Nom : Nkumbe Aurelien   
Classe : Master 1 info

TD/TP Algorithmique

Contents

[Exercice 1 2](#_Toc119840731)

[Exercice 2 2](#_Toc119840732)

[Exercice 3 2](#_Toc119840733)

[Exercice 4 3](#_Toc119840734)

[Exercice 5 3](#_Toc119840735)

[Exercice 6 3](#_Toc119840736)

[Exercice 7 4](#_Toc119840737)

[Exercice 8 5](#_Toc119840738)

[Exercice 9 5](#_Toc119840739)

[Exercise 10 6](#_Toc119840740)

[Exercise 11 7](#_Toc119840741)

[Exercice 12 7](#_Toc119840742)

[Excerice 13 8](#_Toc119840743)

[Exercice 14 11](#_Toc119840744)

## Exercice 1

Donner la valeur des expressions suivantes :

* 2 + 3 \* 4 = 14
* 8 / 2 + 1 = 5
* 8/ (2 + 1) = 2.67
* 12 < 5 = Faux
* « cours » + « algo » = « cours algo »
* 54 / ln (1) = Impossible
* Vrai ET Faux OU Vrai = VRAI
* (2 < 3) ET (4 > 5) = FAUX
* (2 + 6 = 10) OU (8 / 2 > 3) = VRAI

## Exercice 2

1. X > 3 ET Y > 3
2. X = Y ET Y =Z
3. Y <= X ET X <= Z
4. X = Y OU Y = Z OU X = Z
5. (X = Y ET Y <> Z) OU (X = Z ET Y <> Z)

## Exercice 3

Remaques

* La variable pof n’est pas nécessaire
* Certains mots clés ne sont pas dans le bon formalisme (en Majuscule) :

Rôle : Calculer la circonférence du cercle 2 Pi R

ALGORITHME mystere

CONSTANTES (PIF : Reel) <-- 3.14

**VARIABLES** paf , puf : Réels

DEBUT

  SAISIR paf

  puf <-- 2 \* PIF \* paf

  AFFICHER puf

FIN

## Exercice 4

ALGORITHME permutation

**VARIABLES** a, b : Nombres

DEBUT

  SAISIR a

  SAISIR b

  a <-- a + b

  b <-- b - a

  a <-- b - a

  AFFICHER a

  AFFICHER b

FIN

## Exercice 5

1er Cas :

L’algorithme boucle à l’infini (jusqu’à interruption volontaire de l’utilisateur ou du système d’exploitation pour éviter un stack overflow) car la variable x reste toujours inchangé a la valeur 1

2e Cas :

Le programme affiche la valeur de x égale

**3628800**

Preuve

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **K** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **X** | 1 | 1 | 2 | 6 | 24 | 120 | 720 | 5040 | 40320 | 362880 |
| **X \* K** | 1 | 2 | 6 | 24 | 120 | 720 | 5040 | 40320 | 362880 | **3628800** |

## Exercice 6

Analyse

L’algorithme initiale échouera à partir de l’analyse syntaxique, comme amélioration on peut

* On peut déclarer la variable Z
* Afficher les résultats de C et Z en plus

**FONCTIONS\_UTILISEES**

**VARIABLES**

  A EST\_DU\_TYPE NOMBRE

  B EST\_DU\_TYPE NOMBRE

  C EST\_DU\_TYPE NOMBRE

  // new ligne

  Z EST\_DU\_TYPE NOMBRE

**DEBUT\_ALGORITHME**

  LIRE A

  LIRE B

  C PREND\_LA\_VALEUR 2 \* A + B

  SI (C = 10) ALORS

    DEBUT\_SI

    A PREND\_LA\_VALEUR C

    // new ligne

    AFFICHER C

    FIN\_SI

    SINON

      DEBUT\_SINON

      Z PREND\_LA\_VALEUR 2 \* C

      // new ligne

      AFFICHER Z

      FIN\_SINON

**FIN\_ALGORITHME**

## Exercice 7

def exercice7 () :

    LIMIT = 10

    list\_nombres\_paires = []

    print("Trouver les  nombres  paires consecutifs")

    nb = int(input("Entrer un nombre entier <--  "))

    while (len(list\_nombres\_paires) < LIMIT) :

        if (nb % 2 == 0) :

            list\_nombres\_paires.append(nb)

        nb += 1

    print("Les nombres  paires consecutifs trouvés", list\_nombres\_paires)



## Exercice 8

def exercice8 () :

    produit = 1

    print("Trouver le produit des entiers")

    nb = int(input("Entrer un nombre  <--  "))

    for x in range(1, nb + 1) :

        produit \*= x

    print("Le produit factoriel ", nb ,"! est", produit)

Graphical user interface, text

Description générée automatiquement

## Exercice 9

1. Fonction d’Armstrong

def exercice9\_test\_nombre\_armstrong(nombre):

    charaters = list(str(nombre))

    result = 0

    for item in charaters :

        nb = int(item)

        result = result + pow(nb,3)

    return int(result) == int(nombre)

1. Programme

def exercice9 () :

    # Nombre d'Armstrong

    print("Nombre d'Armstrong")

    nb = input("Entrer le nombre a vérifié <--  ")

    nb = int(nb)

    for nb\_rank in range(1,nb+1):

        if (exercice9\_test\_nombre\_armstrong(nb\_rank)) :

            print(nb\_rank)

## Exercise 10

1. Afficher le max

def exercice10\_1 () :

    LIMIT = 10

    max = 0

    for n in range (LIMIT) :

        nb = input("Entrer le ("+ str(n + 1) +") nombre <--") or 0

        nb = int(nb)

        if (nb > max) :

            max = nb

    print("Le max est", max)

1. Conserver les nombres entrés

def exercice10\_2 () :

    LIMIT = 10

    liste\_nombres = [] #++++

    max = 1

    for n in range (LIMIT) :

        nb = input("Entrer le ("+ str(n + 1) +") nombre <--") or 0

        nb = int(nb)

        liste\_nombres.insert(n, nb) #++++

        if (nb > max) :

            max = nb

    print("Le max est", max)

1. Enregistrer et affricher l’index du max

def exercice10\_3 () :

    LIMIT = 10

    liste\_nombres = []

    max = 0

    max\_idx = None  #++++

    for n in range (LIMIT) :

        nb = input("Entrer le ("+ str(n + 1) +") nombre <--") or 0

        nb = int(nb)

        liste\_nombres.insert(n, nb)

        if (nb > max) :

            max = nb

            max\_idx = n + 1 #++++

    print("Le max est", max, " et d'indice " , max\_idx) #++++

## Exercise 11

1. Programme

import random

def exercice11 ():

    LIMIT = 100

    list\_1\_to\_100 = []

    somme = 0

    for x in range(1, LIMIT + 1):

        x = round(random.random() \* LIMIT)

        list\_1\_to\_100.append(x)

        somme += x

    # print(list\_1\_to\_100)

    print("La sommme donne", somme)

1. Découpage en fonction

import random

def exercice11\_random\_integer\_list(limit = 100):

    list\_1\_to\_100 = []

    for x in range(1, limit + 1):

        x = round(random.random() \* limit)

        list\_1\_to\_100.append(x)

    return list\_1\_to\_100

def exercice11\_somme(liste):

    s = 0

    for x in liste:

        s += x

    return s

def exercice11\_main():

    list\_1\_to\_100 = exercice11\_random\_integer\_list(100)

    somme = exercice11\_somme(list\_1\_to\_100)

    print("La sommme donne", somme)

## Exercice 12

import math

def exercice12 ():

    UNIT\_M3 = 8

    print("Calcul du nombre de radiateur pour chauffer un piece")

    hauteur = input("Entrer la hauteur <--") or 0

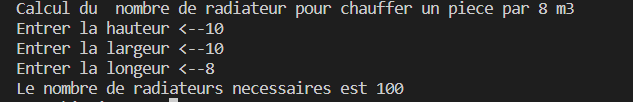
    largeur = input("Entrer la largeur <--") or 0

    longeur = input("Entrer la longeur <--") or 0

    aire = float(longeur) \* float(largeur) \* float(hauteur)

    nb\_radiateurs = aire / UNIT\_M3

    print("Le nombre de radiateurs necessaires est", math.ceil(nb\_radiateurs))



## Excerice 13

1. Completer l’algo

//  Exercice 13 - 1  =================================

**FONCTIONS\_UTILISEES**

**VARIABLES**

  nb EST\_DU\_TYPE NOMBRE

  code EST\_DU\_TYPE NOMBRE

**DEBUT\_ALGORITHME**

  code PREND\_LA\_VALEUR round(random()\*10)

  AFFICHER code

  AFFICHER "\nDeviner  le code caché"

  LIRE nb

  TANT\_QUE (code != nb) FAIRE

    DEBUT\_TANT\_QUE

    AFFICHER "reessayé"

    LIRE nb

    FIN\_TANT\_QUE

  AFFICHER "\nBien joué"

**FIN\_ALGORITHME**

1. Permettre de choisir entre 0 et 100

//  Exercice 13 - 2  =================================

**FONCTIONS\_UTILISEES**

**VARIABLES**

  nb EST\_DU\_TYPE NOMBRE

  tailleCode EST\_DU\_TYPE NOMBRE

  code EST\_DU\_TYPE NOMBRE

**DEBUT\_ALGORITHME**

  AFFICHER "\nChoisir la taille du code entre 0 et 100"

  LIRE tailleCode

  TANT\_QUE (tailleCode < 0 OU tailleCode > 100) FAIRE

    DEBUT\_TANT\_QUE

    AFFICHER "\nReessayé, Choisir la taille du code entre 0 et 100"

    LIRE tailleCode

    FIN\_TANT\_QUE

  code PREND\_LA\_VALEUR round(random()\*tailleCode)

  AFFICHER code

  AFFICHER "\nDeviner  le code caché"

  LIRE nb

  TANT\_QUE (code != nb) FAIRE

    DEBUT\_TANT\_QUE

    AFFICHER "reessayé"

    LIRE nb

    FIN\_TANT\_QUE

  AFFICHER "\nBien joué"

**FIN\_ALGORITHME**

1. Ajouter les conditions

**FONCTIONS\_UTILISEES**

**VARIABLES**

  nb EST\_DU\_TYPE NOMBRE

  tailleCode EST\_DU\_TYPE NOMBRE

  code EST\_DU\_TYPE NOMBRE

**DEBUT\_ALGORITHME**

  AFFICHER "\nChoisir la taille du code entre 0 et 100"

  LIRE tailleCode

  TANT\_QUE (tailleCode < 0 OU tailleCode > 100) FAIRE

    DEBUT\_TANT\_QUE

    AFFICHER "\nReessayé, Choisir la taille du code entre 0 et 100"

    LIRE tailleCode

    FIN\_TANT\_QUE

  code PREND\_LA\_VALEUR round(random()\*tailleCode)

  AFFICHER code

  AFFICHER "\nDeviner  le code caché"

  LIRE nb

  TANT\_QUE (code != nb) FAIRE

    DEBUT\_TANT\_QUE

     SI (nb < code) ALORS

      DEBUT\_SI

      AFFICHER "\nDonner une valeur superieur a "

      AFFICHER nb

      FIN\_SI

      SINON

        DEBUT\_SINON

        AFFICHER "\nDonner une valeur inferieur a "

        AFFICHER nb

        FIN\_SINON

    LIRE nb

    FIN\_TANT\_QUE

  AFFICHER "\nBien joué"

**FIN\_ALGORITHME**

1. Limiter le nombre de tentative

//  Exercice 13 - 4  =================================

**FONCTIONS\_UTILISEES**

**VARIABLES**

  nb\_tentative EST\_DU\_TYPE NOMBRE

  nb EST\_DU\_TYPE NOMBRE

  tailleCode EST\_DU\_TYPE NOMBRE

  code EST\_DU\_TYPE NOMBRE

**DEBUT\_ALGORITHME**

  nb\_tentative PREND\_LA\_VALEUR 3

  AFFICHER "\nChoisir la taille du code entre 0 et 100"

  LIRE tailleCode

  TANT\_QUE (tailleCode < 0 OU tailleCode > 100) FAIRE

    DEBUT\_TANT\_QUE

    AFFICHER "\nReessayé, Choisir la taille du code entre 0 et 100"

    LIRE tailleCode

    FIN\_TANT\_QUE

  code PREND\_LA\_VALEUR round(random()\*tailleCode)

  AFFICHER code

  AFFICHER "\nDeviner  le code caché"

  LIRE nb

  TANT\_QUE (nb\_tentative > 1 ET code != nb) FAIRE

    DEBUT\_TANT\_QUE

    nb\_tentative PREND\_LA\_VALEUR nb\_tentative - 1

    AFFICHER "\nVous avez "

    AFFICHER nb\_tentative

    AFFICHER " tentatives restantes"

    SI (nb < code) ALORS

      DEBUT\_SI

      AFFICHER "\nDonner une valeur superieur a "

      AFFICHER nb

      FIN\_SI

      SINON

        DEBUT\_SINON

        AFFICHER "\nDonner une valeur inferieur a "

        AFFICHER nb

        FIN\_SINON

    LIRE nb

    FIN\_TANT\_QUE

  SI (nb == code) ALORS

      DEBUT\_SI

      AFFICHER "\nBien joué"

      FIN\_SI

      SINON

        DEBUT\_SINON

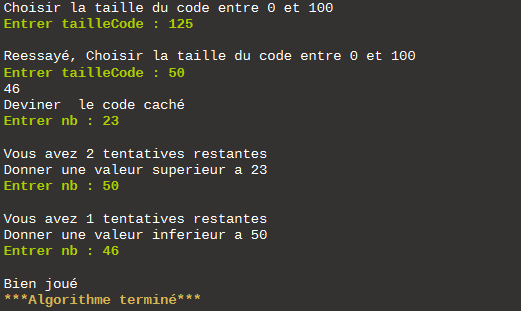
        AFFICHER "\nDésolé, vous avez perdu, "

        AFFICHER "le code a deviné etait "

        AFFICHER code

        FIN\_SINON

**FIN\_ALGORITHME**



## Exercice 14

def exercice14 ():

    UNIT\_LKM = 100

    cout = None

    print("Calcul cout du carburant pour le voyage")

    distance\_km = input("Entrer la distance du voyage <--") or 0

    prix\_litre = input("Entrer le prix du litre de carburant <--") or 0

    consommation\_moy\_l\_km = input("Entrer la consommation moyenne (en L/"+ str(UNIT\_LKM) +"km) <--") or 0

    # Raisonnement par la regle de 3

    # consommation\_moy\_l\_km -----> UNIT\_LKM

    # nb\_litre ?????            -----> distance\_km

    nb\_litre = float(distance\_km)  \* float(consommation\_moy\_l\_km) / UNIT\_LKM

    cout = nb\_litre \* float(prix\_litre)

    print("Le cout du carburant pour le voyage est", math.ceil(cout))

