1. **Tests et Condition**

**Exemple : La puissance nominale d’un appareil électrique**

**La puissance nominale est la puissance électrique qu’il reçoit lorsqu’il est soumis à sa tension nominale.**

Elle s’exprime en Watt (symbole W).

Pour les lampes, les appareils de chauffage, les plaques de cuisson, ..., on peut calculer cette puissance en utilisant la formule :  où U est la tension en volt (symbole V) et I l’intensité en ampère (symbole I).

On souhaite programmer en Python, une application qui :

* **Demande à l’utilisateur la tension U en V :**
* **Demande l'intensité I en A :**
* **Calcule et affiche la puissance : P = … ET print(…)**
* **Selon cette puissance, affiche un classement :**
	+ Si P<5  : « L’appareil est de faible puissance ».
	+ Si  : « L’appareil est de puissance moyenne ».
	+ Si  : « L’appareil est de forte puissance ! ».

Pour programmer les 3 tests sur la puissance, on utilisera deux « if » imbriqués : attention au « : » et aux indentations !

1. Écrire le programme Python.
2. Compléter le tableau à l’aide de ce programme :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Appareil | Fer à repasser | Lampe led | Chauffage appoint | Plaque cuisson | Veilleuse bébé | Lampe Halogène |
| **U en V** | 220 | 10 | 220 | 220 | 220 | 220 |
| **I en A** | 5 | 0,5 | 4 | 25 | 0,01 | 0,5 |
| **P en W** |  |  |  |  |  |  |
| **Message obtenu** |  |  |  |  |  |  |

**Exercice n°1 :**

**C)** Écrire en Python la suite d'instruction qui affiche une alerte si la température (on utilisera la variable « température ») dépasse 40 °C.

**A)** Qu'affiche ce programme si x=100, puis si x=200?

if x<=150:

print (x+10)
else:

print (x-30)

**B)** On donne le programme suivant :

if x>100:

print("x est grand")
if x>200:

print ("x est très grand")
else:

print ("x est petit")

Qu'affiche ce programme lorsque x= 150, puis x=300 ?

**Conclusion : les instructions composées ont une structure bien définie en Python :**

**En tête**, **double point**, **bloc d'instructions indenté** :

**Quelques opérateurs de comparaison : (Pour info)**

 **<** strictement inférieur

 **>** strictement supérieur

 **<=** inférieur ou égal

 **>=** supérieur ou égal

 **= =** égal **(Attention à ne pas confondre avec = qui est une affectation !)**

. **!=** différent

Il est possible d’associer/combiner des conditions.

*Condition1* ***ET*** *Condition2 (syntaxe Python :* ***and****)*

*Condition1* ***OU*** *Condition2 (syntaxe Python :* ***or****)*

1. **Des structures de boucles : boucles bornées et non bornées.**

Tous nos traitements jusqu'alors sont effectués une seule fois. On peut cependant souhaiter les répéter.

Deux types de boucles viennent rapidement à l'esprit : celles où l'on connait, avant le traitement, combien de fois elles seront effectuées (**elles sont dites bornées**) et celle dont la terminaison dépend du traitement lui-même et devra donc être évaluée pour déterminer s'il faut quitter la boucle **: elles sont dites non bornées. Il est important de distinguer ces deux boucles.**

1. **La boucle bornée : "pour i de … à … "**

**Exercice n°2 : une table de multiplication.**

On souhaite Ecrire un programme qui affiche les 10 premiers termes de la table de multiplication par 7.

Tester ces trois propositions :

|  |  |  |
| --- | --- | --- |
| for i in range(10): print(i\*7) | for i in range(1,10): print(i\*7) | for i in range(1,11): print(i\*7) |

Laquelle vous semble mieux convenir ?

**Attention à ce piège en Python :**

* **for i in range(10) : de 0 à 9 inclus**. (Le 0 est sous-entendu…)
* **for i in range(1,10) : de 1 à 9 inclus** (le 10 est exclu !)
* **for i in range (1,11) : de 1 à 10 inclus !**
* **for i in range(1,11,2) : de 1 à 10 par pas de 2 : 1,3,5,7,9 !**

**Exercice n°3 :** écrire un programme qui affiche une suite de 8 nombres dont chaque terme soit égal au triple du terme précédent.

**Exercice n°4 : deux boucles imbriquées !**

Ecrire un programme qui affiche les 10 premiers termes des 6 premières tables de multiplication !

1. **La boucle "tant que" : while.**

**Exercice n°5 :** Une citerne contient 1000 litres. Du fait de l'évaporation elle perd 3 % de son contenu chaque jour.

Au bout de combien de jours contiendra-t-elle moins de 100 litres ?

Ecrire un script en Python afin de résoudre cet exercice.

|  |  |
| --- | --- |
| **Algorithme à compléter :** | **Son codage en Python à construire avec …** |
| Citerne ← … Jours ← … Tant que Citerne … Citerne ← Jours ← Afficher la Citerne  FinTantQue Afficher Jours | On utilisera la boucle while while test :  Instruction 1 Instruction 2 |

***A l’aide du logiciel EduPython, vérifier le script.***

**Exercice n°5 :** Déterminer le nombre de tours effectués par la boucle.

for i in range(5,20,2):

print(i)

**Exercice n°6 :** Compléter le programme Python comptant de5 en 5 entre 20 et 100.

for i in range (...) :

print(i)

**Exercice n°7 :** Combien de tours effectue cette boucle non bornée ?

x=0

while x<10:

x=x+3

1. **Applications**
2. **Compléter le script en Python correspondant au script Scratch ci-dessous :**

1 from random import \*

2 NombreMystere = randint(0,50)

3 …………………………………………………….…………...

4 while (NombreMystere !=reponse):

5 …………………………………..……………………………...

6 …………………………………………………..……..……...

7 …………………………………………………………..……...

8 …………………………………..……............................

9 …………………………………..…….............................

10 …………………………………..……............................

1. **Écrire le programme Python du script Scratch ci-dessous :**

