Structures de base [bs]
Résumé de cours

Université de Haute Alsace

Unisciel algoprog Version 12 mai 2018

Table des matières

1 Python - Résumé de cours

1.1 Le langage

Identifiant
Séquence de lettres (A...Z, a...z), de chiffres (0...9), de lettres accentuées ou du caractère souligné (_). Il doit commencer par une lettre ou un souligné.

La casse
Le langage est *case-sensitif* et les accentués sont autorisés en Python 3 (mais pas en Python 2). Ceci signifie que `cout`, `Cout` et `COUT` réfèrent trois mots différents et `coût` est licite.

1.2 Variables, types et valeurs

Variable
Élément informatique qu'un script peut manipuler.
Décrite par :

- Un **identifiant** unique qui la désigne.

- Un **type** qui définit de quel « genre » est l'information associée.
- Une **valeur** qui doit respecter le type.

Types intégrés

Domaine	Algorithmique	Équivalent Python
$\mathbb{Z}$	`Entier`	`int`
$\mathbb{R}$	`Réel`	`float`
$\mathbb{B}$	`Booléen`	`bool`
$\mathbb{A}$	`Caractère`	—
$\mathbb{T}$	`Chaîne`	`str`

Python : Le type Caractère

Il n'existe pas : un caractère est simplement une chaîne de longueur 1.

Littéraux

- `Entier` : Suite de chiffres éventuellement préfixé par un signe (+ ou −).
- `Réel` : S'écrit en notation décimale ou en notation scientifique.
- `Booléen` : Ils identifient le `Vrai` (mot-clé `True`) et le `Faux` (mot-clé `False`).
- `Chaîne` : Se place entre quotes (') ou entre guillemets (").

1.3 Déclarations

Déclaration de variables

```
nomVar = expression
nomVar1, nomVar2, ... = expr1, expr2, ...
```

1.4 Structure générale

Commentaire orienté ligne

```
... # rend le reste de la ligne non-exécutable
```

Commentaire orienté bloc

```
"""
rend le code entouré non exécutable...
"""
```

Bloc

```
Bloc: #<- deux points
   instruction1  #indentation
   instruction2  #même indentation
   ...
```

Structure générale

```
from biblio import des_trucs_utiles
déclaration_des_objets_globaux
déclarations_et_définitions_de_fonctions_utiles

def PGPrincipal():
   corps_du_programme

PGPrincipal()
```

1.5 Interactions avec l'extérieur

Saisie de données

```
nomVarI = input(["Invite"])        # lecture d'une chaîne
nomVarI = int(input(["Invite"]))   # entrée typée d'un entier
nomVarI = float(input(["Invite"])) # entrée typée d'un réel
nomVarI = bool(input(["Invite"]))  # entrée typée d'un booléen
nomVar1, ..., nomVarN = eval(input(["Invite"])) # typage dynamique
```

Affichage de résultats

```
print(expr1,...,exprN,end="") # SANS retour de ligne
print(expr1,expr2,...,exprN)  # AVEC retour de ligne
```

1.6 Expressions algébriques

Expression, opérandes, opérateurs

Éventuellement accompagnés de parenthèses, une **expression** est une séquence « bien formée » (au sens de la syntaxe) d'**opérandes** (valeurs littérales, variables ou expressions) et d'**opérateurs** destinée à l'évaluation.

Opérateurs arithmétiques

Opérateur Mathématique	Signification	Équivalent Python
+	(unaire) valeur	`+a`
-	(unaire) opposé	`-a`
+	addition	`a + b`
-	soustraction	`a - b`
*	multiplication	`a * b`
/	division décimale	`a / b`
div	division entière	`a // b`
mod	modulo (reste de la division entière)	`a % b`
^	élévation à la puissance	`a ** b`

Ordre de priorité des opérateurs arithmétiques

Comme en mathématique :

1. Les opérateurs unaires (+, −) (priorité la plus élevée)
2. L'opérateur d'exponentiation () (s'il existe)
3. Les opérateurs multiplicatifs (∗, /, div, mod)
4. Les opérateurs additifs (+, −) (priorité la plus basse)

La règle d'associativité s'applique en cas d'ambiguïté entre opérateurs du même ordre de priorité.

Règle de promotion

Pour qu'une opération numérique binaire $(+, -, *, /)$ puisse s'effectuer, il faut que ses deux opérandes soient du **même type** ou d'un **type compatible**. Lorsque ce n'est pas le cas, il y a **promotion** de l'opérande de type le plus faible vers le plus grand.

Fonctions mathématiques

Elles agissent sur des paramètres à valeurs réelles et donnent un résultat réel.

Pour les utiliser

```python
import math
```

1.7 Affectation interne

Affectation interne

```python
nomVar = expression
```