Algorithmique et programmation

Master 1 Humanités numériques

2023-2024

Jérôme Darmont

https://eric.univ-lyon2.fr/jdarmont/

Actualité du cours

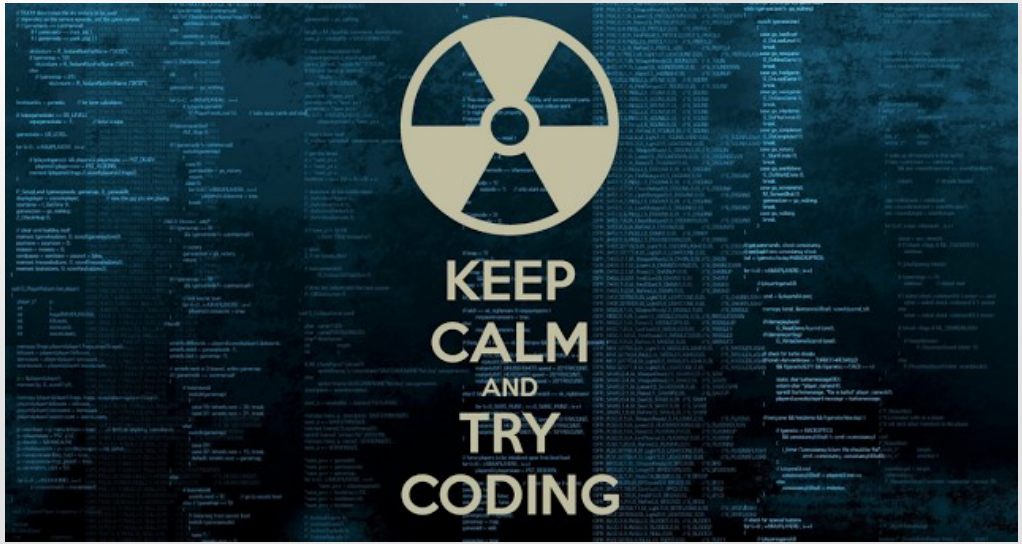
https://eric.univ-lyon2.fr/jdarmont/?page_id=3135

https://eric.univ-lyon2.fr/jdarmont/?feed=rss2

https://social.sciences.re/@darmont #hnprog

Objectifs du cours

- Découvrir les bases de la programmation informatique...



warwithpc.blogspot.fr

- ...qui seront réutilisées en M2

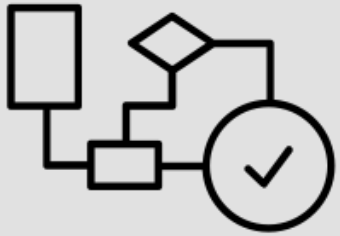
(fouille de données/de textes)

Qu’est-ce qu’un algorithme ?

mirfaces.com

Al-Khwârizmî
(780-850)

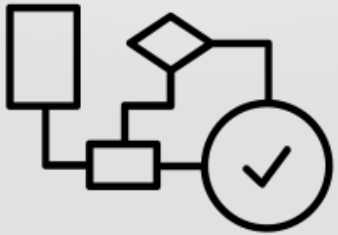
Problème



Données

Suite d’opérations

Exemple d'algorithme

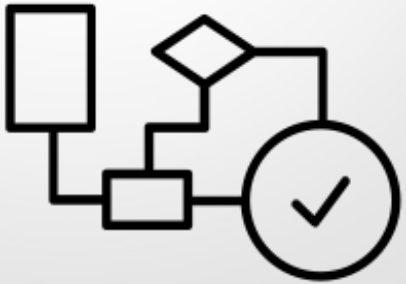


1
2
3
4

étapes/opérations

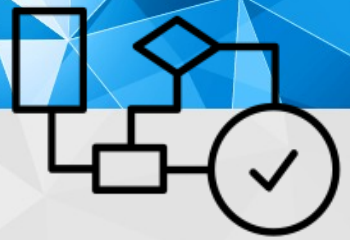
Résultat

Qu’est-ce qu’un programme ?



Traduction d’un algorithme en un langage interprétable par un ordinateur

Outil 1 : Langage algorithmique



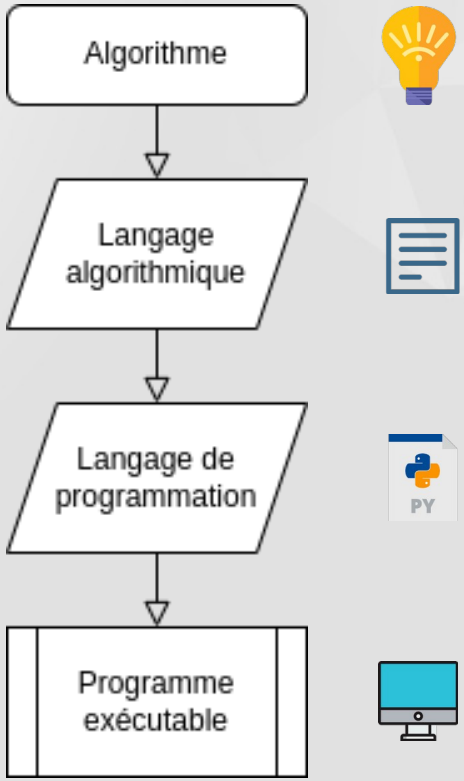
- Indépendant de tout langage de programmation
- En français
- Rigoureux

Outil 2 : Langage de programmation

- Fonctionne sur tous les systèmes d'exploitation
- Très utilisé pour apprendre la programmation
- Dispose de nombreuses bibliothèques
  (notamment traitements de document textuels, *machine learning*…)

Méthodologie

Et second exemple d'algorithme !



Plan du cours

1. Stocker et manipuler des données en mémoire
2. Structures alternatives
3. Structures itératives
4. Sous-programmes
5. Structures de données avancées
6. Stocker des données persistantes

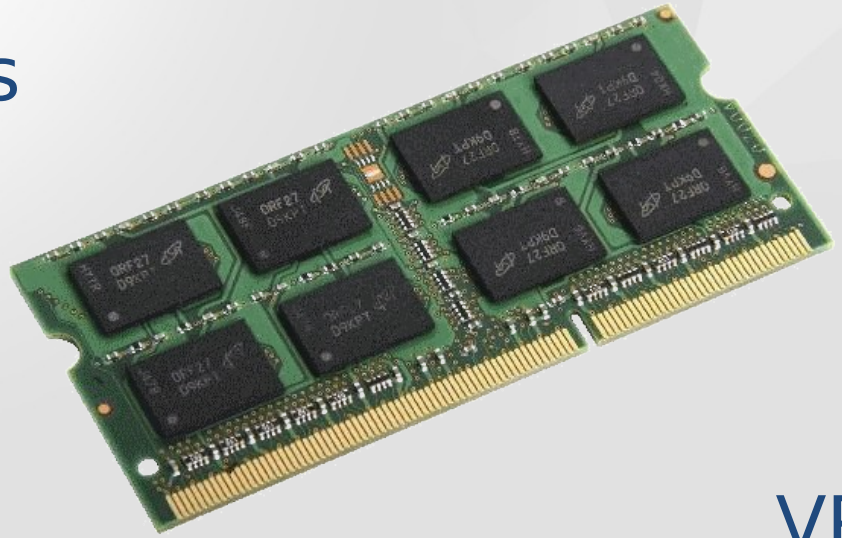
Partie 1

Stocker et manipuler des données en mémoire

Stocker des données dans la mémoire d’un ordinateur

44 ans

1466,62 €



Jérôme

VRAI

Deux types de données à déclarer

Variables
Var âge : Entier
Var salaire : Réel
Var nom : Chaîne
Var v_f : Booléen

Constantes
Const N ← 10
Const PI ← 3,1416
Const MONNAIE ← "Euro"

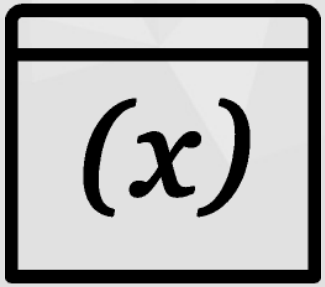
(Type de données)

Pas de déclaration dans Python

Trois étapes

- Déclarer une variable ou constante
  Réserver sa place dans la mémoire

- Initialiser une variable
  Lui donner une valeur



- Utiliser une variable ou constante
  Utiliser ou modifier sa valeur

Initialisation de variables

Algorithmique	Python
âge ← 44 salaire ← 1466,62 nom ← "Jérôme"	age = 44 Salaire = 1466.62 nom = "Jérôme" nom = 'Jérôme'
v_f ← Vrai {Contraire : Faux}	v_f = True # Contraire : False
	a = b = 20 x, y = 30, 40

En vert : commentaires (non exécutés)

Connaître le type d’une variable

```
type(age)        # int (entier)

type(salaire)    # float (réel)

type(nom)        # str (chaîne)

type(v_f)        # bool (booléen)
```

Calculs

Algorithmique	Python
âge ← 44 + 1 âge ← âge + 1 salaire ← salaire x 1,05 quotient ← a ÷ b nom ← nom | " Darmont"	age = 44 + 1 age = age + 1 age += 1 salaire = salaire * 1.05 quotient = a / b nom = nom + " Darmont"

Opérateurs arithmétiques	
+ - x ÷ ^ div mod	+ - * / ** // %
^ ** : exposant	mod % : modulo (reste de division entière)

Entrées/sorties (1/3)

Entrées

Saisir

Sorties

Afficher

Entrées/sorties (2/3)

Algorithmique	Python
Lire(nom) Lire(âge) Lire("Salaire :", salaire) Écrire("Hello world!") Écrire(âge)	nom = input() age = int(input()) salaire = float(input("Salaire : ")) print("Hello world!") print(age)

Entrées/sorties (3/3)

Algorithmique	Python
Écrire("Âge ", âge, "Salaire ", salaire)	print("Âge", age, "Salaire", salaire)

Structure d'un algorithme/programme

Algorithmique

```
Algorithme Exemple
   {Déclaration des constantes}
   Const MONNAIE ← "Euros"
   {Déclaration des variables}
   Var salaire : Réel
   Var nom : Chaîne
Début
   Lire("Nom :", nom)
   Lire("Salaire :", salaire)
   salaire ← salaire × 1,1
   Écrire("Nouveau salaire de ", nom,
                  " : ", salaire, " ", MONNAIE)
Fin
```

Suite d'instructions exécutées dans l'ordre

Python

```
# Aucune déclaration de variable
# (c'est moche)

# La constante... est une variable
# (initialisée)
MONNAIE = "Euros"

nom = input("Nom :")
salaire = float(input("Salaire : "))
salaire = salaire * 1.1
print("Nouveau salaire de ", nom,
          " : ", salaire, " ", MONNAIE)
```

Exemple

- Calculer le prix TTC d’un article quelconque sachant son prix HT et le taux de TVA (fixé à 20 %).
- Formuler la solution en langage algorithmique, puis en Python.
- Indication : déterminer les variables, les entrées et les sorties.

Exemple - Algorithmique

Algorithme prixTTC

Const TVA ← 0,2
Var prixHT, prixTTC : Réels

Début

Lire("Prix HT :", prixHT)	{Entrée}
prixTTC ← prixHT × (1 + TVA)	{Traitement/calcul}
Écrire("Prix TTC :", prixTTC)	{Sortie}

Fin

Partie 2

Structures alternatives (tests)

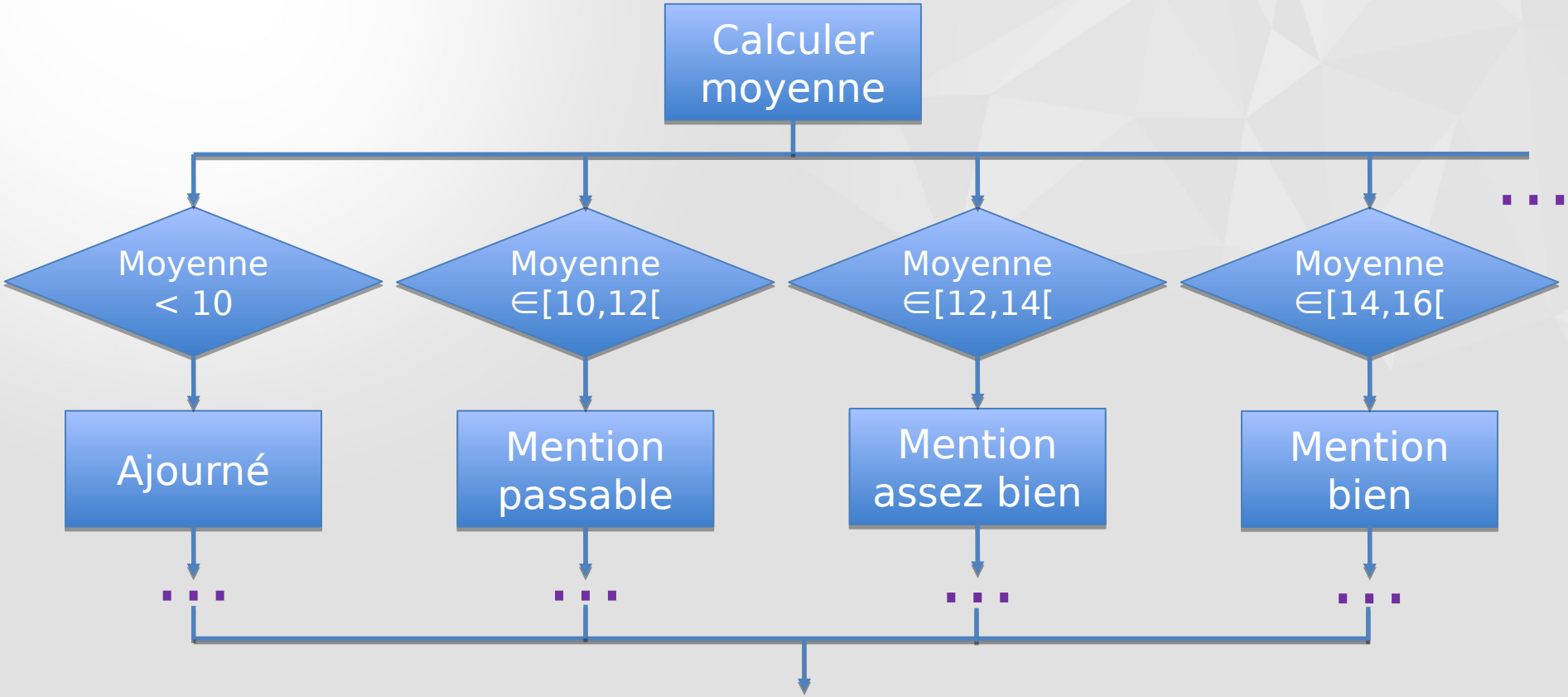
Qu’est-ce qu’un test ?

Expression d’un choix

- Ex. Si le mot de passe est correct, alors
  l’utilisateur·trice peut se connecter.

- Ex. Si la moyenne générale est supérieure ou égale à 10, alors
  l’étudiant·e valide son année.
  Sinon,
  l’étudiant·e échoue.

Exemple


Calculer moyenne
Moyenne < 10
Moyenne ∈[10,12[
Moyenne ∈[12,14[
Moyenne ∈[14,16[
Ajourné
Mention passable
Mention assez bien
Mention bien

Types de tests (1/2)

Si condition alors action

Si condition alors action1
Sinon action2

Types de tests (2/2)

Si condition1 alors action1
Sinon si condition2 alors action2

Si condition1 alors action1
Sinon si condition2 alors action2
Sinon action3

Les « Sinon si » peuvent se répéter.

Écriture des tests (1/2)

Algorithmique	Python

Si nom = "Jérôme" alors
  Écrire("Bravo")
Fin si
 	
Si âge < 18 alors
  Écrire("Mineur")
Sinon
  Écrire("Majeur")
Fin si
 	
Si salaire < 1466,62 alors
  Écrire("Pauvre")
Sinon si salaire ≤ 3000 alors
  Écrire("Moyen")
Sinon
  Écrire("Riche")
Fin si
	if nom == "Jérôme":
	  print("Bravo")
	
	if age < 18:
	  print("Mineur")
	else:
	  print("Majeur")
	
	if salaire < 1466.62:
	  print("Pauvre")
	elif salaire <= 3000:
	  print("Moyen")
	else:
	  print("Riche")

Écriture des tests (2/2)

Algorithmique	Python

Si âge < 18 alors
  Si salaire > 0 alors
  salarie_mineur ← Vrai
  Sinon
  salarie_mineur ← Faux
  Fin si
Fin si
	if age < 18:
	  if salaire > 0:
	  salarie_mineur = True
	  else:
	  salarie_mineur = False

Si âge ≥ 25 et salaire = 0 alors
  Écrire("Droit au RSA")
Fin si
	if age >=25 and salaire == 0:
	  print("Droit au RSA")

Opérateurs utilisés dans les tests

Opérateurs de comparaison	
Algorithmique	**Python**
= ≠ < ≤ > ≥	== != < <= > >=

Opérateurs logiques	
Algorithmique	**Python**
et ou non	and or not

Tables de vérité (1/3)

- Soient deux variables booléennes C_1 et C_2.
  - Par exemple : $C_1 \leftarrow (\text{âge} \geq 25)$
    $C_2 \leftarrow (\text{salaire} = 0)$

âge	22	25	27
C_1	Faux	Vrai	Vrai

salaire	0	1000	2000
C_2	Vrai	Faux	Faux

Tables de vérité (2/3)

- Quel est le résultat des tests : Si **non C_1**
  Si **C_1 et C_2**
  Si **C_1 ou C_2** ?

C_1	C_2	non C_1	C_1 et C_2	C_1 ou C_2
Vrai	Vrai	Faux	Vrai	Vrai
Vrai	Faux	Faux	Faux	Vrai
Faux	Vrai	Vrai	Faux	Vrai
Faux	Faux	Vrai	Faux	Faux

Tables de vérité (3/3)

(âge ≥ 25 et salaire ≠ 0) ou
(âge < 25 et salaire = 0)

- Soit la variable booléenne
  $C_3 \leftarrow (C_1$ et (non C_2)) ou ((non C_1) et C_2).
- Quelles sont les valeurs possibles de C_3 en fonction de celles de C_1 et C_2 ?

C_1	C_2	non C_2	C_1 et (non C_2)	non C_1	(non C_1) et C_2	C_3
Vrai	Vrai	Faux	**Faux**	Faux	**Faux**	**Faux**
Vrai	Faux	Vrai	**Vrai**	Faux	**Faux**	**Vrai**
Faux	Vrai	Faux	**Faux**	Vrai	**Vrai**	**Vrai**
Faux	Faux	Vrai	**Faux**	vrai	**Faux**	**Faux**

Exemple

- Définir des constantes login et mot de passe.
- Faire saisir à un·e utilisateur·trice un login et un mot de passe.
- Si le login et le mot de passe saisis correspondent aux constantes, afficher « connexion réussie », sinon afficher « erreur d’authentification ».
- Formuler la solution en langage algorithmique, puis en Python.

Exemple - Algorithmique

Algorithme motPasse

Const LOGIN ← "jerome"

Const MDP ← "p@p@oute"

Var loginSaisi, mdpSaisi : Chaînes

Début

Lire("Login :", loginSaisi)

Lire("Mot de passe :", mdpSaisi)

Si loginSaisi = LOGIN et mdpSaisi = MDP alors

Écrire("connexion réussie")

Sinon

Écrire("erreur d'authentifiation")

Fin si

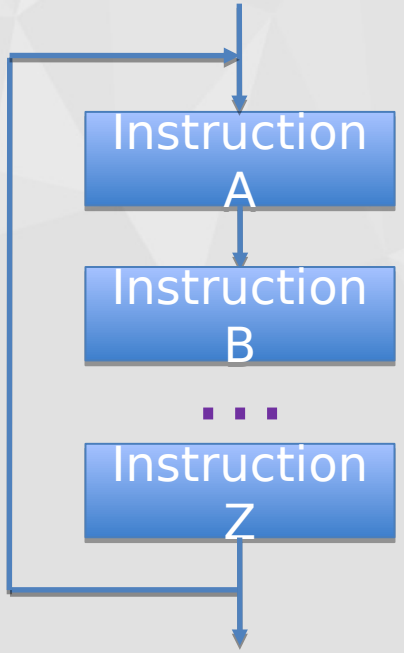
Fin

Partie 3

Structures itératives (boucles)

Qu’est-ce qu’une boucle ?

- Répétition d’un ensemble d’instructions
  - Ex. Répéter N fois le même traitement (plutôt que copier/coller N fois les instructions)
  - Ex. Saisir les notes de tou·tes les étudiant·es d’une promotion
  - Ex. Augmenter tou·tes les salarié·es d’une entreprise

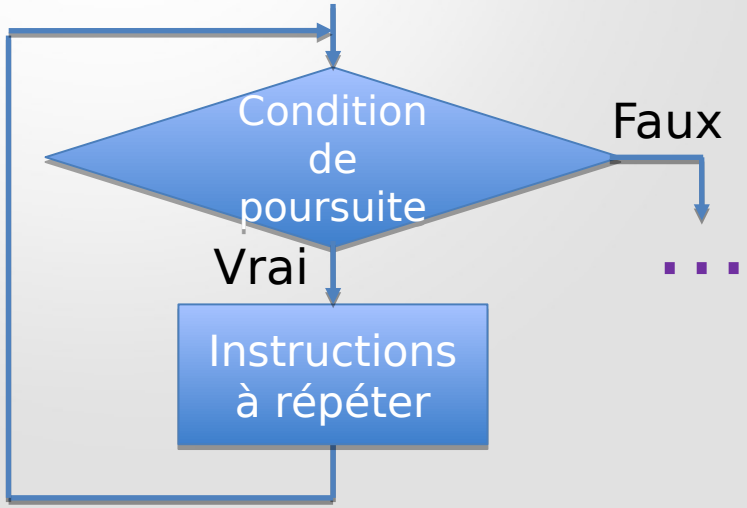


Types de boucles

- On connaît le nombre d'itérations.
  - Ex. Choisir 7 nombres au hasard (Loto).
- Tant qu'une condition n'est pas réalisée, on répète le traitement.
  - Ex. Saisir des noms tant que des personnes s'inscrivent.
- On répète le traitement jusqu'à ce qu'une condition se réalise.

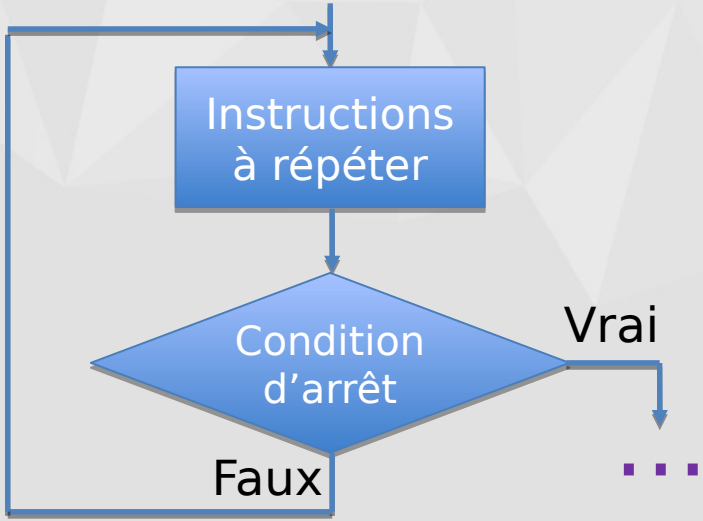
Tant que où répéter jusqu'à ?

Tant que… répéter



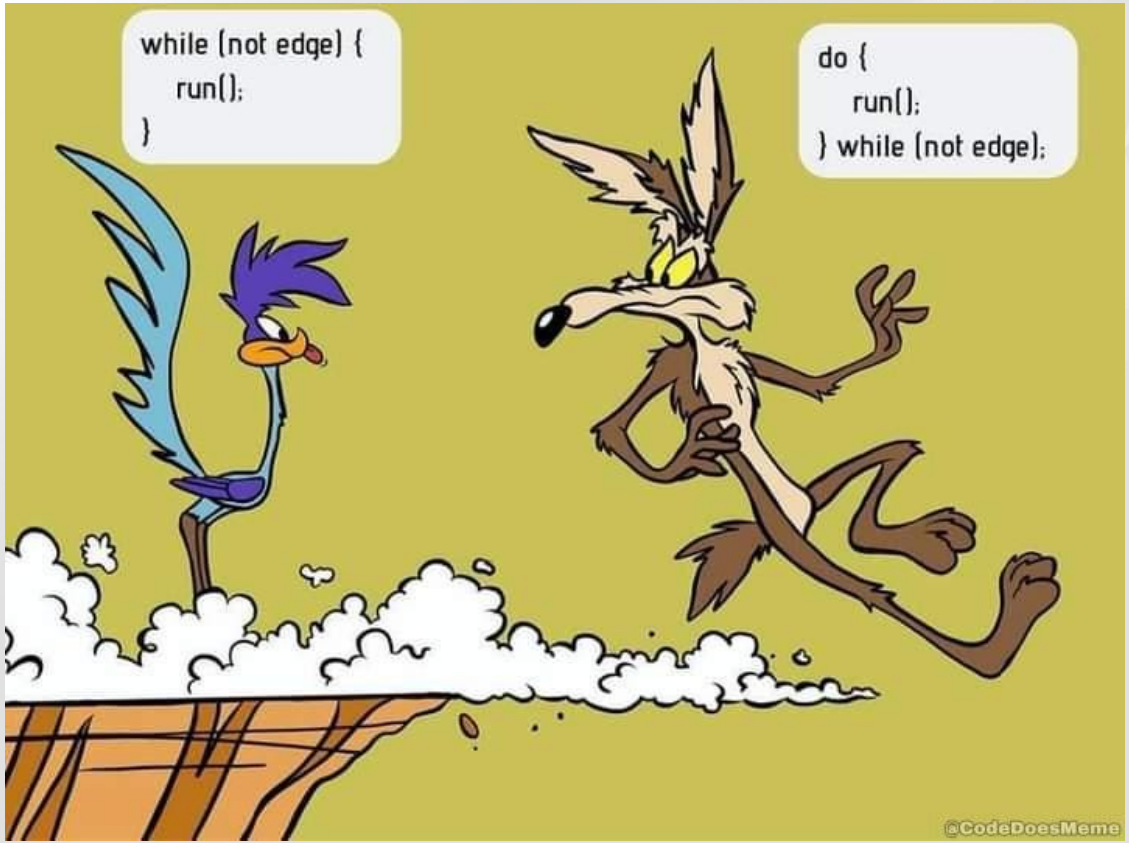
Les instructions peuvent ne pas être exécutées.

Répéter… jusqu'à



Les instructions sont forcément exécutées une fois.

Tant que où répéter jusqu'à ?



Écriture des boucles

Algorithmique	Python

Pour i de 1 à 10 faire
  Écrire(7 x i)
Fin pour
	for i in range(1, 11):
	  print(7 * i)

i ← 1
Tant que i ≤ 10 faire
  Écrire(7 x i)
  i ← i + 1
Fin tant que
	i = 1
	while i <= 10:
	  print(7 * i)
	  i += 1

i ← 1
Répéter
  Écrire(7 x i)
  i ← i + 1
Jusqu'à i > 10
	i = 1
	while True: # Boucle infinie
	  print(7 * i)
	  i += 1
	  if i > 10:
	  break # Très inélégant
	 

Exemple

- Afficher la somme des nombres de 1 à 100.
- Stocker un mot de passe dans une constante. Faire en saisir un à un·e utilisateur·trice jusqu’à ce qu’il soit identique au mot de passe constant.
- Saisir des noms de personnes tant que le mot-clé FIN n’est pas entré.
- Formuler les solutions en langage algorithmique, puis en Python.

Exemple 1 - Algorithmique

Algorithme unCent

Var i, somme : Entiers

Début

somme ← 0
Pour i de 1 à 100 faire
    somme ← somme + i {Cumul de la somme}
Fin pour
Écrire("Somme :", somme)

Fin

{Si on est matheux, somme ← n × (n + 1) ÷ 2
100 × 101 ÷ 2 = 5050}

Exemple 2 - Algorithmique

Algorithme motPasse2

```
    Const MDP ← "All0M@m@nB0b0"
    Var mdpSaisi : Chaîne

Début
    Répéter
        Lire("Mot de passe :", mdpSaisi)
    Jusqu'à mdpSaisi = MDP
Fin
```

Exemple 3 - Algorithmique

Algorithme saisieNoms

Var nom : Chaîne

Début

Lire("Nom :", nom)

Tant que nom ≠ "FIN" faire {On appelle FIN un drapeau ou *flag*}

{Traitement}

Lire("Nom :", nom)

Fin tant que

Fin

Boucles imbriquées (1/2)

- Exemple : comment calculer une table de multiplication ?

	1	**2**	**3**	**4**	**5**
1	1	2	3	4	5
2	2	4	6	8	10
3	3	6	9	12	15
4	4	8	12	16	20
5	5	10	15	20	25

Boucles imbriquées (2/2)

Algorithmique	Python

Pour i de 1 à 5 faire
  Pour j de 1 à 5 faire
  Écrire(i x j)
  Fin pour
Fin pour
	for i in range(1, 6):
	  for j in range(1, 6):
	  print(i * j)

Il est possible d'imbriquer :

- n'importe quel type de boucle (pour, tant que, répéter jusqu'à),
- des types de boucles différents,
- autant de fois que nécessaire.

Partie 4

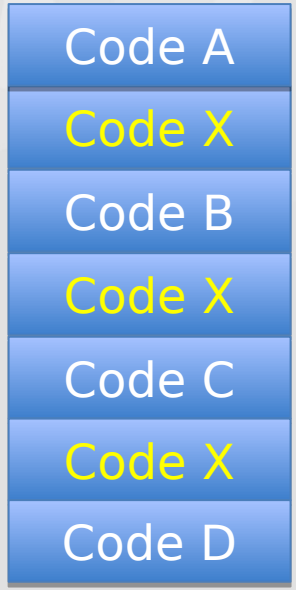
Sous-programmes

Qu’est-ce qu’un sous-programme ? (1/3)

- Que faire quand un même fragment de code se répète dans un programme ?

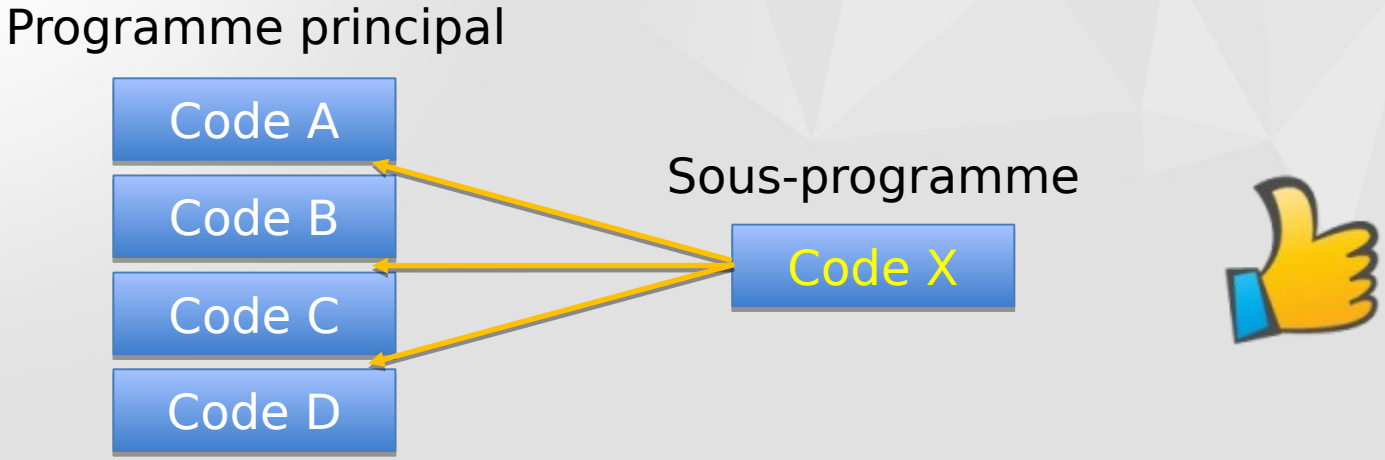
- Copier/coller le fragment

  - Le code devient plus volumineux, moins lisible.
  - En cas de correction, il faut modifier plusieurs fois.



Qu’est-ce qu’un sous-programme ? (2/3)

- Créer un sous-programme



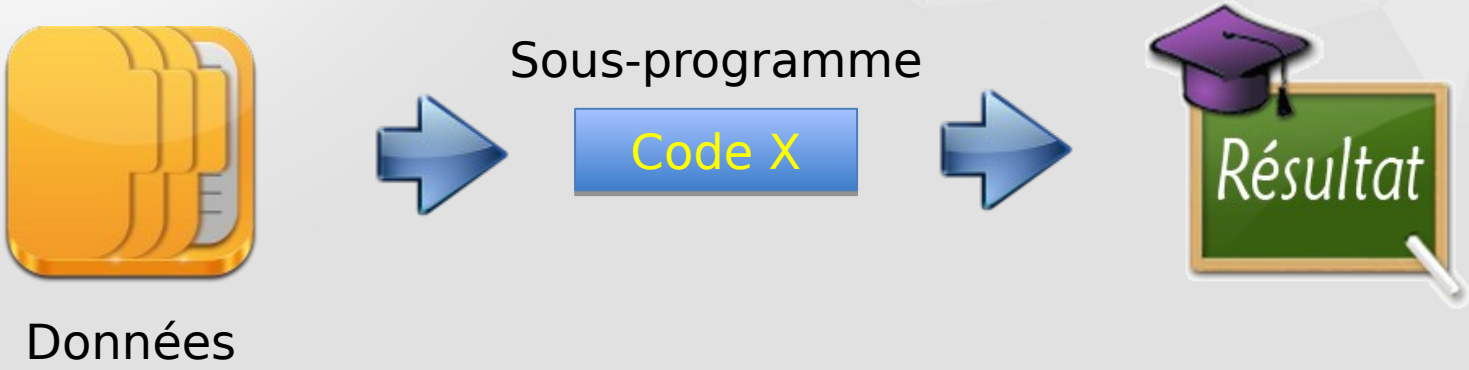
- Le programme principal appelle le sous-programme.

Qu’est-ce qu’un sous-programme ? (3/3)

Un même sous-programme peut-être appelé par plusieurs programmes.

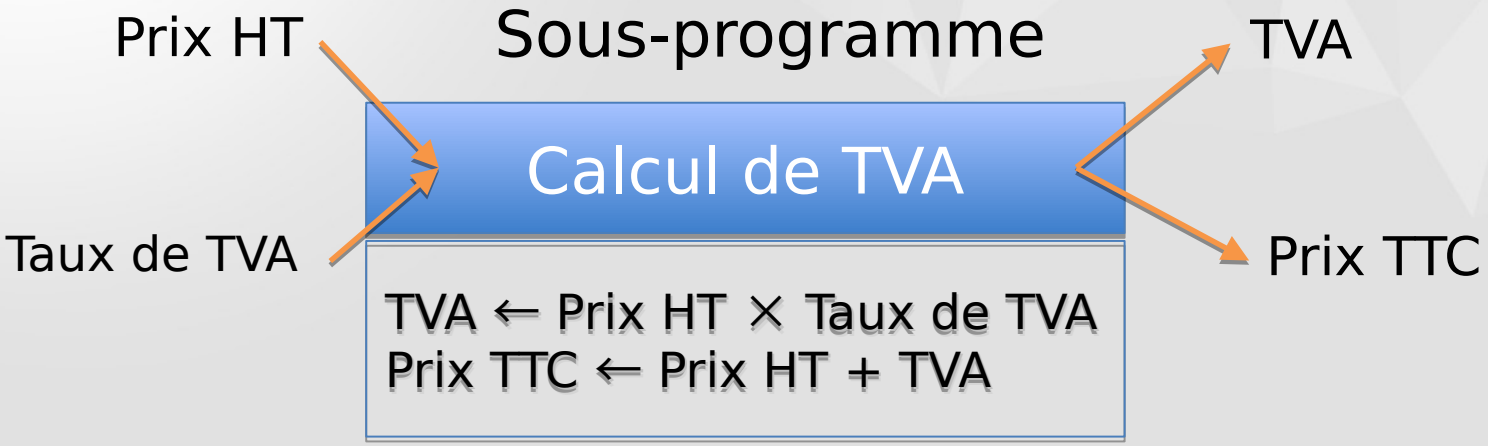
Paramètres

- Comme tout algorithme, un sous-programme a des entrées et des sorties.



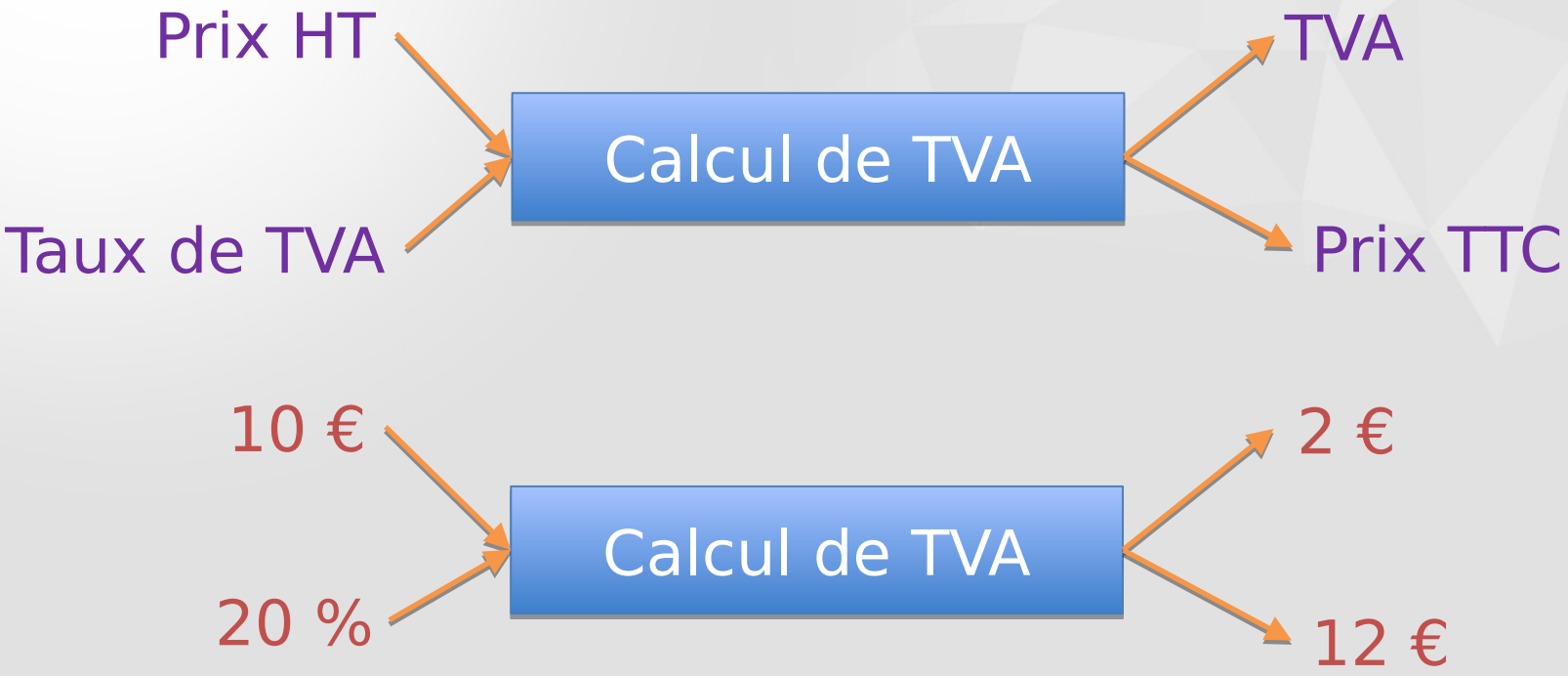
Exemple de paramètres

Paramètres d'entrée

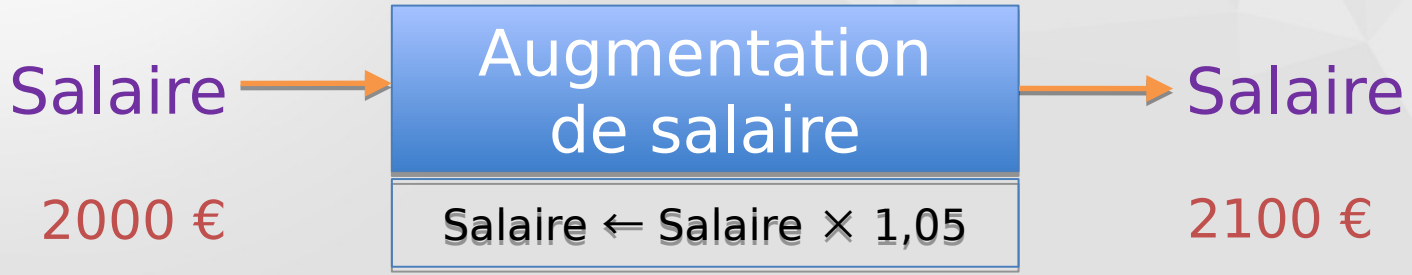


Paramètres de sortie

Paramètres formels/Paramètres réels



Paramètres d’entrée/sortie



Types de sous-programmes

- Fonction : effectue en général un calcul dont le résultat est retourné au programme principal
  - Ex. Diviser une note sur 20 par 2 pour obtenir une note sur 10.
- Procédure : effectue tout type de traitement
  - Ex. Couper une chaîne de caractères contenant un nom et un prénom en deux
  - Affichages à l'écran

Écriture des fonctions

Algorithmique	Python

Fonction tva(prixHT : Réel) : Réel
  Const TAUX ← 0,2
Début
  Retourner prixHT x (1 + TAUX)
Fin
 	
{Appel de la fonction
 dans le programme principal}
prixTTC ← tva(10,5)
	def tva(prixHT):
	  TAUX = 0.2
	
	  return prixHT * (1 + TAUX)
	
	# Appel de la fonction
	# dans le programme principal
	prixTTC = tva(10.5)

Spécificité des fonctions Python

- Elles peuvent retourner plus d’un résultat.

```
# Exemple
def tva2(prixHT):
    TAUXPLEIN = 0.2
    TAUXREDUIT = 0.055
    return prixHT * (1 + TAUXPLEIN), prixHT * (1 + TAUXREDUIT)

# Appel de la fonction dans le programme principal
prixTTCplein, prixTTCreduit = tva2(10.5)
```

Écriture des procédures (1/2)

Algorithmique	Python

Procédure table7()
  Var i : Entier
Début
  Pour i de 1 à 10 faire
  Écrire(7 x i)
  Fin pour
Fin
	def table7():
	
	  for i in range(1, 11):
	  print(7 * i)

Procédure table(n : Entier)
  Var i : Entier
Début
  Pour i de 1 à 10 faire
  Écrire(n x i)
  Fin pour
Fin
	def table(n):
	
	  for i in range(1, 11):
	  print(n * i)

Écriture des procédures (2/2)

Algorithmique	Python

Procédure tablegen(n : Entier, m : Entier)
  Var i : Entier
Début
  Pour i de 1 à m faire
  Écrire(n x i)
  Fin pour
Fin
	def tablegen(n, m):
	
	  for i in range(1, m + 1):
	  print(n * i)

{Appel des procédures
 dans le programme principal}
table7()
table(3)
max ← 15
tablegen(3, max)
	# Appel des procédures
	# dans le programme principal
	table7()
	table(3)
	max = 15
	tablegen(3, max)

Types de paramètres

- Algorithmique
  - Paramètres d'entrée : Procédure table(Entrée n : Entier)
    Procédure table(n : Entier)
  - Paramètres de sortie : Procédure f(Entrée x : Entier, Sortie y : Entier)
  - Paramètres d'entrée/sortie :
    Procédure salairePlus(Entrée/sortie Salaire : Réel)
- Python
  - Uniquement des paramètres d'entrée
  - Les sorties sont retournées (return) car Python n'a que des fonctions.

Exemple

- Créer une fonction prenant deux nombres entiers en paramètres d’entrée et retournant le plus grand.
- Créer une procédure prenant en paramètres d’entrée un nom d’étudiant·e et une note, et affichant « *nom* a obtenu *note* / 20 » à l’écran.
- Appeler la fonction et la procédure.
- Formuler les solutions en langage algorithmique, puis en Python.

Exemple – Algorithmique (1/2)

```
Fonction maxi(n1, n2 : Entiers) : Entier
Début
    Si n1 > n2 alors
        Retourner n1
    Sinon
        Retourner n2
    Fin si
Fin

Procédure afficheNote(nom : Chaîne, note : Réel)
Début
    Écrire(nom, " a obtenu ", note, " / 20")
Fin
```

Exemple – Algorithmique (2/2)

```
Algorithme programmePrincipal
    Var nb1, nb2, maximum : Entiers
    Var nomEtu : Chaîne
    Var noteEtu : Réel
Début
    Lire("1er nombre :", nb1)
    Lire("2e nombre :", nb2)
    maximum ← maxi(nb1, nb2)          {ou directement}
    Écrire("Maximum :", maximum)      {Écrire("Maximum :", maxi(nb1, nb2))}
    Lire("Nom :", nomEtu)
    Lire("Note :", noteEtu)
    afficheNote(nomEtu, noteEtu)
Fin
```

Modules

- Module (ou bibliothèque) :
  Ensemble de procédures et de fonctions
  - liées thématiquement
  - pouvant être appelées depuis plusieurs programmes

- Nombreux modules intégrés à Python
  - Ex. math, os, tkinter (interface graphique)
- Très nombreux modules externes
  - Ex. matplotlib (courbes et graphiques), scikits-learn (fouille de données)

Utiliser des modules (1/2)

```
# Importer un ou plusieurs modules
import os                       # Module « os » (système d'exploitation)
import os, math                 # Modules « os » et « math »


# Utiliser les fonctions d'un module
os.chdir("~/jerome")            # Fonction changement de dossier (module « os »)
racineCarree = math.sqrt(4)     # Fonction racine carrée (module « math »)
```

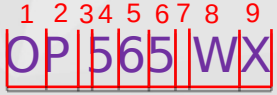
Utiliser des modules (2/2)

```
# Importer les fonctions d'un module
from os import chdir                # Fonction changement de dossier (module « os »)
from math import *                  # Toutes les fonctions du module « math »


# Utiliser les fonctions
chdir("~/jerome")                   # Plus besoin de préfixer la fonction
racineCarree = sqrt(4)              # par le nom du module


# Mode d'emploi des fonctions
help("os")                  # Liste et description de toutes les fonctions du module
help("os.chdir")            # Description de la fonction « chdir » du module « os »
```

Partie 5

Structures de données avancées (chaînes et listes)

Définitions

- Chaîne de caractères : texte (court) composé d'une suite ordonnée de caractères
  - Ex. Un message : Hello world!
  - Ex. Un numéro d'immatriculation : 

  - Ex. Un mot de passe : V9=24gEwY::74pFp/pH&

- Liste : ensemble ordonné de données
  - Ex. Des notes : 11 8,5 16 14,3 11,6 10
  - Ex. Des contacts : Agnieszka Bruno Isabel Jean-Philippe Sabine

Extraire des parties de chaînes

Algorithmique	Python
lettre3 ← nom[3]	lettre3 = nom[2] # 1re position = 0

extrait ← SousChaîne(nom, 3, 5)
extrait ← SousChaîne(nom, 1, 5)
	extrait = nom[2 : 5] # Arrêt avant 5
	extrait = nom[: 5] # Depuis le début

Transformer des chaînes

Algorithmique	Python

longueur ← Longueur(nom)
nom_maj ← Majuscules(nom)
txt ← " A bra ca da bra "
txt ← SupprimerEspaces(txt)
	longueur = len(nom) # Fonction
	nom_maj = nom.upper() # Méthode
	txt = " A bra ca da bra "
	txt = txt.strip()

Rechercher/remplacer dans des chaînes

Algorithmique	Python
pos ← ChercherPosition(txt, "bra")	pos = txt.find("bra")
txt ← Remplacer(txt, "bra", "BRA")	txt = txt.replace("bra", "BRA")

Créer des listes

Algorithmique	Python
Var lst : Liste d'entiers {Déclaration}	

lst ← ( ) {Liste vide}
lst ← (1, 5, 7)
Écrire(lst(2)) {2e valeur}
	lst = [ ] # Liste vide
	lst = [1, 5, 7]
	print(lst[1]) # 2e valeur
	
	# Multi-type !
	lst2 = [1, 3.14, "ok", [ ]]

Meme informatique

Parcourir une liste

Algorithmique	Python

Pour tout nombre dans lst faire
    Écrire(nombre)
Fin pour
	for nombre in lst:
	    print(nombre)

(Boucle pour tout élément de liste spécifique)

Mettre une liste à jour

Algorithmique	Python
Lst( ) ← 9.2 {Ajout en fin}	lst.append(9.2) # Ajout en fin
Ajouter(lst, 2, 4.5) {Ajout 2e position}	lst.insert(1, 4.5) # Ajout en 2e position
Lst(2) ← 3.0 {Modification 2e pos.}	Lst[1] = 3.0 # Modification 2e pos.
Suppr(lst, 3) {Suppression 3e pos.}	lst.pop(2) # Suppression 3e pos.

Manipuler une liste

Algorithmique	Python
taille ← Longueur(lst) Trier(lst) pos ← ChercherPosition(lst, 3.0) mots ← Découper(txt, " ") {mots est une liste de chaînes}	taille = len(lst) lst.sort() pos = lst.index(3.0) mots = txt.split(" ") # mots est une liste de chaînes

Exemple

- Créer une liste de noms.
- Afficher la liste de noms.
- Avant l’affichage, mettre tous les noms de la liste en majuscules.
- Avant l’affichage, trier la liste de noms.
- Formuler les solutions en langage algorithmique, puis en Python.

Exemple - Algorithmique

```
Algorithme listeNoms
    Var liste : Liste de Chaînes
    Var i : Entier
    Var nom : Chaîne
Début
    liste ← ("ba", "bo", "be", "bu", "bi")
    Pour i de 1 à Longueur(liste) faire
        liste(i) ← Majuscule(liste(i))
    Fin pour
    Trier(liste)
    Pour tout nom dans liste faire
        Écrire(nom)
    Fin pour
Fin
```

Partie 6

Stocker des données persistantes (fichiers)

Qu’est-ce qu’un fichier ?

- Fichier : ensemble nommé de données enregistrées sur un support de stockage permanent (disque dur ou SSD, clé USB, carte mémoire, etc.)

- Fichier texte : fichier dont le contenu est un ensemble de lignes

Créer un fichier

Algorithmique	Python
{Déclarations} Var f : Fichier Var lignes : Liste de Chaînes Var l : Chaîne	
lignes ← ("Angèle", "Bernard", "Cherifa", "Doug") f ← Ouvrir("fichier.txt", "écriture") Pour l dans lignes faire Écrire(f, l) Fin pour Fermer(f)	lignes = ["Bernard\n", "Cherifa\n", "Doug\n"] # \n = passage à la ligne f = open("fichier.txt", "w") # (w)rite f.write("Angèle\n") # Ecriture unique f.writelines(lignes) # Ecriture multiple f.close()

Manipuler un fichier

	Algorithme	Python
Ajout	lignes ← ("Ekaterina", "Fali") f ← Ouvrir("fichier.txt", "ajout") Pour l dans lignes faire Écrire(f, l) Fin pour Fermer(f)	lignes = ["Ekaterina\n", "Fali\n"] f = open("fichier.txt", "a") # (a)ppend f.writelines(lignes) f.close()

Lecture	f ← Ouvrir("fichier.txt", "lecture")
	Lire(f, l)
	Tant que non FinDeFichier(f) faire
	  lignes( ) ← l
	  Lire(f, l)
	Fin tant que
	Fermer(f)
		f = open("fichier.txt", "r") # (r)ead
		lignes = f.readlines()
		  # lignes est une liste de chaînes
		
		f.close()

Exemple

- Créer une liste de noms et une liste de salaires.
- Stocker les noms et les salaires dans un fichier. Un nom et le salaire correspondant doivent être séparés par une virgule (fichier CSV – *comma-separated values*).
- Lire le fichier et afficher les valeurs qu'il contient.
- Formuler les solutions en langage algorithmique, puis en Python.

Exemple - Algorithmique

Algorithme créerCSV

Var f : Fichier
Var listeNoms : Liste de Chaînes
Var listeSal : Liste de Réels
Var i : Entier
Var ligne : Chaîne

Début

listeNoms ← ("Aïcha", "Boris","Cécile")

listeSal ← (2500, 1500, 2250)

f ← Ouvrir("noms-salaires.csv", "écriture")
Pour i de 1 à Longueur(listeNoms) faire {Il faut que les deux listes aient la même longueur.}
    Écrire(f, listeNoms(i) | "," | convChaîne(listeSal(i))) {| est l'opérateur de concaténation}
Fin pour
Fermer(f)

Exemple – Algorithmique (suite)

```
        f ← Ouvrir("noms-salaires.csv", "lecture")

        Lire(f, l)

        Tant que non FinDeFichier(f) faire
              Écrire(l)
              Lire(f, l)
        Fin tant que

        Fermer(f)
Fin
```

THE END