHON 3.6 - La bona
# ... Pots fer càlculs dins les claus, però millor fer-los abans,
# ... de manera que es pugui llegir bé el conjunt de la frase.
print(f"M'he menjat {quants} cocodrils.")
print(f"En {qui} s'ha menjat {quants} cocodrils.")
```

NO HAVER D'ESCAPAR LA BARRA \

```
print("si no vols haver d'escapar \\ les barres")
print(r"pots fer servir \ raw strings")

print(rf"...i funcionen també \ formatant {coses}")
```

AJUNTAR TEXT SENSE +

```
# Pots fer-ho normal...
print("Hola bon dia," + " com esteu tots?")
# Però de fet no cal suma entre dos cadenes de text.
print("Hola bon dia," " com esteu tots?" ' Jo molt bé.')
```

COMBINAR TEXTOS

```
llista = ["joaquim", "rinoceronts", "ermesenda"]
print("".join(llista))    # >>> "joaquimrinocerontsermesenda"
print(", ".join(llista))  # >>> "joaquim, rinoceronts, ermesenda"
```

TEXT COMENÇA AMB... / ACABA AMB...

```
if cognom.startswith("A"):
    print("Oita, la primera de la fila!")
if cognom.endswith("z"):
    print("Quin cognom més curiós...")
```

MAJÚSCULES I COSES VÀRIES

```python
print(text.upper())  # BON DIA A TOTS
print(text.lower())  # bon dia a tots
print(text.title())  # Bon Dia A Tots
```

TOTES LES COSES SÓN CERTES O FALSES

```python
num = 0
if num != 0:
    print("no cal comparar: tot el que no és zero (o buit) és cert")
if num:
    print("Tinc un número que no és zero.")
if not num:
    print("No hi ha número (tinc un zero).")

llista = []
if not llista:
    print("La llista és buida.")

text = ""
if not text:
    print("No hi ha text entre les cometes.")
```

AND I OR RETORNEN LA COSA

```python
# and / or retornen el primer element que decideix el resultat
print(f"{qui or 'Ningú'} vindrà a dinar")
#    0 or 2  -> 2     |     0 and 2 -> 0
#    1 or 2  -> 1     |     2 and 0 -> 0
#    2 and 1 -> 1     |     1 and 2 -> 2
```

...I NO MIRA EL SEGON SI EL PRIMER FALLA

```python
# només executarà la segona part si la primera és bona
if llista and llista[0] == 5:  # (mai farà index error)
   print("oita quin 5 més bonic")
```

PUC FER IF / ELSE EN UNA LÍNIA

```python
print(f"Dinarem {quants}" if quants else "No dinarem.")

# també serveix per executar coses segons condicions
fes_el_dinar(quants) if quants else None

# Compte amb les assignacions (X if Y else Z va junt SEMPRE)
a = 5 if quants else None  # vol dir “a = (X if Y else Z)”
# NO pas: “assigna tal” if Y, else “deixa-ho com estava”
```

COMPARACIÓ TERNÀRIA

```python
if 0 < x < 5:
    print("el num està entre 0 i 5")
```

EL BUCLE FÀCIL

```python
for element in llista:
    print(element)
```

...SI ET CAL L'ÍNDEX

```python
for i, element in enumerate(llista):  # et diu la posició i l'element
    print(i, element)

for i, element in enumerate(llista, start=5):  # la i serà 5, 6, 7, 8...
    print(i, element)
```

DETECTAR BREAK

```python
def primer_num_superior_a_tal(llista, tal):
    for num in llista:
        if num > tal:
            print(num)
            break
    else:  # i.e. "el for no ha fet cap break"
        print("No n'he trobat cap.")
```

...O UN RETURN SI NO HA RETURNAT

```python
def primer_num_superior_a_qual(llista, qual):
    for num in llista:
        if num > qual:
            return num
    return "No n'he trobat cap."  # si el troba no arriba aquí :)
```

FER EL BUCLE AL REVÉS

```python
def bucle_invertit(llista):
    for element in reversed(llista):
        print(element)
```

FER EL BUCLE ORDENAT

```python
def bucle_ordenat(llista):
    for num in sorted(llista):
        print(num)
```

DEIXAR COSES PENDENTS DE PROGRAMAR

```python
def func_que_em_fa_mandra_escriure():
    ...  # així guardo el "pass" per quan no ha de fer res
```

ELEMENT DINS LLISTA

```
if 2 in llista:
    print("tinc com a mínim un 2")

if "a" in text:
    print("aaaaaaaaaaaaaa")
```

GENERAR I FILTRAR LLISTES (LIST COMPREHENSIONS)

```
# [CÀLCUL for ELEMENT in LLISTA] -> llista de calculats
llista = [x**2 for x in range(5)]                # [0, 1, 4, 9, 16]
llista = [3*x + i for i, x in enumerate(llista)]  # més variables

matriu = [[1, 3], [4, 5], [3, 4]]                  # des d'una matriu
multis = [x*y for x, y in matriu]                  # [3, 20, 12]

# [CÀLCUL for ELEMENT in LLISTA if CONDICIÓ]
llista = [5*x for x in llista if x%2]          # filtro només imparells
llista = [x for x in llista if x]                  # filtro no nuls
```

TRIAR VALOR PER SI LA LLISTA QUEDÉS BUIDA

```
permesos = [1, 3, 8]
llista = [n for n in demanats if n in permesos] or [1, 3, 8]
# short-circuit: si queda buit serà fals, i em quedaré [1, 3, 8]
```

COSES GUAIS DEL MÒDUL RANDOM

```
import random

x = random.randint(3, 8)   # enter entre 3 i 8 (inclosos)

y = random.choice(llista)  # tria un element de la llista a l'atzar

# random.choices(llista, proporcions)
z = random.choices([0, 2], [1, 5])  # El 2 sortirà 5x més sovint que el 0
a, b = random.choices([0, 2], [1, 5], k=2)  # treu-me'n dos de cop

random.shuffle(llista)     # barreja la llista a l'atzar

random.sample(llista, 2)   # agafa de la llista dos elements a l'atzar
```

...I UN PARELL DE TRUCS

```
if not random.randint(0, 5):  # una de cada 6 vegades (quan triï el 0)
    print("quina sort, nen")

def moneda():  # així faig "if moneda:" quan vull fer cara o creu
    return bool(random.getrandbits(1))
```

DESACTIVAR CODI CONVERTINT-LO EN TEXT

```
for x in llista:
    print("estic fent coses que es faran")

"""
for x in llista:
    print("aquesta part està desactivada")
"""
```

NOM ALTERNATIU PER LA FUNCIÓ

```
def fes_una_suma_molt_xula(a, b):
    return a+b

s = fes_una_suma_molt_xula  # sense parèntesis vol dir la funció en sí

s(1, 3)
```

EXPLICAR QUÈ FA LA FUNCIÓ

```
def funcio(num, llista):
    """Fa coses amb la llista i els nums i tal (aquestes explicacions
    els IDE les veuen i les fan servir per explicar-te què fan les
    funcions sense que hagis d'anar a veure el codi tu mateix)

    :param num: explicació del num
    :param llista: explicació de la llista
    :return: explicació del resultat
    """
    return coses_que_ha_calculat
```

VALOR PER SI NO TROBO LA CLAU D'UN DICCIONARI

```
key = "pernils"
dict = {"hola": "Com estàs", "barcelona": "no m'enamora"}
valor = dict.get(key, "No hi era")  # si no troba, retorna "No hi era"
```

LLISTES QUE NO OCUPEN MEMÒRIA (GENERADORS)

```
# crea la llista abans de començar (ocupant molta memòria)
for x in [num for num in range(100000)]:
   print(x)

# calcula cada num just quan li cal (i l'oblida just després)
for x in (num for num in range(100000)):
   print(x)
```

NOM ALTERNATIU PER UN MÒDUL

```
import numpy as np
```

VARIABLE QUE NO FARÉ SERVIR

```python
for _ in range(5):  # s'hi val, i així marco que no afecta en res
    print("Hola")
```

FUNCIÓ QUE INVENTA LLISTES QUE NO OCUPEN (YIELD)

```python
# la funció no fa el que té escrit a dins, sinó que li encarrega
# de fer-ho a algú altre (perquè té yield en lloc de return).
# i.e. quan la crido, el que fa és retornar una còpia de la recepta.
def diu_si_3_cops_i_despres_forever_no():
    for _ in range(3):
        yield "Sí!"  # yield és com return però només pausa (no tanca)
    while True:
        yield "No!"

# li entrego una còpia de la recepta a la variable "respostes"
respostes = diu_si_3_cops_i_despres_forever_no()
# ...ara "respostes" sap què ha de fer (però de moment no ho fa)

# Cada cop que demani "next", "respostes" seguirà la recepta fins que
# arribi a un yield. Llavors em retornarà el que trobi al yield,
# i s'esperarà allà fins que jo n'hi demani un altre.
for _ in range(25):
    print(next(respostes))

# Com que la recepta és infinita (while True), jo podria anar-li
# dient "next()" per sempre (i ella m'aniria dient "No!").
# Per contra, si la fes finita, podria fer "for _ in respostes".

# Ah, i podria entregar més d'una còpia de la recepta
# (a diferents variables), i cadascuna recorreria les instruccions
# al seu ritme, a mesura que li demanessin "next" a ella.
```

IF TOTS SÓN CERT / IF ALGUN ÉS CERT

```python
noms = ["Joaquima", "Ermessenda", "Zalutacions"]

if all("a" in nom.lower() for nom in noms):
    print("Tots els noms tenen la lletra 'a'.")
    # all(True, True, True) -> True

if any("z" in nom.lower() for nom in noms):
    print("Hi ha algun nom amb la lletra 'z'.")
    # any(False, False, True) -> True
```

SUBSTITUIR PARTS DEL TEXT

```python
text = "Vaig a contractar el Joaquim"
text = text.replace("Joaquim", "Pernil")  # Vaig a contractar el Pernil
text = text.replace("a", "i")  # Viig i contrictir el Pernil
```

SEPARADOR DE MILERS

```
numero_llarg = 12_345_678  # puc posar _ dins el número i funciona igual
multiplicat = 50_000 * 4_000  # 200000000 (i.e. 200_000_000)
```

ESTALVIAR AUXILIARS (ITERABLE PACKING / UNPACKING)

```
llista = [5, 4]
x, y = llista    # assignar vàries coses alhora (x=5, y=4)
x, y = 3, 2      # versió directa, sense llista (x=3, y=2)
x, y = y, x      # intercanviar sense auxiliar  (x=2, y=3)
x, y = y, x+y    # actualitzar sense auxiliar   (x=3, y=5)
```

AGAFAR “LA RESTA” (EXTENDED UNPACKING)

```
llista = [1, 2, 3, 4, 5]
primer, segon, *altres = llista  # primer=1, segon=2, altres=[3, 4, 5]
primer, *altres, ultim = llista  # primer=1, altres=[2, 3, 4], ultim=5
a, *b, c, d = llista  # a=1, b=[2, 3], c=4, d=5
a, *_, c, d = llista  # agafo els que vull i ignoro “la resta”
```

...GENERAR UNS QUANTS ALEATORIS DE COP

```
x, y, z = (random.randint(0, 10) for _ in range(3))
# faig un aleatori entre 0 i 10, tres vegades
```

ASSIGNACIÓ MÚLTIPLE

```
a = b = c = 1  # a=1, b=1, c=1
```

AGAFAR COSES PEL FINAL

```
llista = ["Pernil", "Joaquim", "Ermessenda"]
print(llista[-1])  # Ermessenda
```

TROBAR POSICIÓ D’UNA COSA

```
llista = [5, 2, 3, 1, 5, 8]
posicio_del_tres = llista.index(3)          # 2
posicio_de_la_i = "Ornitorrinc".index("i")  # 3 (et diu la primera)
```

AGAFAR I ESBORRAR L’ÚLTIM

```
llista = [2, 8, 4, 3]
print(llista.pop())  # 3 (...i ara la llista és [2, 8, 4])
```

...O QUALSEVOL ALTRE

```
llista = [2, 8, 4, 3]
print(llista.pop(1))  # 8 (...i ara la llista és [2, 4, 3])
```

EXTENDRE LLISTES

```
llista = [1, 2, 3, 4, 5]
extra = [6, 7, 8, 9]
llista += extra  # llista=[1, 2, 3, 4, 5, 6, 7, 8, 9]

llista = [1, 2, 3, 4, 5]
extra = (6, 7, 8, 9)  # si “extra” és un altre tipus d’iterable...
llista.extend(extra)  # ...podem fer servir .extend()

llista = [1, 2, 3]
llista.extend(range(4, 6))  # llista=[1, 2, 3, 4, 5]
```

RESULTAT ANTERIOR (CONSOLA PYTHON)

```
>>> 3*2
6

>>> 5*_  # La barra guarda el resultat (com l’ANS de la calculadora)
30
```

IMPRIMIR VARIABLE I NOM ALHORA (python 3.8)

```
num = 42
print(f"{num=}")  # >>> 'num=42'
```

...O UN CÀLCUL I EL RESULTAT

```
print(f"{6*7=}")  # >>> '6*7=42'
```

ORDENAR LLISTES (AL LLOC)

```
llista.sort()
llista.sort(reverse=True)  # si la vull invertida
llista.sort(key=len)  # puc demanar-li criteris alternatius per ordenar
llista.sort(reverse=True, key=len)  # de més llarg a més curt
```

RETORNAR DIVERSOS VALORS ALHORA

```
return edat, color_preferit  # ...i els reculls amb tuple unpacking
```

FORMATS ESPECIALS

```
print(f"{numero:.2f}")  # escriu-me només 2 decimals
print(f"{0.025:.2%}")  # fes-me el percentatge: >>> '2.50%'

print(f"{nom:>10}")  # omple amb espais fins que ocupi 10: >>> '     david'
print(f"{nom:*<8}")  # asteriscs fins a 8 (text a l’esquerra): >>> 'david***'
print(f"{nom:=^9}")  # iguals fins a 9 (text al centre): >>> '==david=='
```

FORMATAR DATES

```python
import datetime
data = datetime.datetime.now()

print(f"{data:%Y-%m-%d %H:%M}")  # >>> 2022-08-28 15:32
# Podeu veure més %codis (nom del dia/mes, AM/PM, etc) aquí
```

VARIABLES DINS EL FORMAT

```python
quants_decimals = 2
print(f"{numero:.{quants_decimals}f}")
```

BUCLES AMB DICCIONARIS

```python
diccionari = {clau1: "valor1", clau2: "valor2"}

for clau in diccionari:  # per defecte ell agafa les claus
    print(clau)

for valor in diccionari.values():  # puc agafar els valors
    print(valor)

for clau, valor in diccionari.items():  # o agafar-ho tot per parelles
    print(f"El valor de {clau} és {valor}.")
```

COORDINAR LLISTES (ZIP)

```python
gent = ["Mar", "Joan", "Lee", "Nastúrcia"]
premis = ["Or", "Plata", "Bronze"]

for persona, premi in zip(gent, premis):  # (la més llarga quedarà escapçada)
    print(f"{persona} ha guanyat {premi}.")

# És un iterador. Si vols guardar el resultat pots convertir-lo en llista:
resultats = list(zip(gent, premis))
# >>> [(Mar, Or), (Joan, Plata), (Lee, Bronze)]
```

TAULA DE VALORS (MAP)

```python
valors_de_x = [2, 3, 5, -1, 14, 42]

def f(x):
    return x + 100

for y in map(f, valors_de_x):  # aplica la funció a cada número
    print(y)

# També és un iterador. Podem guardar la llista convertint-la:
valors_de_y = list(map(f, valors_de_x))  # [102, 103, 105, 99, 114, 142]
punts = list(zip(valors_de_x, valors_de_y))  # [(2, 102), (3, 103), ...]
```

FUNCIONS ANÒNIMES (LAMBDA)

```
# En lloc de definir la f(x) amb antelació, la puc fer sobre la marxa:
··· lambda x: x-100 ···   # f(x)=x-100

# Útil en llocs com la taula de valors, on f(x) és curta i no li cal nom:
valors_de_x = [1, 2, 3, 4, 5]
for y in map(lambda x: x-100, valors_de_x):  # f(x) = x-100
    print(y)
```

PITÀGORES FÀCIL

```
from math import hypot
hypot(3, 4)  # 5
```

S'ASSEMBLEN DOS NÚMEROS?

```
from math import isclose

# Genèric (i.e. error relatiu màxim 1e-9, error absolut màxim 0.0)
a, b = 10.34444444, 10.34444445
if isclose(a, b):
    print("Si fa o no fa són el mateix número.")

# Triant error absolut màxim
a, b = 150, 153
if isclose(a, b, abs_tol=3):
    print("Estan a màxim 3 de distància.")

# Triant error relatiu màxim
a, b = 100, 97
if isclose(a, b, rel_tol=0.04):
    print("La diferència és menys del 4%.")
```

INFINIT

```
from math import inf

for x in range(100):
    if x**10 < inf:
        print(x**10, "No pots superar l'infinit MUAHAHA")

# Va bé com a valor inicial per buscar màxims i mínims:
maxim = -inf
for x in opcions:
    maxim = max(maxim, x)
```

CODI D'UNA LLETRA / LLETRA D'UN CODI

```
# Les lletres tenen un codi numèric (ASCII): A=65...Z=90 | a=97...z=122
print(f"El codi de la C és {ord("C")}")     # El codi de la C és 67
print(f"La lletra número 85 és {chr(85)}")  # La lletra número 85 és U
```

DETECTAR ESTRUCTURES (PATTERN MATCHING) (python 3.10)

```
tasques = ["menja poma", "dorm", "mira amunt", "fuig"]

for tasca in tasques:
   match tasca.split():  # Retallo pels espais: ["menja", "poma"]
        # if la tasca (retallada) és "dorm":
        case ["dorm"]:
            print("Me'n vaig a dormir una estona :)")
        # elif la tasca comença amb "menja" i té una segona cosa al darrere:
        # (...a la cosa que vingui al darrere li diré aliment)
        case ["menja", aliment]:
            print(f"Menjo {aliment}")
        # elif la tasca és "fuig":
        case ["fuig"]:
            print("Campi qui pugui!")
        # else:
        case _:
            print("Sé pas què vols que faci")

# en tots aquests casos (case), els claudàtors només hi són per agrupar
# les coses i que sigui fàcil d'entendre visualment. Podrien ser parèntesis
# o comes normals (tuples) i no passaria res.
```

...ESTRUCTURES DE MIDA VARIABLE

```
tasques = ["menja poma", "menja poma pizza", "dorm"]

for tasca in tasques:
    match tasca.split():
        # puc englobar qualsevol quantitat d'aliments amb unpacking
        case ["menja", *aliments]:
            print("Menjo vàries coses:")
            for aliment in aliments:
                print(f"- {aliment}")
        # aquí li he posat nom a l'acció de l'else, per dir-la al print
        case altre:
            print(f"No sé què vol dir {altre}")
```

...COMBINANT SINÒNIMS

```
tasques = ["mira amunt", "guaita avall", "dina pizza", "pizza per dinar"]

for tasca in tasques:
    match tasca.split():
        # "mira" o "guaita", i després una altra cosa que es dirà direcció
        case ["mira" | "guaita", direccio]:
            print(f"Observant {direccio}")
        # "dina" i una altra cosa, o bé una cosa i després "per" i "dinar"
        case ["dina", aliment] | [aliment, "per", "dinar"]:
            print(f"Dinarem {aliment}")
```

...POSAR NOM ALS SINÒNIMS

```
tasques = ["menja flam", "menja pizza", "menja gelat", "menja cols"]

for tasca in tasques:
    match tasca.split():
        # em serveix "flam" i també "gelat", però vull saber quin era.
        case ["menja", ("flam" | "gelat") as postre]:
            print(f"Menjo un postre que és {postre}.")
        # si menjo una altra cosa (pizza, col...) farà aquest de sota
        case ["menja", aliment]:
            print(f"Menjo {aliment}.")
```

...CONDICIONS EXTRA

```
menu = ["flam", "pizza", "gelat"]
tasques = ["menja flam", "menja pizza", "menja gelat", "menja cols"]

for tasca in tasques:
    match tasca.split():
        # Ha de ser ["menja", cosa] però a més a més ha d'estar al menú.
        case ["menja", aliment] if aliment in menu:
            print(f"Per dinar voldré {aliment}.")
        # Si l'aliment ha fallat a dalt, aquí sí que ho agafarà.
        case ["menja", _]:
            print(f"El que volia menjar no surt al menú... :(")
```

OMPLIR DICCIONARI PER DEFECTE

```
vots = dict.fromkeys(["Joaquim", "Miquela", "Hermínia"], 0)
# vots = {"Joaquim": 0, "Miquela": 0, "Hermínia": 0}
```