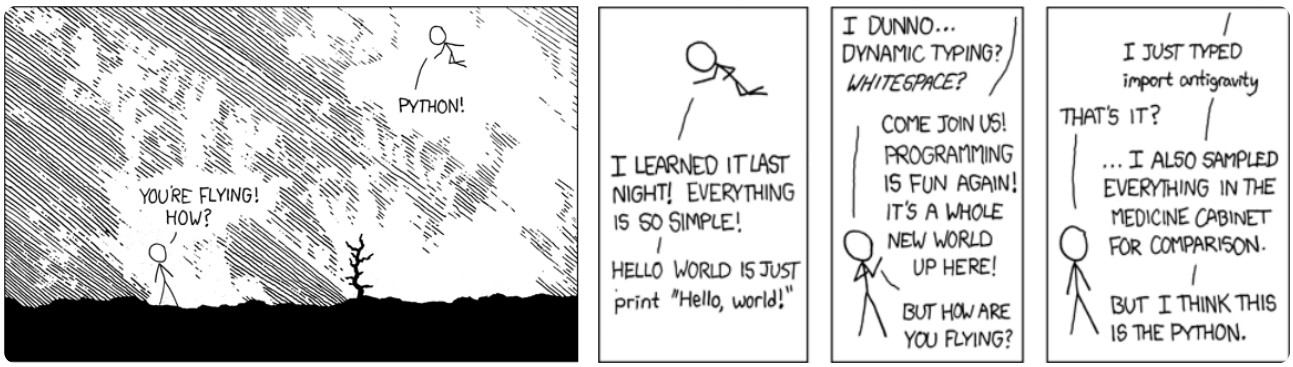


Initiation Python

Auteur : Fabien Roca · **Publié le** 22/10/2019 · 2 vues · 3 téléchargements PDF

Petit à petit le Python fait son entrée dans les salles de classe remplaçant le Turbo Pascal et comme étant la transition normale au langage en bloc, comme Scratch, utilisé dans les écoles primaires et en collège.

Vous trouverez de plus en plus de formations, de tutoriels ou cours en ligne, plus ou moins spécialisés et adaptés.

Cette découverte est faite pour que vous preniez des notes au fur et à mesure que vous croisez des commandes ou des structures.

Vous pourrez, dans un premier temps, vous aider d'interfaces dynamiques présentes en ligne qui pourront vous permettre de réaliser votre programme en blocs et de trouver sa « traduction » en Python. Par contre, il vous faudra faire attention car celles-ci sont souvent prévues pour des cartes programmables par exemple. De plus, le langage en bloc, bien que très pratique n'est pas assez complet pour nous permettre de traiter les notions d'images, ou de web par exemple comparativement à toutes les bibliothèques développées pour Python.

https://fr.vittascience.com/microbit/?lang=fr&mode=mixed

Étapes du projet

ÉTAPE 1

L'interface

Pour programmer en Python, il existe plusieurs interfaces différentes possédant nativement toutes sortes de bibliothèques.

Nous préférerons travailler avec Mu mais il est tout à fait possible de travailler avec n'importe laquelle de ces interfaces. En revanche, si aucun de ceux-ci n'est installé sur votre ordinateur, vous pouvez vous rendre sur des éditeurs en ligne tels que http://brython.info , qui vous permettront de débuter dans de bonnes conditions.

Avant de commencer, il est important de savoir que quelque soit l'interface elle vous demandera de sauvegarder la séquence (programme ou code) avant de l'exécuter : souvent une flèche du style "play" ou "lecture".

ÉTAPE 2

Passons aux choses sérieuses

Un petit exemple valant mieux qu'un long discours, recopier à la lettre le programme ci-contre.

- Établir avec une phrase ce que chaque ligne fait réaliser à l'ordinateur (on réfléchira en terme d'espace mémoire).
- Définir la notion de variable.
- Expliquez ce que fait la commande input

```
1  nom=input("Quel est votre nom ? ")
2  age=int(input("Quel est votre age ?"))
3  annee=2019-age
4  print("Bienvenue ",nom,"vous etes ne en ",annee)
```

ÉTAPE 3

Les données et types de variables

L’avantage, ou l’inconvénient suivant certains, est que Python est capable de reconnaitre (pas toujours mais très souvent) la nature, ou type, de nos données.

Saisissez le programme ci-contre et relevez les réponses transmises par Python.

Vérifiez que ces résultats renvoyés par Python sont compatibles avec ce qui est présenté ci-dessous:

Pour commencer, nous travaillerons principalement avec ces cinq types de données qui sont les entiers (**int** pour integer), les décimaux (**float**), les chaines de caractères (**str** pour string), les listes (**list**) et enfin les booléens (**bool** pour boolean) qui eux ne sont que binaires, ils sont les résultats de tests (vrai ou faux), ou de questions fermées (ouvert ou fermé, blanc ou noir…) .

Une fois les différents types de variables connus il est nécessaire de voir les interactions possibles entre elles. Les deux types qui nous intéresserons en priorité sont les nombres entiers (int pour integer) et les chaines de caractères (str pour string).

```
1  print(type(1))
2  print(type(1.0))
3  print(type('chateau'))
4  liste=['cafe','lait','brioche']
5  print(type(liste))
6  print(type(True))
```

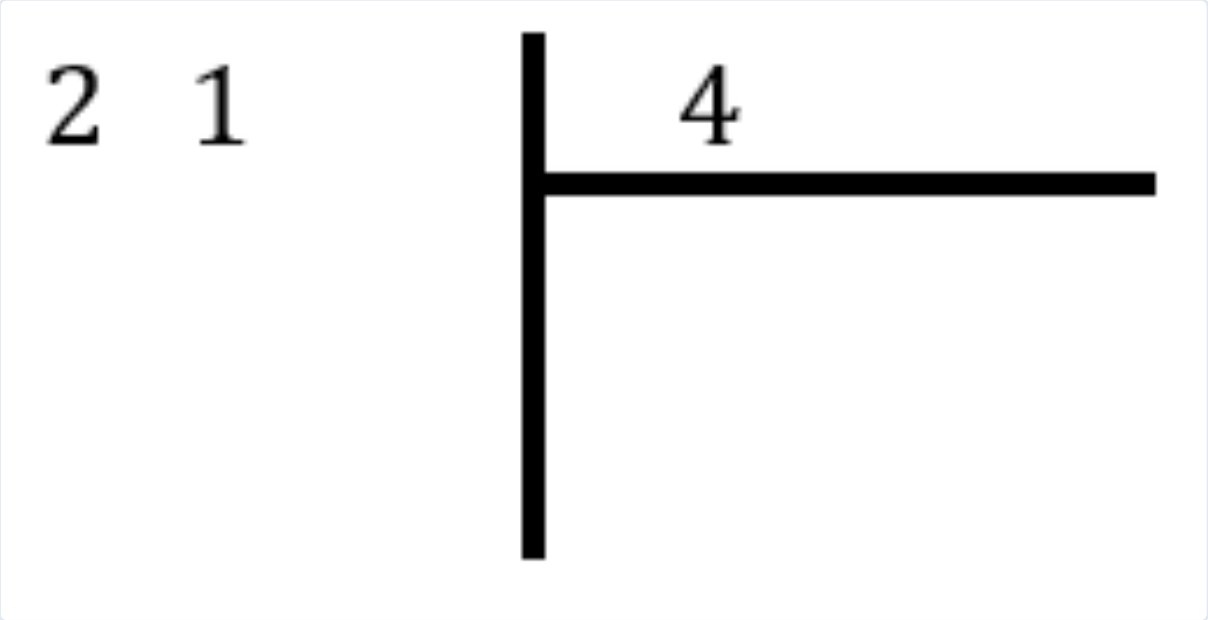
ÉTAPE 4

Variables : Les calculs sur les entiers

Commençons par les entiers. Revenons en enfance et compléter la division euclidienne en n'utilisant que des entiers. Tapez ensuite la séquence des commandes Python.

En déduire ce que renvoie chacune des fonctions / , // et % .

Pour finir, nous travaillerons avec la valeur 12. En saisissant la séquence Python correspondante, déterminez la différence entre la ligne 2 et la ligne 4.




```
1  print(21/4)
2  print(21//4)
3  print(21%4)
```

```
1  valeur=12
2  valeur=valeur+1
3  print(valeur)
4  valeur+=2
5  print(valeur)
```

ÉTAPE 5

Variables : Travail sur les chaines de caractères.

Passons aux chaines de caractères, appelées string. Elles peuvent s'écrire de différentes façons soit entre guillemets, entre apostrophes, ou entre triples guillemets mais pas doubles guillemets ou autre combinaison.

Pour joindre deux chaines de caractères, il suffit de les additionner par exemple **'mais'** + **'son'** formera le mot maison. On peut rajouter par la suite encore une chaîne de caractère à cette première.

De plus, il est également possible de ne récupérer, par exemple, que les quatre premiers termes mais pour le travail sur les suites, nous verrons plus tard.

On parle généralement de concaténation, malheureusement, bien que Python arrive à "comprendre" le type de beaucoup de variables, il ne sait que faire avec une expression telle que :

chaine = **'1'** + **1** TypeError: Can't convert int to str implicitly

En effet, vous souhaitez ajouter le caractère 1 au chiffre 1. Il va donc falloir « convertir » ou plutôt « typer » les variables que nous allons utiliser pour indiquer à Python si nous souhaitons ajouter des chiffres ou joindre deux caractères. Autrement dit il est nécessaire dans ce cas d'expliquer à Python si nous avons une concaténation de strings ou une « simple » addition comme le montre le dernier code ci-dessus.

```
1  "ceci est une chaîne de caractères"
2  'ceci est une chaîne de caractères'
3  """ceci est une chaîne de caractères"""
4  ""ceci est une chaîne de caractères""
```

```
1  chaine='mais'+'on'
2  print(chaine)
3  chaine+='ette'
4  print(chaine)
5  chaine=chaine[0:4]
6  print(chaine)
```

```
1  chaine='1'+ str(1)
2  print(chaine)
3  chaine=int('1')+ 1
4  print(chaine)
```

ÉTAPE 6

Variables : Comparaison

Les comparaisons se font bien entendu à l'aide de comparateurs : égal, différent, supérieur ou inférieur.

Comme on l'a vu précédemment, le signe **=** sert à "**créer une case mémoire** et à y attribuer un nom et une valeur" : ex : banane=6

Il faut donc trouver un signe qui permettrait de comparer deux valeurs , pour cela il nous suffira de doubler le symbole = pour obtenir : **==**

Égal	**==**	**Différent**	**!=**
Supérieur	**>**	**Supérieur ou égal**	**>=**
Inférieur	**<**	**Inférieur ou égal**	**<=**

Dans tous les cas, une erreur (un oubli) du second signe égale vous sera rappelé par l'interface.

SyntaxError: keyword can't be an expression

La réponse à une comparaison est un booléen, l'interpréteur vous répondra toujours True ou False (Vrai ou Faux).

En utilisant **une ligne de code**, tester qui, entre proverbialiser et proverbialement, apparaît en premier dans le dictionnaire.

```
1  print(3==3)
2  print(3>2)
3  print(3<2)
4  print('a'<'b')
5  print('avion'>'avarie')
```

```
True
True
False
True
True
```

ÉTAPE 7

Exemple : IMC : Un indice pour votre santé

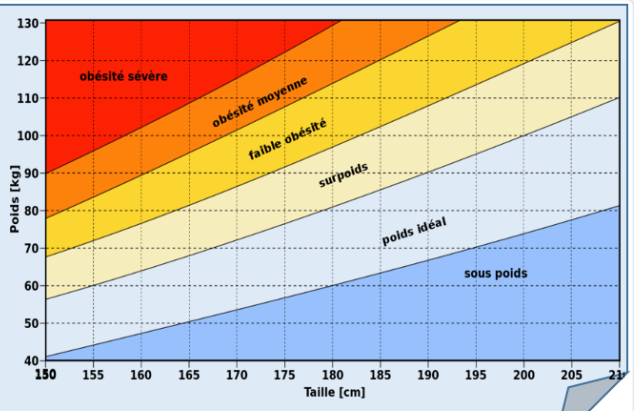
En vous basant sur ce que vous venez de réaliser, et d'après l'encadré ci-contre, réalisez un programme qui calcule et affiche l'IMC de l'utilisateur. (Ne l'effacez pas après vérification, vous le réutiliserez par la suite.)

Remarque : attention aux types de données que vous allez demander (input) à l'utilisateur du programme.

Indice de masse corporelle (IMC)	Interprétation (d'après l'OMS)
moins de 18,5	Insuffisance pondérale (maigreur)
18,5 à 25	Corpulence normale
25 à 30	Surpoids
30 à 35	Obésité modérée
35 à 40	Obésité sévère
plus de 40	Obésité massive

La formule de l'IMC

L'indice de masse corporelle est un calcul simple qui correspond au poids en kilogrammes (kg) divisé par le carré de la taille exprimée en mètres (m^2)



ÉTAPE 8

Les conditions

Parfois il est nécessaire de vérifier si un résultat est juste ou faux (**True** ou **False**) pour adapter la suite du programme ou la réponse à donner à l'utilisateur. Pour réaliser de telles conditions, il faudra utiliser le Si (**if**) et le Sinon (**else**).

Le programme ci-contre est « traduit » ligne par ligne à sa gauche. Pour les nombre 5, 10 et 12, essayez d'anticiper la réponse de l'ordinateur.

Remarque :

Sur les lignes 3, 5, 7 et 9, il y a un « espace » séparant la marge du texte. Cet espacement est appelé indentation, c'est un des principes fondamentaux de Python. Cela signifie que l'action ne sera réalisée que si la condition directement au-dessus est vraie. Il faut le prendre comme un paragraphe qui ne sera lu que si la condition qui le précède est vraie.

Il peut y avoir plusieurs conditions à étudier, pour cela nous ne travaillerons pas avec une suite de **if** mais avec des **elif** interposés. Ces **elif**, diminutifs de « **else if** », peuvent être traduits par « **Sinon si** » et sont utilisés tels qu'on peut le voir dans le second code ci-contre. A nouveau, pour les nombre 5, 10 et 12, essayez d'anticiper la réponse de l'ordinateur.

Remarque : comme évoqué précédemment, l'indentation (décalage d'une ligne par rapport à la précédente permettant l'affichage sous forme de paragraphes) est très importante. Ne seront effectuées que les actions directement placées sous la condition validée.

Recopiez exactement le programme ci-dessus pour vérifier votre prévision de comportement.

Reprendre le programme sur l'indice de masse corporelle (IMC) rédigé précédemment et rajouter du code pour indiquer la catégorie à laquelle appartient l'utilisateur (insuffisance pondérale, corpulence normale ...)

```
1  nombre = int(input('Donnez un nombre : '))
2  if nombre < 10 :
3      print('Votre nombre est plus petit que dix')
4  else :
5      print('Votre nombre est plus grand que dix')
```

```
Demande une valeur à l'utilisateur : nombre
Si le nombre est plus petit que 10 (True) alors
    Affiche : Votre nombre est plus petit que dix
Sinon (False) :
    Affiche : Votre nombre est plus grand que dix
```

```python
nombre = int(input('Donnez un nombre : '))
if nombre < 10 :
    print('Votre nombre est plus petit que dix')
elif nombre==10:
    print('Votre nombre est dix')
elif nombre <= 15 :
    print('Votre nombre est plus grand que dix mais au plus égal à quinze')
else :
    print('Votre nombre est plus grand que quinze')
```

ÉTAPE 9

Double comparaison

Il peut être intéressant et même souvent nécessaire de comparer deux expressions (plus petit que l'un mais plus grand que l'autre, pair mais pas nul ...) pour cela on utilise les opérateurs **and** (et) , **or** (ou) et **not** (inverse).
L'expression globale aura la forme :

comparaison1 and comparaison2

comparaison1 or comparaison2

A première vue ces opérateurs semblent faciles à manipuler mais souvent on se retrouve dépourvu face aux réponses faites par Python.

Pour l'opérateur **OU**, il suffit qu'une des deux comparaisons soit vraie pour que l'expression totale soit vraie.

En revanche, pour l'opérateur **ET** il faut que les deux comparaisons soient vraies pour que l'expression globale soit vraie.

En déduire les valeurs qui complète les tableaux ci-dessous, aussi appelés tables de vérité.

Tableau pour l'opérateur ET :

ET	True	False
True		
False		

Tableau pour l'opérateur OU :

OU	True	False
True		
False		

```
1  print(1>2 or 2>1)
2  print(1>2 and 2>1)
3  print(not(1>2))
4  print(True or False)
5  print(True and False)
```

```
True
False
True
True
False
```

ÉTAPE 10

Année bissextile

Si l'année n'est pas divisible par 4 alors elle n'est pas bissextile.

Si elle est divisible par 4, elle est généralement bissextile sauf si elle est divisible par 100 et pas par 400.

Réalisez un programme qui demande une année à l'utilisateur et lui indique si elle est bissextile ou pas.

Exemples :

- Bissextiles : 1896 ; 1996 ; 2000 ; 2020
- Non Bissextiles : 1900 ; 1998 ; 2001 ; 2019

ÉTAPE 11

Les boucles

Il existe deux sortes de boucles, les boucles dites **For** et celles dites **While**.

Les boucles qui vont exécuter un nombre donné de fois la même opération, ce sont celles que nous appelons les boucles **Pour** (**for**).

Les boucles qui vont exécuter la même opération jusqu'à ce qu'une condition soit réalisée, elles sont appelées les boucles **Tant que** (**while**)

Recopiez exactement le programme ci-contre pour réaliser dix fois l'affichage d'un message.

Remarque :

Python ne commence jamais à compter à 1 sauf si cela ne lui est pas clairement stipulé. De même, il finit toujours à l'entier directement inférieur à la valeur indiquée.

Modifiez le programme ci-dessus pour qu'il commence à compter à 1 et finisse à 10

```
1 for boucle in range(10):
2     print('Je suis a la boucle n°',boucle)
3
4 boucle = 0
5 while boucle<10:
6     print('Je suis a la boucle n°',boucle)
7     boucle = boucle + 1
```

ÉTAPE 12

Recherche de seuil

total = 1 + 2 = 3

total = 1 + 2 + 3 = 6

total = 1 + 2 + 3 + 4 = 10

...

total = 1 + 2 + 3 + 4 + ... + n > 1 000 000

Réalisez un programme qui indiquera à partir de quelle valeur, notée **n** la somme total est strictement supérieure à un million.

ÉTAPE 13

Le Juste Prix

L’ordinateur choisit un nombre au hasard entre 1 et 1000, il vous demande de proposer une valeur, **si** cette valeur est plus petite que la valeur choisie, celui-ci vous indique « C’est plus », **en revanche, si** cette valeur est plus grande, il vous indique « C’est moins ». Le jeu se continue **tant que** la valeur choisie par l'utilisateur est différente de celle choisie par l'ordinateur.

Un message félicitera l'utilisateur lui indiquant le nombre qu'il a trouvé.

Remarque :

le programme ci-contre permet de générer un nombre appelé n qui sera compris entre 1 et 6. La fonction exacte sera utilisée plus loin dans la fiche.


```
1  from random import randint
2  n = randint(1,6)
3  print(n)
```

ÉTAPE 14

Les listes (1) Etudions

Les listes sont des variables très utilisées, elle vont nous permettre de stocker nos données ou d'autres variables mais aussi les trier ou encore les traiter. Elles peuvent contenir plusieurs types de données simultanément. Encore une fois le premier terme est représenté par l'indice 0.

Tout d'abord nous considérons une liste contenant des chaines de caractères (string), une liste d'entiers, des entiers et des flottants. Regardons ce que peut nous afficher Python à partir d'une liste donnée.

Tapez le programme et observez les retours de Python pour chacune des commandes.

```
liste = ['café', 'brioche', 'cacao', [3,2,1], 1, 2, 6.0]
print(len(liste))
print(liste[0])
print(liste[-1])
print(liste[3][0])
print(liste[0][3])
print(liste[0:2])
print(liste[3:])
print('café' in liste)
print(liste.index('brioche'))
print(liste.index('chocolat'))
```

ÉTAPE 15

Les listes (2) Agissons

Repartons de la même liste et essayons de la modifier en supprimant, insérant ou modifiant des valeurs contenues dans la liste précédemment étudiée.

Saisissez le programme ci-contre et observez les retours de Python pour chacune des commandes.

```
1  liste = ['café', 'brioche', 'cacao', [3,2,1], 1, 2, 6.0]
2  liste[1:1] = ['pain']
3  print(liste)
4  liste[2:2] = ['jus','croissant']
5  print(liste)
6  liste[6:] = []
7  print(liste)
```

ÉTAPE 16

Les listes (3) Soyons méthodiques

Les listes étant des objets, il est possible d'appliquer des méthodes qui pourront par la suite nous permettre de gagner du temps dans de nombreux programmes.

```
1  liste = ['café', 'brioche', 'cacao']
2  liste.append('pain')
3  liste.sort()
4  liste.remove('cacao')
5  liste.reverse()
6  liste.pop(1)
7  print(liste)
```

append( ) ajoute une valeur à la fin de la liste.
sort( ) trie la liste.
remove() supprime la première occurrence de la valeur.
reverse() inverse l'ordre de la liste.
pop(i) retire et renvoie l'élément de la liste d'index i.
(Si i est vide, il traite le dernier élément)

ÉTAPE 17

La suite de Fibonacci

C’est une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent. Elle commence généralement par les termes 0 et 1 et ses premiers termes sont : 0, 1, 1, 2, 3, 5, 8, 13, 21 ...

Réalisez un programme qui « sauvera » dans une liste les trente premiers termes de la suite et affiche en fin de programme le 12ème terme.

ÉTAPE 18

Les fonctions

Elles sont utilisées lorsqu'un « bout de code » est utilisé à plus d'un endroit dans le code. Sans les fonctions, le seul moyen de réutiliser ce bout de code serait de le retaper ou de le copier-coller. Les fonctions se trouvent en général en début de code source, après la déclaration des constantes. Une fonction commence par « def » comme définition (de la fonction) et indique le nombre de paramètres qui lui sont nécessaires à son bon fonctionnement.

Il est possible d'importer des fonctions « déjà prêtes » mais nous verrons également que nous pouvons créer les nôtres pour nous faciliter la vie et surtout la lecture du code par la suite.

Comme vu précédemment, on peut essayer la fonction « randint » présente dans le module «random».

```
1  def puissance(nombre,exposant):
2      resultat=nombre**exposant
3      print(resultat)
4      return resultat
5
6  puissance(3,2)
7  puissance(2,3)
```

ÉTAPE 19

L'importation de fonctions

Est-il possible avec la fonction randint d’obtenir le chiffre 1 ? le chiffre 6 ?

Est-il possible avec la fonction randrange d’obtenir le chiffre 1 ? le chiffre 6 ?

Nous pouvons à présent importer de "petites" fonctions qui vont nous permettre de réaliser notre première fonction.

```
1  from random import randint
2  n = randint(1,6)
3  print(n)
```

*Nous n’importons juste que la fonction **randint** du module **random***
On note n le nombre généré aléatoirement.
On affiche la valeur présente dans la variable n.

```
1  from random import randrange
2  n = randrange(1,6)
3  print(n)
```

```
1  from random import randint
2  liste=[]
3
4  def nombre():
5      n = randint(1,10)
6      print('Le numero tire au sort est le ',n)
7      return n
8
9  for k in range(10):
10     liste.append(nombre())
11
12 print('\nLa liste de tous les nombres est \n', liste)
```

ÉTAPE 20

Euromillions

Cette loterie est composée de deux grilles (A et B). La première contient tous les nombres entre 1 et 50. La seconde, B, aussi appelée grilles des étoiles, comportent tous les nombres entre 1 et 12. Le tirage consiste en tirer cinq numéros pour la grille A et 2 étoiles pour la grille B. Le joueur qui mise deux euros pour valider une grille de jeu, gagne au minimum si son ticket présente :

(2 numéros) ou (1 numéro et 2 étoiles).

Réalisez un programme qui demandera à l'utilisateur de rentrer ses numéros et ses étoiles avant de réaliser un tirage Euromillions complet, lui indiquant si sa grille est gagnante ou pas.

ÉTAPE 21

Euromillions (Pour aller plus loin)

En utilisant le tableau moyen des gains ci-contre, vous pourrez donner le gain du joueur au tirage.

En réalisant un grand nombre de tirages aléatoires, vous pourrez déterminer son espérance de gain (à comparer à la somme totale misée).

EUROMILLIONS

Rang	Combinaison	Gain
1	5 numéros + 2 étoiles	Jackpot (17 à 190 M €)
2	5 numéros + 1 étoile	303 798 €
3	5 numéros + 0 étoile	31 448 €
4	4 numéros + 2 étoiles	3 076 €
5	4 numéros + 1 étoile	164 €
6	3 numéros + 2 étoile	104 €
7	4 numéros + 0 étoiles	58 €
8	2 numéros + 2 étoiles	19 €
9	3 numéros + 1 étoile	14 €
10	3 numéros + 0 étoile	12 €
11	1 numéro + 2 étoiles	10 €
12	2 numéros + 1 étoile	8 €
13	2 numéros + 0 étoile	4 €