Premier pas en CAML

Tous les fichiers peuvent être trouvés sur

`http://algo-td.infoprepa.epita.fr/`

Mise en route

Certains exemples à tester dans cette première partie présentent des notions nécessaires pour la suite (y compris les qcm, éventuellement...).

Exercice 1 (Étranges)

Pour cet exercice, commencer par récupérer le fichier `practical1_eval.ml` en ligne. Ouvrir ensuite ce fichier sous `emacs` / `CAML` .

Évaluer ces phrases, observer et expliquer les résultats.

Exercice 2 (Fonctions)

Récupérer le fichier `practical1_functions.ml` en ligne. Ouvrir ensuite ce fichier sous `emacs` / `CAML` .

— La première partie montre le fonctionnement des fonctions à plusieurs paramètres. Elle utilise la directive `#trace` expliquée ci-dessous.
— Les exemples de la deuxième partie expliquent la portée "statique"...

Évaluer ces phrases, observer les résultats et essayer de comprendre les règles.

Quelques explications issues du manuel CAML sur les directives utilisées

> Note : all directives start with a # (sharp) symbol. This # must be typed before the directive, and must not be confused with the ♯ prompt displayed by the interactive loop. For instance, typing `#quit;;` will exit the toplevel loop, but typing `quit;;` will result in an "*unbound value quit" error.*
>
> `#trace` *function-name* `;;`
>
> After executing this directive, all calls to the function named *function-name* will be "traced". That is, the argument and the result are displayed for each call, as well as the exceptions escaping out of the function, raised either by the function itself or by another function it calls. If the function is curried [a], each argument is printed as it is passed to the function.
>
> `#untrace` *function-name* `;;`
>
> Stop tracing the given function.

a. Ce mot sera expliqué plus tard !

Exercice 3 (Exceptions)

Récupérer le fichier `practical1_exceptions.ml` en ligne.

Étudier ces exemples d'utilisation des fonctions `failwith` et `invalid_arg` pour déclencher des exceptions. Ces fonctions devront être utilisées à partir de maintenant pour gérer les cas d'erreur.

> `val invalid_arg : string -> 'a`
>
> Raise exception Invalid_argument with the given string.
>
> `val failwith : string -> 'a`
>
> Raise exception Failure with the given string.

Énigmes

Exercice 4 (Nombre mystère)

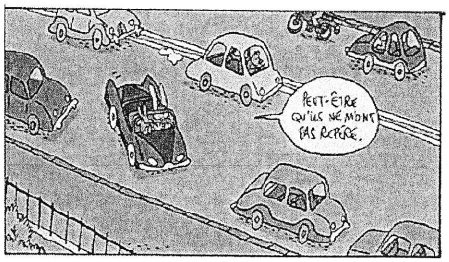
Soit un entier positif à 3 chiffres. La somme des trois chiffres ne peut dépasser 10. Le produit des deux premiers chiffres est inférieur au dernier chiffre. Le chiffre du milieu est égal à la différence du dernier par le premier.

Écrire une fonction CAML qui détermine si un entier donné en paramètre satisfait toutes ces conditions.
La fonction devra déclencher une exception si le paramètre n'est pas valide (n'est pas positif et/ou ne contient pas trois chiffres !).

Question subsidiaire : un tel nombre existe-t'il ?

Exercice 5 (Délit de fuite)

Trois agents de la circulation ont remarqué qu'un automobiliste avait enfreint le code de la route. Aucun d'eux n'a retenu le numéro (à 4 chiffres) qui figurait sur la plaque minéralogique, mais chacun d'eux a retenu une particularité de ce nombre.



Un des agents se rappela que les deux premiers chiffres étaient identiques, l'autre que les deux derniers chiffres étaient identiques. Enfin le troisième (qui était un peu mathématicien) a affirmé que ce nombre de quatre chiffres était un carré parfait.

Écrire une fonction CAML qui détermine si un entier donné est positif à 4 chiffres et peut correspondre à la plaque.
Une exception devra être déclenchée si l'entier n'est pas un nombre positif à quatre chiffres.

Fonctions utiles :
- `val sqrt : float -> float` Square root.
- `val int_of_float : float -> int` Truncate the given floating-point number to an integer.
- `val float_of_int : int -> float` Convert an integer to floating-point.

Question subsidiaire : quelle est la valeur de la plaque ?

Exercice 6 (La surface du jardin)

Paul et Virginie considèrent la surface totale du jardin (nombre entier de m^2). Paul diminue ce nombre de 5000. Virginie écrit ensuite 2 fois côte à côte le nombre à 3 chiffres restant. Puis Paul divise ce nouveau nombre par 7 et lui retranche 7 fois le reste entier de la division. Virginie divise alors ce résultat par 11, et lui retranche 11 fois le reste entier de la division. Paul divise enfin ce dernier nombre par 13 et lui retire 13 fois le reste entier de cette dernière division. Il reste alors 555.

Écrire une fonction CAML qui teste si un entier donné peut correspondre à la surface du jardin de Paul et Virginie.

Question subsidiaire : quelle est la surface du jardin ?

Dates

Exercice 7 (Lendemain)

Écrire une fonction qui calcule la date du lendemain à partir d'une date donnée. Les dates (paramètre et résultat) seront des chaînes de caractères de la forme "jj/mm/aaaa". La fonction devra tester si la date est valide.

```
# tomorrow "31/12/2017" ;;        (* French shape *)
 - : string = "01/01/2018"
# tomorrow "45/02/2005" ;;
 Exception: Failure "tomorrow:not a valid day".
```

Plus : Et si on veut pouvoir gérer des dates de la forme "24 décembre 2017"

```
# tomorrow "24 décembre 2017" ;;
 - : string = "25 décembre 2017"
```

Encore plus [1] **:**

```
# tomorrow "fr" "31 décembre 2017" ;;
 - : string = "1 janvier 2018"
# tomorrow "en" "December 31, 2017" ;;
 - : string = "January 1, 2018"
```

Rappels :

```
# let s = "Hello" ^ " " ^ "World" ;;
 val s : string = "Hello World"

# String.length s ;;
 - : int = 11
```

Autres fonctions utiles du module `String`

val sub : string -> int -> int -> string

`String.sub` *s start len* returns a fresh string of length *len*, containing the substring of *s* that starts at position *start* and has length *len*.

Raise `Invalid_argument` if *start* and *len* do not designate a valid substring of *s*.

val int_of_string : string -> int

Convert the given string to an integer. The string is read in decimal (by default) or in hexadecimal (if it begins with 0x or 0X), octal (if it begins with 0o or 0O), or binary (if it begins with 0b or 0B).

Raise `Failure "int_of_string"` if the given string is not a valid representation of an integer, or if the integer represented exceeds the range of integers representable in type int.

1. Toujours plus : voir les bonus en ligne !