Introduction à Python

```
# (dans un programme en python, les lignes commençant par "#" sont ignorées)

########################################
# Les nombres (types 'int' et 'float'):

x = 5 + (2 * 3) - (12 / 3)
x = x ** 2  # x puissance 2
y = 23 % 10  # 23 modulo 10 vaut 3 (c'est le reste de la division)
print("x vaut", x)  # affiche: x vaut 49.0

print(type(x))  # affiche: <class 'float'>
x = int(x)  # conversion en int (nombre entier)
print("x vaut", x)  # affiche: x vaut 49
print(type(x))  # affiche: <class 'int'>


########################################
# utilisation du module 'math':

import math
y = math.sqrt(x)  # racine carree
z = math.cos(x)  # cosinus
# documentation du module math: https://docs.python.org/3/library/math.html


########################################
# déclaration de fonctions:


def carre(x):
    return x ** 2


def puissance(x, y):
    return x ** y


carre_de_trois = carre(3)


########################################
# Le type bool (True / False)
age = 10
est_majeur = age >= 18
print(type(est_majeur))  # affiche <class 'bool'>
print(est_majeur)  # affiche False

# Opérateurs de comparaisons:
# < : plus petit que. Exemple: "2 < 4" vaut True
# > : plus grand que
# == : égal
# >= : supérieur ou égal
# <= : inférieur ou égal
# != : différent
# in : appartenance à un itérable. Exemple: "42 in [41, 42, 43]" vaut True
# not : inverse un bool. Exemple: "not True" vaut False
```

```
########################################
# Les conditions (mots-clefs "if", "elif", "else")

a = 10
b = 12
if a > b:
    print(a, "est plus grand que", b)
elif a < b:
    print(b, "est plus grand que", a)  # <-----
else:
    print("égalité")


if False:
    print("tic")
else:
    print("tac")  # <-----


if a >= b and b >= a:
    print("égalité")


if a > b or a < b:
    print("différents")


########################################
# Le type str (représente du texte / "chaine" de caractère ("string"))

x = "hello"
y = "world"
z = x + " " + y  # concaténation
print(z)  # affiche "hello world"
print("na" * 4)  # affiche "nananana"

x = "12"
print(type(x))  # <class 'str'>
x = int(x)  # conversion en int
print(type(x))  # <class 'int'>

# On ne peut pas additionner des int des des str car ils ne sont pas du même type
# "plop" + 4  -> provoquerait une erreur (TypeError)
# Si on veut obtenir la chaine "plop4" par concaténation, il faut donc convertir le
int en str:
print("plop" + str(4))  # affiche "plop4"

# La fonction input permet de lire une chaine écrite par l'utilisateur:
chaine = input("Écrivez qqch: ")
print("Vous avez écrit:", chaine)

s = "hello"
print(s[0])  # affiche "h"
print(s[1])  # affiche "e"
print(s[-1])  # affiche "o"
print(s[3:])  # affiche "lo"
print(s[1:-1])  # affiche "ell"

taille = len(s)  # taille vaut 5

print(str.upper(s))  # "HELLO"
print(s.upper())  # même chose, en utilisant la méthode upper directement attachée
à s
s = "abracadabra"
```

```
print(s.replace('a', 'o'))  # "obrocodobro"

# documentation des autres méthodes disponibles pour le type string:
# https://docs.python.org/3/library/stdtypes.html#string-methods


########################################
# Le type list

l = []  # une liste vide
l = [1, 2, 3, "etc."]  # une liste de 4 éléments
print(l[0] == 1)  # affiche True
print(42 in l)  # False
l.append(42)  # ajout d'un élément à la fin de la liste
print(l)  # affiche [1, 2, 3, 'etc.', 42]
del l[-1]  # suppression du dernier élément de la liste
sublist = l[:-1]  # comme pour les str on a la syntaxe [debut:fin]
print(sublist == [1, 2, 3])  # True
print(len(sublist))  # 3

numbers = [1, 2, 3] + [4, 5, 'etc.']  # concaténation de listes

message = "hello world"
print(list(message))  # ['h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd']
# On peut splitter une string en une liste de mots:
print(message.split())  # ['hello', 'world']
print(message.split('l'))  # ['he', '', 'o wor', 'd']
# À l'inverse on peut aussi réassembler une liste de mots en une string:
mots = ['hey', 'you', '!!!']
phrase = " ".join(mots)
print(phrase)  # "hey you !!!"

# bonus: la fonction "sorted" pour trier dans l'ordre:
unordered = [2, 4, 5, 1, 3, 0]
ordered = sorted(unordered)
print(ordered == [0, 1, 2, 3, 4, 5])  # affiche True
# documentation de la fonction sorted:
https://docs.python.org/3/library/functions.html#sorted


########################################
# Les boucles (for / while)

for mot in mots:
    # il y a 3 mots, on va donc passer 3 fois dans cette boucle
    print(mot.upper())

# Affichera:
# HEY
# YOU
# !!!

x = None
while x is None or x < 100:
    x = input('Entrez un nombre supérieur à 100: ')
    x = int(x)


x = 10
while True:
    print(f"Encore {x} tours restants")
    x = x - 1
    if x == 0:
        break  # le mot-clef break met fin à une boucle
```

```
# La fonction range() génère une séquence de nombres
for i in range(3):
    print(i)  # affichera: 0 puis 1 puis 2


# En revanche la fonction range ne renvoie pas une liste:
suite = range(100)
print(isinstance(suite, list))  # affiche False
# En effet la fonction range génère ses nombres un par un,
# lorsque l'on itère dessus (avec une boucle for par exemple)
# pour éviter de saturer la RAM.
# Si on veut une vraie list, il suffit de convertir:
suite = list(suite)


########################################
# List comprehensions

liste = [4, 8, 15, 16, 23, 42]
nombres_pairs = [x for x in liste if x % 2 == 0]
carres = [nombre ** 2 for nombre in liste]


########################################
# Les itérables

# Le point commun entre le type str, list, range,
# est qu'ils sont itérables, c'est à dire qu'on peut
# parcourir leurs éléments à l'aide d'une boucle for par exemple.

import typing


def is_this_iterable(var):
    if isinstance(var, typing.Iterable):
        return f"{var} est itérable"
    else:
        return f"{var} n'est pas itérable"


for x in [42, 'hey', [], range(5), None, True]:
    print(is_this_iterable(x))

# Affichera:
#    42 n'est pas itérable
#    hey est itérable
#    [] est itérable
#    range(0, 5) est itérable
#    None n'est pas itérable
#    True n'est pas itérable


########################################
# Lire des fichiers sur le disque

filename = "intro_python.py"
file_content = open(filename).read()
count = len(file_content)
print(f"Ce fichier contient {count} caractères.")


########################################
# le type 'dict' (= tableau associatif)
# Permet d'associer des clefs à des valeurs
```

```
d = {}  # un dict vide
scores = {"alice": 50, 'bob': 0, 'charlie': 49, 'ines': 0}
print(len(scores))  # affiche 4

score_charlie = scores['charlie']  # 49
del scores['ines']  # on enlève cette clef du dict
print(len(scores))  # affiche 3

scores['saskia'] = 100  # ajout d'une clef "saskia" associée à la valeur 100

# affichons les scores:
for key in scores:
    print(key, "a le score", scores[key])


# équivalent:
for key in scores.keys():
    print(key, "a le score", scores[key])


# équivalent:
for key, value in scores.items():
    print(key, "a le score", value)


# En pratique on manipule souvent des données plus complexes, telles que
# des dict qui contiennent d'autres dicts,
# qui contiennent des listes, etc:
paris_weather_data = {
    "coord": {
        "lon": 2.35,
        "lat": 48.86
    },
    "weather": [
        {
            "id": 800,
            "main": "Clear",
            "description": "clear sky",
            "icon": "01d"
        }
    ],
    "main": {
        "temp": 292.57,
        "pressure": 1015,
        "humidity": 48,
        "temp_min": 290.15,
        "temp_max": 294.26
    },
    "dt": 1569058553,
    "timezone": 7200,
    "name": "Paris",
}

temperatures = {
    "paris": paris_weather_data["main"]["temp"]
}


########################################
# Une fonction un peu plus utile: récupérer la météo
# (utilise le service gratuit https://openweathermap.org/)
# Exemple: météo à paris:
# https://api.openweathermap.org/data/2.5/weather?
APPID=515b9c16560819dfe610251459c619d7&q=Paris
```

```
from urllib.request import urlopen  # pour récupérer le contenu d'une page web
import json  # pour convertir du texte au format JSON en dict


OWM_APIKEY = "515b9c16560819dfe610251459c619d7"


def get_weather_in_city(ville):
    # Cette fonction prend en paramètre un nom de ville, et renvoie la météo
associée
    url = "http://api.openweathermap.org/data/2.5/weather?APPID=" + OWM_APIKEY
    url = url + "&q=" + ville
    response = urlopen(url)  # de type httpresponse
    contenu = response.read()  # de type bytes
    contenu = contenu.decode()  # de type str
    data = json.loads(contenu)  # de type dict
    weather = data["weather"][0]["description"]
    return weather


weather_of_cities = {}

for city in ['Paris', 'London', 'Moscow']:
    meteo = get_weather_in_city(city)
    weather_of_cities[city] = meteo

print(weather_of_cities)

for city_name, weather in weather_of_cities.items():
    s = f"Weather in {city_name} is {weather}"
    print(s)
```