Synthèse programmation

Langage de programmation

Un **programme** informatique est une suite d'instructions déterminées par le programmeur pour répondre à un problème (jeux, application, système réel, …).

Il est mis au point dans un **langage** de haut niveau (scratch, python, C, ...), testé et corrigé, puis traduit en langage compréhensible par le **microprocesseur** (ou microcontrôleur) dans lequel il sera transféré : sous forme de « 0 » et « 1 » , le code **binaire**.

Algorithme	Langage de programmation graphique	Langage de programmation de haut niveau (ex : Python)	Langage machine (binaire)

i=1
tant que i≤10
Afficher « Je ne dois pas discuter en classe »
i=i+1
		`i = 1`
		`while i<=10:`
		`  print("Je ne dois pas discuter en classe")`
		`  i = i + 1`
			001110101011100
			001010100111110
			101000011101010
			111110111000100
			001010010000111

Les variables

Type de variables :

Types de variables	Code Python	Exemples
Entiers	int (pour integer)	45 ou -9
Nombres à virgule flottante	float	15,34 ou -32,1
Chaîne de caractère	str (pour string)	« Coucou ! » ou « 962 » ou « Bonjour à tous »
Booléens	bool (pour boolean)	2 valeurs possibles : True ou False

Listes	list	`liste1 = [0,1,2,3]`
		`liste2 = [0,5,3,6,5.6]`
		`liste3 = ["bagages","mouton",3,"montre",6.98,"True"]`
		`liste4 = [] --> liste vide`

Affectation et calculs :

Code Python	Signification
i = 6	Affecte à la variable i la valeur 6
i = i +1	Calcule i + 1 et affecte le résultat à la variable i (i augmente de 1)
2**3	Calcule 2 à la puissance 3 (= 2^3 = 8)
14 // 3	Calcule le quotient de 14÷3 = 4
14%3	Calcule le reste (ou modulo) de 14÷3 = 2

Gestion des entrées/sorties :

Afficher à l'écran

`print(a)`	Affiche la valeur de la variable a (si elle existe)
`print("a")`	Affiche la lettre « a »
`print("La valeur de a est :",a)`	Affichage mixte (texte et valeurs)

Interagir avec l'utilisateur

(poser une question et stocker la réponse dans une variable)

x = input("Saisir votre année de naissance :")	Affiche la question à l'utilisateur avec un champ de saisie et attribue la réponse à la variable x

Les fonctions

Dans un programme, il est possible d'écrire de petits programmes ou sous-programmes intermédiaires appelées **fonctions**. Elles permettent de simplifier le programme principal et de décomposer le problème de départ en sous-problèmes.

Définition de fonction	**definir** nom_fonction(liste de paramètres): bloc d'instructions	def compteur(stop) : i = 0 while i < stop: print(i) i = i + 1
Appel de fonction	**appeler** nom_fonction(liste de paramètres)	compteur(4)

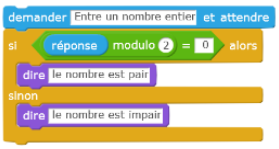
La fonction range() :

for v in ["a","e","i", "o","u","y"] : ....BLOC D'INSTRUCTIONS...	Effectue la boucle, la variable v prenant à chaque tour successivement les valeurs de la liste (ici les voyelles)
for i in range(5) : ....BLOC D'INSTRUCTIONS...	ici, i prend tour à tour les nombres de 0 à 4 (intervalle [0 ; 5[ )
for i in range(6,21) : ....BLOC D'INSTRUCTIONS...	Parcourt l'intervalle [6 ; 21[avec un pas de 1
for i in range(3,11,2) : ....BLOC D'INSTRUCTIONS...	Parcourt l'intervalle [3 ; 11[avec un pas de 2 : 3, 5, 7 et 9

Les conditions

Pour aiguiller dans différentes direction l'éxécution du programme en fonction du souhait de l'utilisateur ou des intéractions avec l'environnement extérieur, il est possible d'utiliser des **instructions conditionnelles :**

Théorie	si condition : instruction A	si condition : instruction A sinon: Instruction B

Exemple	if x >10 :
	Print(x, »est plus grand que 10)
		
		s=input("Entrez un nombre entier")
		s=int(s)
		if s%2==0:
		print("le nombre est pair")
		else:
		print("le nombre est impair")

Opérateurs de comparaison :

Opérateur	==	!=	<	>	<=	>=	or	and
Signification	égal à	différent de	strictement inférieur à	strictement supérieur à	inférieur ou égal à	supérieur ou égal à	ou	et

Les boucles bornées et non bornées

Lorsque des instructions sont répétées plusieurs fois dans un programme, on utilise une **boucle bornée** :

Pour i allant de borne_mini à borne_maxi : instruction A	for i in range(1,4): print("i a pour valeur ",i)

Dans d'autres cas, il est nécessaire de répéter des instructions jusqu'à ce qu'une certaine condition soit vérifiée. On utilisera alors une **boucle non bornée** :

Tant que condition: instruction A	x = 1 while x < 10: print("x a pour valeur ",x) x = x * 2