Premier pas en Caml
Soyez fainéants MAIS efficaces !

Mise en route

Exercice 1 (Étranges)

```
max_int ;;
max_int + 1 ;;
min_int ;;
min_int * 2 ;;
max_int * 2 ;;

exp 40. = exp 40. +. 1. ;;     (* exp: exponential function *)
exp 40. = exp 40. +. 15. ;;
exp 40. = exp 40. +. 16. ;;
```

Évaluer ces phrases, observer et expliquer les résultats.

Exercice 2 (Fonctions)

Récupérer le fichier `practical1_functions.ml` sur gitlab. Ouvrir ensuite ce fichier sous `ocaml-top` (`emacs`).

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

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 *myst* ayant les propriétés suivantes :
- *myst* est un carré parfait [1] ;
- la somme des carrés des deux plus grands chiffres de *myst* est également un carré parfait.

Écrire la fonction Caml `mystery:int->bool` qui détermine si un entier donné est le nombre mystère *myst*. La fonction devra déclencher une exception si le paramètre n'est pas un nombre positif à trois chiffres.

```
# mystery 123 ;;
- : bool = false
# mystery 12345 ;;
Exception: Invalid_argument "mystery: "not a 3-digits integer".
```

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 : un tel nombre existe-t'il ?

Exercice 5 (Crack the Code)

Dans l'école, il existe un coffre fort qui contient tous les sujets d'examens. Le code C est un constitué de 4 chiffres tels que :
- les 4 chiffres de C sont pairs ;
- la somme des 4 chiffres de C est 8 ;
- C est divisible par 23 ;
- en ajoutant à C son "miroir" (4321 est le miroir de 1234), on obtient un nombre palindrome à 4 chiffres : un entier identique à son "miroir" (par exemple 1991).

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 quatre chiffres !).
Question subsidiaire : quel est ce code ?

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 la fonction Caml `surface` qui teste si un entier donné peut correspondre à la surface du jardin de Paul et Virginie.

```
# surface ;;
- : int -> bool = <fun>
# surface (-50) ;;
Exception: Invalid_argument "surface: negative parameter".
# surface 1234 ;;
- : bool = false
# surface ??? ;;
- : bool = true
```

1. Carré parfait = le carré d'un entier.

Dates

Exercice 7 (Lendemain)

Écrire une fonction qui calcule la date du lendemain à partir d'une date donnée. La fonction devra déclencher une exception si la date n'est pas valide.

Level 1

— En paramètre, une date est représentée par 3 entiers : jour mois année.
— En résultat, une date est représentée par un triplet d'entiers (jour, mois, année).

```
# day_after 12 9 2024 ;;
- : int * int * int = (13, 9, 2024)
# day_after 28 2 2024 ;;
- : int * int * int = (29, 2, 2024)
# day_after 28 2 2025 ;;
- : int * int * int = (1, 3, 2025)
# day_after 31 12 2024 ;;
- : int * int * int = (1, 1, 2025)
# day_after 12 15 2005 ;;
- : int * int * int = (13, 15, 2005)

# day_after 12 15 2005 ;;
Exception: Failure "the day after:not a month number".
# day_after 31 6 2029 ;;
Exception: Failure "the day after:this day does not exist for month 6".
```

Level 2

Les dates (paramètre et résultat) seront des chaînes de caractères de la forme "jj/mm/aaaa".

```
    # tomorrow "31/12/2023" ;;       (* French shape *)
    - : string = "01/01/2024"
    # tomorrow "28/02/2020" ;;
    - : string = "29/02/2020"
    # tomorrow "45/02/2005" ;;
    Exception: Failure "tomorrow:not a valid day".
```

Plus : Et si on veut pouvoir gérer des dates de la forme "December 24, 2020" ?

```
    # tomorrow "December 24, 2023" ;;
    - : string = "December 25, 2023"
```

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.