Programmation Fonctionnelle I, Printemps 2017 – TD1

http://deptinfo.unice.fr/~roy

Exercice 1.1 a) Sur le plan du développement décimal, comment caractériseriez-vous les nombres *rationnels* parmi les nombres *réels* en mathématique ? Est-ce identique en Scheme ?

b) Avec quelle fonction primitive calculeriez-vous $\sqrt{2}$ en Scheme ? La valeur obtenue est-elle :
rationnelle ? réelle ? exacte ? inexacte ?

c) Donnez un exemple de nombre *réel irrationnel* en Scheme.

Exercice 1.2 Imaginez que vous êtes au toplevel de Racket :

a) Demandez la valeur de la variable `π` [Rép: `3.14`...] ? Cette valeur est-elle un nombre rationnel ? Calculez l'aire d'un cercle de rayon `3/2` [Rép: `7.06`...].

b) Demandez la valeur de la variable `e` [Rép : 2.718…], base des logarithmes népériens.

c) Calculez la racine cubique de `10` [Rép : `2.15`...]. Vous pourrez utiliser la fonction puissance `(expt x y)`.

d) Calculez le logarithme népérien de `10` [Rép : `2.30`...], puis son logarithme en base `2` [Rép : `3.32`...]. Quel est le logarithme en base 2 de 1024 ?

e) Calculez une mesure approchée de l'angle en radians, dans `[0,π/2]`, dont le cosinus vaut `0.746`. Quelle en est la mesure en degrés ? [Rép : `0.72`...rd = `41.75`...degrés]. Utilisez la fonction *arc cosinus* nommée `acos` en programmation.

f) Définissez la constante `pi` : faire comme si cette variable n'était pas prédéfinie [utilisez bien entendu une petite formule trigonométrique].

g) Définissez la constante `e`, base des logarithmes népériens. faire comme si cette variable n'était pas prédéfinie [utilisez bien entendu une petite formule utilisant la fonction exponentielle `exp`].

h) La fraction `132/315` est-elle irréductible ? Comment le vérifieriez-vous au toplevel (deux réponses possibles) ?

i) La fonction `(number->string n b)` permet d'obtenir une chaîne de caractères contenant la représentation de l'entier $n \geq 0$ en base `b` [b = 2, 10 ou 16]. A l'aide de la fonction `printf`, faites afficher une phrase indiquant quelle est l'écriture *binaire* de l'entier 2013 puis sa traduction en *hexadécimal* [b = 16]. Inversement, la fonction `(string->number str b)` recalcule l'entier dont on donne l'écriture en base `b` sous forme de chaîne. Rédigez une phrase Scheme l'utilisant.

Exercice 1.3 Définissez les fonctions suivantes, et faites suivre chaque fonction d'une demande de calcul.

a) la fonction constante $k2 : x \mapsto 2$

b) la fonction identité $id : x \mapsto x$ [en réalité, `identity` est une primitive].

c) la fonction `somcarrés` : $(x,y) \mapsto x^2 + y^2$ si `x` et `y` sont des nombres, ou bien `#f` sinon.

d) le prédicat `(multiple? n d)` prenant deux entiers `n` et `d`, et retournant `#t` si et seulement si `n` est un multiple de `d`.

e) le prédicat `(naturel? x)` prenant un objet Scheme `x` de type quelconque, et retournant `#t` lorsque `x` est un entier $\geq$ `0`.

f) la fonction `(volume-sphère R)` retournant le volume d'une sphère de rayon `R` . Ex : `(volume-sphère 1)` → `4.188`…

g) la fonction `(signe x)` prenant un réel x et retournant -1 si x < 0, 0 si x = 0 et 1 si x > 0.

h) En introduisant des *définitions locales* [afin de n'effectuer aucun recalcul !], définissez les fonctions :

$$f(x) = \frac{\sin x^2}{x^2} \quad \text{et} \quad g(x) = \frac{\sin x^2}{x^2 + \sin x^2}$$

i) Une année bissextile est une année divisible par 4, sauf que si elle est divisible par 100, elle doit être aussi divisible par 400. Par exemple, 2000 et 2004 sont bissextiles mais 2100 ne l'est pas. Programmez la fonction `(bissextile? n)` retournant `true` si et seulement si l'année `n` est bissextile.

N.B. *Tu éviteras les **and** dans un **cond**. Cest un conseil d'ami. Pourquoi ?? Expliquez le bien-fondé de ce conseil en étudiant par exemple la logique de la construction :*

```
(cond ((and t1 t2) e1)
      ( •     ; <-- maintenant, quelle est la situation en ce point ?
```

Programmation Fonctionnelle I, Printemps 2017 – TP1

`http://deptinfo.unice.fr/~roy`

Lancez Racket ! Sur Windows, il est dans le menu *Démarrer/Programmation*. Sur Mac, il faut placer la dernière version de Racket dans votre Dock. Agrandissez la fenêtre au maximum !

Une fenêtre de Racket, par exemple celle que vous voyez après le lancement, est composée de deux cadres:

- celui du haut contiendra votre programme. C'est un **éditeur** de programmes Scheme.
- celui du bas contiendra vos **interactions** avec Racket. Vous pourrez y faire de petits calculs ou tester vos programmes « *au toplevel* » comme on dit... Mais en principe les instructions d'affichage de résultats se font en haut. Il est rare d'avoir à travailler « en bas » !

☞ **Vérifiez au toplevel que vous êtes bien dans le langage « Etudiant niveau avancé ». Sinon, ouvrez le menu *Langage*, puis *Sélectionner le langage…*, cliquez dans « *Choisir un langage* », puis « *Langages d'enseignement*», et choisissez *"Etudiant niveau avancé"*, cliquez dans *"Montrer les détails"*, cochez *"write"* et *"fractions mêlées"*, décocher *"Montrer le partage de valeurs"*. Enfin, toujours dans le menu Langages, ajoutez le teachpack *valrose.rkt* que vous avez téléchargé (conservez-le toujours sur votre clé USB)…**

```
Bienvenue dans DrRacket, version 6.6 [3m].
Langage: Etudiant niveau avancé [personnalisé]; memory limit: 1024 MB.
Teachpack: (lib "valrose.rkt" "installed-teachpacks").
```

Exercice 1.1 **Au toplevel**, le *prompt* matérialisé par le signe > attend une expression à évaluer. Calculez par exemple la moyenne des entiers de l'intervalle `[1,10]`:

```
> (/ (+ 1 2 3 4 5 6 7 8 9 10) 10)    ; calcul exact sur des données exactes ⇒ résultat exact !
5.5      ; ou autre ?
```

a) Cliquez avec le bouton droit de la souris sur le résultat 5.5, et demandez à voir d'autres formes d'affichage… Est-ce que 5.5 est un nombre exact ? Vérifiez-le au toplevel en posant la question `exact?`

b) Vous souhaitez faire le même calcul jusqu'à 12 ? Inutile de recopier la première ligne à la main, tapez sur *Esc-p* [*Previous*], et corrigez… Le toplevel se « souvient » de vos demandes de calculs précédentes dans son *historique*.

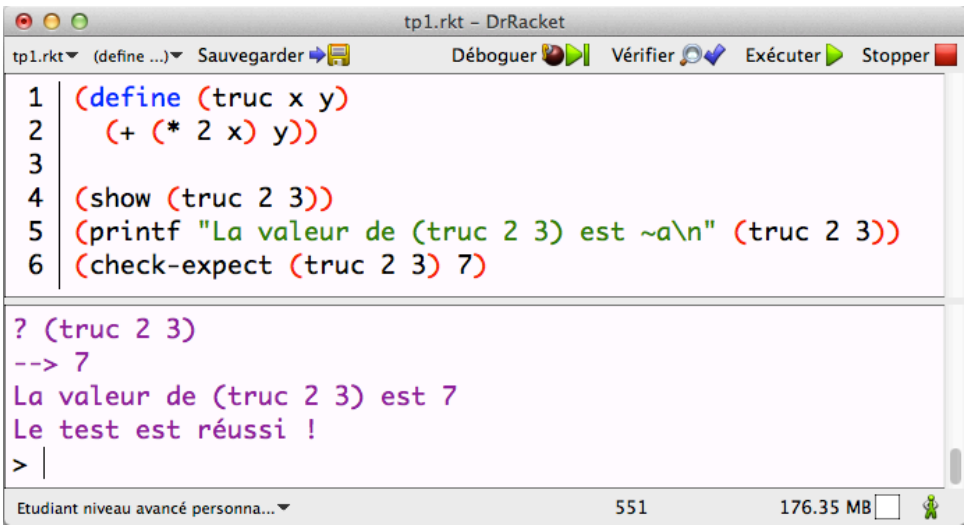
c) Tapez simplement le mot `sin`. Que se passe-t-il ? Comment le comprenez-vous ? Le mot anglais *procedure* est synonyme de *fonction*.

d) Tapez simplement le mot `sinus`. Que se passe-t-il ? Comment le comprenez-vous ?

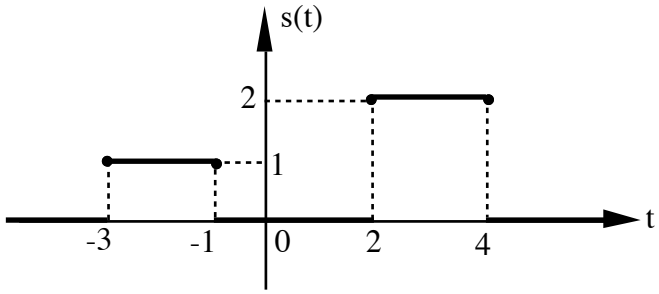
e) Faites un clic droit à l'intérieur du mot `sin` et allez chercher de la doc sur ce mot. Notre langage est « *advanced* ».

☞ **A partir de maintenant, vous travaillez dans le cadre du haut, celui des définitions, et plus au toplevel !**

FONCTIONS. *Définissez les fonctions suivantes dans l'éditeur [cadre du haut]. Dans l'éditeur, vous ferez suivre chaque définition de fonction d'un affichage de résultat avec `printf`, ou plus simplement avec un (show …), ou bien d'un test automatique vec l'une des 3 variantes de `check`. Les résultats tomberont automatiquement au toplevel ! Exemple :*



Exercice 1.2 Définissez avec un `cond` à 5 branches [**on préfère éviter les `and` dans un `cond`**] la fonction `s` représentant un signal dépendant du temps `t` et dont le graphique est donné ci-dessous. Le mot-clé `else` sera obligatoire dans ce cours... Balayez l'axe des abscisses de -∞ à +∞, il y a cinq cas à envisager.



N.B. *Le mot `else` est-il vraiment obligatoire à la fin d'un `cond` ? Essayez d'évaluer l'expression suivante :*

```
(cond ((> 2 3) 4)
      ((> 3 4) 5))
```

Exercice 1.3 **a)** Définissez une fonction `(distance x1 y1 x2 y2)` retournant la distance entre deux points du plan `M1(x1,y1)` et `M2(x2,y2)`. Le repère est orthonormé, utilisez *Pythagore*.

b) En déduire une fonction `(point-intérieur? x y a b r)` retournant `true` si et seulement si le point `(x,y)` est strictement à l'intérieur du cercle de centre `(a,b)` et de rayon `r`.

Exercice 1.4 **a)** L'aire A d'un triangle quelconque de côtés `a`, `b`, `c` est donnée par la formule $A = \sqrt{p(p-a)(p-b)(p-c)}$ dans laquelle `p` représente le demi-périmètre. Ecrire la fonction `(aire-triangle a b c)` retournant l'aire du triangle. Faites afficher l'aire d'un triangle de côtés 3, 5, 6 [Rép : 7,48…], puis de côtés 1, 2, 3 [devinez de tête avant], puis de côtés 2, 5, 8 [expliquez le résultat]. *Attention, ne calculez pas plusieurs fois le demi-périmètre !*

b) Programmez une fonction `(triangle? a b c)` prenant trois réels `a`, `b`, `c`, et retournant `#t` si et seulement s'il existe un triangle de côtés `a`, `b`, `c`.

Exercice 1.5 Supposons que l'impôt sur le revenu annuel soit calculé *par tranches* de la manière suivante. Un salarié ne paye rien pour les 8000 premiers euros qu'il gagne. Il paye 10% sur chaque euro gagné entre 8000 € et 25000 €, et enfin 20% sur chaque euro gagné au-dessus de 25000 €..

a) Définissez la fonction `(tranche s b h p)` retournant l'impôt dû pour un salaire annuel `s` dans la tranche `[b,h]` dont le pourcentage est p%. Exemple :

```
(tranche 2500 2000 3000 10) → 50        (tranche 4000 2000 3000 10) → 100
```

b) *A.N.* Utiliser la fonction `tranche` pour en déduire la fonction `(impôt s)` retournant l'impôt total calculé par tranches pour un salaire annuel `s` avec les données numériques données au début de l'exercice. Exemple :

```
(impôt 40000) → 4700
```

N.B. Tout comme Python note `float('inf')` le symbole +∞, Scheme le note `+inf.0` (c'est un nombre flottant).

Exercices Supplémentaires

Exercice 1.6 Définissez une fonction `(somme-carrés-max a b c)` prenant trois réels `a, b, c` et retournant la somme des carrés des deux plus grands. Exemple : `(somme-carrés-max 4 1 3)` → `25`

Exercice 1.7 Programmez *par récurrence* sur n ≥ 0 la fonction `(somme n)` retournant la somme des entiers de `1` à `n`. Par exemple `(somme 10)` → `55`. Complétez le schéma suivant :

```
(define (somme n)      ; n entier ≥ 0, retourne 1+2+…+n
  (if (= n 0)
      ...
```

*Une fraction non négligeable de la population étudiante de 1ère année semble ignorer qu'il existe des objets avec des pages nommés **livres**. Depuis leur invention, ils sont le vecteur le plus efficace [avec la transmission orale] pour retenir et faire avancer les connaissances. Commencez à vous construire une bibliothèque scientifique dans les matières que vous étudiez. Oubliez les jokers et autres Annales, le Bac c'est fini. A propos, il existe un livre de cours...*

Annexe : Python vs Scheme - Séance 1

http://deptinfo.unice.fr/~roy

Python et Scheme. Deux langages de programmation, deux styles bien distincts. Pour Python, les boucles et la mutation. Pour Scheme, la récurrence et aucune mutation (la boucle sera un cas particulier de récurrence). En réalité, Scheme contient aussi les mécanismes de mutation, les vraies boucles, etc. Mais ce cours sera axé sur l'aspect purement fonctionnel de Scheme. C'est le cours avancé PF2 du semestre 3 qui abordera les aspects non fonctionnels de Scheme, dont les classes, etc.
Python est *simple* par rapport aux autres langages comme C ou Java. Mais Scheme est encore plus simple. Sa syntaxe est réduite aux parenthèses, sa structure de donnée essentielle est la **liste** mais qui n'est **pas du tout celle de Python** ! La plupart des *pythonneries* comme `[x*x for x in L if x%2 == 0]` sont très facilement simulables en Scheme avec `map` et `filter`. *Voire avec des boucles for sans mutations, intégrées aux dernières versions de Racket mais nous n'en parlerons pas en L1*. **Le but de ce cours en L1 est de se focaliser sur les fondements, pas sur la cosmétique !**
Scheme est un descendant de Lisp (1958, langage d'Intelligence Artificielle). Il est surtout utilisé dans l'enseignement et dans la recherche sur les fondements des langages de programmation (beaucoup de théoriciens). Usage limité dans l'industrie.
En Scheme les variables internes aux fonctions sont globales par défaut, et pour introduire une variable locale, on utilise (en PF1) le mot **local**. En Python, c'est l'inverse : les variables internes aux fonctions sont locales par défaut, et pour introduire une variable globale, on utilise le mot **global**.
Scheme distingue les nombres exacts et les nombres inexacts. En PF1, le nombre 1.25 est exact et identique au rationnel 5/4. Par contre le nombre #i1.25 est inexact. En Python, il faut passer par la classe Fraction pour manipuler des rationnels.

PYTHON	*SCHEME*
`pi = acos(-1)`	`(define pi (acos -1))`

`if pi > 1 :`
`  ...`
`else :`
`  ...`
	`(if (> pi 1)`
	`  ...`
	`  ...)`

`def f(x) :` `  return 2 * x + 1`	`(define (f x)` `  (+ (* 2 x) 1))`
`lambda x : x+3`	`(lambda (x) (+ x 3))`
`print('{} + {} = {}\n'.format(2,3,2+3)`	`(printf "~a + ~a = ~a\n" 2 3 (+ 2 3))`

`>>> 0.1 + 0.1 + 0.1 == 0.3  # nombres approchés`
`False`
	`> (= (+ 0.1 0.1 0.1) 0.3)  ; ici exacts`
	`#t`
	`> (= (+ #i0.1 #i0.1 #i0.1) #i0.3)  ; ou approchés`
	`#f`

`>>> from fractions import Fraction`
`>>> 2 * Fraction(1,2)`
`Fraction(1,1)`
	`> (* 2 1/2)  ; cool`
	`1`

`>>> (2+3j) * (2-3j)` `(13+0j)`	`> (* 2+3i 2-3i)  ; cool` `13`

`>>> from math import pi`
`>>> type(pi) == float`
`True`
	`> (and (real? pi) (complex? pi))`
	`#t`

`from random import randint` `randint(0,n-1)  # un entier aléatoire de [0,n-1]`	`(random n)  ; un entier aléatoire de [0,n-1]`
`randint(a,b)`	`(+ a (random (+ (- b a) 1)))`
`p if q else r`	`(if p q r)`
`if p : q` `else : r`	`(if p q r)`

`def couleur() :  # une couleur au hasard`
`  hasard = randint(0,2)`
`  if hasard == 0 : return "red"`
`  elif hasard == 1 : return "yellow"`
`  else : return "blue"`
	`(define (couleur)  ; une couleur au hasard`
	`  (local [(define hasard (random 3))]`
	`  (cond ((= hasard 0) "red")`
	`  ((= hasard 1) "yellow")`
	`  (else "blue"))))`

C'est tout pour aujourd'hui...