

Algorithmes et structures de données Cours 10

Patrick Reuter

<http://www.labri.fr/~preuter/asd2008>

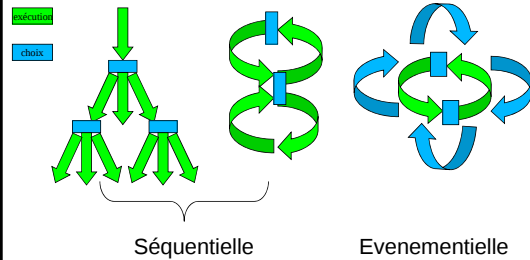
Programme

- Retour sur TD7 machine
- Aujourd'hui : salle TD
- Listes 1D

Les entrées du clavier

Programmation évènementielle

Entrées de l'utilisateur



Les entrées du clavier

`a = input()`

affecte la variable **a** avec une variable de type entier ou flottant.

`a = raw_input()`

affecte la variable **a** avec une variable de type chaîne de caractères.

Programmation séquentielle

Entrée de l'utilisateur :

`a = raw_input();`

Ceci peut être du texte ou des nombres

Programmation séquentielle

Entrée de l'utilisateur :

```
print "Taper quelque chose : ",
a = raw_input();
print "Vous avez tapé : ",a
```

Programmation séquentielle

Entrée de l'utilisateur :

```
a = raw_input();

Ceci peut être du texte ou des
nombres
```

Programmation séquentielle

Entrée de l'utilisateur :

```
print "Taper quelque chose : ",
a = raw_input();
print "Vous avez tapé : ",a

print "le nombre de caractères : ",
print len(a)
```

Programmation séquentielle

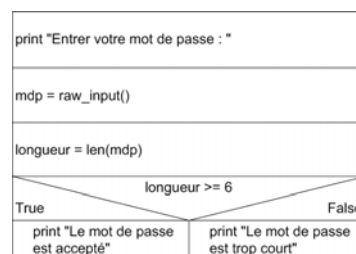
Entrée de l'utilisateur :

```
print "Taper quelque chose : ",
a = input();
print "Vous avez tapé : ",a

print "au carré cela fait : ",
print a*a # ou bien a**2
```

Retour sur TD7

TD7 – Exercice 7.4.1



Condition nécessaire/suffisante

- Condition nécessaire
- Condition suffisante
- Condition nécessaire et suffisante

Conditions nécessaire

- A est une condition nécessaire pour B
- B implique A
- $B \Rightarrow A$.
- B seulement si A

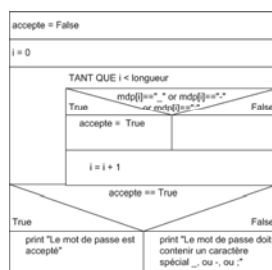
Condition suffisante :

- A est une condition suffisante pour B
- A implique B
- $A \Rightarrow B$
- B si A

Condition nécessaire et suffisante

- B si et seulement si A

Préparation TD 7.4.2



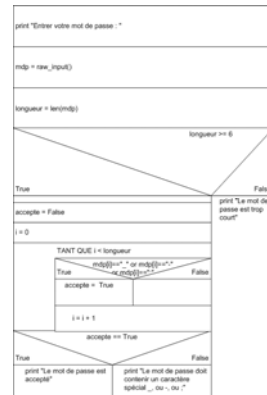
- FAUX :

if (mdp[i]=="_" or ":" or ";")

- CORRECT

if (mdp[i]=="_" or mdp==":" or mdp==";")

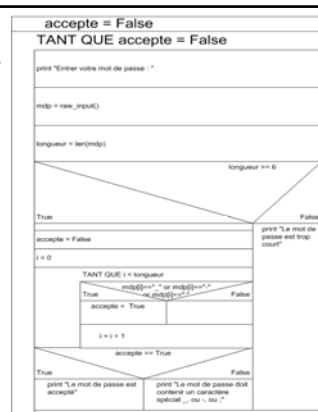
Exercice TD 7.4.2



Exercice TD 7.4.3

- Encapsuler dans une boucle TANT QUE

Exercice TD 7.4.3



- Exercice : voulez-vous rejouer

```

a="oui"
while (a == "oui"):
    # votre jeu ici
    print "Voulez-vous rejouer (oui/non) ?"
    a = raw_input()
    print "Au revoir"
  
```

Variables

Variables

Type simples:

Type booléen

– Vrai/faux (p.ex. True, **boolean**)

Type entier

– Nombre entier (p.ex. 5, **int**)

Type flottant

– Nombre à virgule flottant (p.ex. 5.12, **float**)

Variables

Type composés:

Type chaîne de caractères

– (p.ex. "Bonjour", **string**)

Type liste

– Nombre entier (p.ex. [1,2,3,9,6,7,8,4,5], **list**)

Chaînes de caractères

- indice d'une liste ou d'une chaîne de caractères :

```
s = "Bonjour";  
premiereLettre = s[0];  
deuxiemeLettre = s[1];
```

Fonctions prédéfinies

- par exemple connaître la longueur d'une chaîne :

```
longueur = len(nombres);
```

→ parenthèses !

- **Combinaison :**

```
derniereLettre = s[len(s) - 1];
```

Les listes (1D)

Les listes

- Collection hétérogène, ordonnée et modifiable d'éléments séparés
- par des virgules, et entourée de crochets.

```
nombre = [17, 38, 10, 25, 72]
```

```
print nombre[0]      # 17  
print nombre[4]      # 72  
print len(nombre)    # 5
```

Afficher tout les nombres :

```
nombre = [17, 38, 10, 25, 72]
```

Affectation

```
nombre = [17, 38, 10, 25, 72];  
nombre[0] = 100;  
print nombre;
```

Résultat :

```
[100, 38, 10, 25, 72]
```

Exercice Listes 1

Afficher tout les nombres :

```
nombre = [17, 38, 10, 25, 72]
```

Exercice Listes 1

Afficher tout les nombres :

```
nombres = [17, 38, 10, 25, 72]
```

```
i = 0
TANT QUE i < len(nombres) FAIRE
    afficherLigne(nombres[i])
    i = i + 1
FIN TANT QUE
```

Exercice Listes 1

```
nombres = [17, 38, 10, 25, 72]
i = 0
longueur = len(nombres)
while i < longueur:
    print nombres[i]
```

Exercice Listes 2

Afficher uniquement les nombres pairs :

```
nombres = [17, 38, 10, 25, 72]
```

Exercice Listes 2

Afficher uniquement les nombres pairs :

```
nombres = [17, 38, 10, 25, 72]
i = 0
TANT QUE i < len(nombres) FAIRE
    SI nombres[i] % 2 == 0 ALORS
        afficherLigne(nombres[i])
    FIN SI
    i = i + 1
FIN TANT QUE
```

Exercice Listes 2

```
nombres = [17, 38, 10, 25, 72]
i = 0
longueur = len(nombres)
while i < longueur:
    if (nombres[i] % 2 == 0):
        print nombres[i]
    i = i + 1
```

Exercice Listes 3

Afficher si la liste contient au moins un nombre pair

```
nombres = [17, 38, 10, 25, 72]
```

Exercice Listes 3

```
nombres = [17, 38, 10, 25, 72]

pair = False

i = 0
longueur = len(nombres)
while i < longueur:
    if (nombres[i] % 2 == 0):
        pair = True
        i = i + 1

if pair == True:
    print "oui"
else:
    print "non"
```

Exemple d'Application

- Jeux de cartes
 - Décider si un tas de cartes contient l'As de Pik

