# Python from Scratch



Sébastien Limet

Laboratoire d'Informatique Fondamentale d'Orléans (LIFO)
IUT d'Orléans

29 novembre 2019


# Plan

# Contexte

## Groupe de travail IREM 2018-2019

Informatique du collège au lycée: passage de Scratch à Python

- Comment préparer les collégiens à l'informatique du lycée?
- Comment les professeurs de lycée pourraient réutiliser les acquis du collège?
- 5 réunions avec 4 professeurs de collège 2 professeurs de lycée
- Production: quelques fiches d'activité plutôt pour les élèves de collège

# Contexte

## Conclusions du groupe de travail

- Le lien fort entre Scratch et Python: les structures de base de la programmation
- Travailler sur les structures algorithmiques:
  - Les variables et leurs valeurs
  - Le pseudo-code
  - Le déroulé d'un programme (notion d'environnement)

# Contexte

## Conclusions du groupe de travail

- Le lien fort entre Scratch et Python: les structures de base de la programmation
- Travailler sur les structures algorithmiques:
  - Les variables et leurs valeurs
  - Le pseudo-code
  - Le déroulé d'un programme (notion d'environnement)

## Extrait programme Algorithmique et Programmation du Collège

L'attendu de fin de cycle évoquant un programme **simple**, il n'est pas question de formaliser d'aucune manière la distinction algorithme/programme. En particulier, il ne s'agit pas de définir un langage de description d'algorithme (ni pseudo-langage ni organigramme) même si son utilisation peut être envisagée, ponctuellement, mais sans souci de normalisation.

# Programme de la formation

## Hétérogénéité du public

- Personnes connaissant Scratch mais peu Python
- Personnes connaissant Python mais peu Scratch
- Personnes connaissant les deux...

$\Rightarrow$ Attentes différentes de chacun

## Parti pris

- Présentation en parallèle d'un noyau commun aux deux langages

$\Rightarrow$ Donne une vision des éléments de transition d'un langage à l'autre

$\Rightarrow$ Permet de voir le type d'exercices possibles pour la transition (en fin de cycle 4 et en début lycée)

# Programme de la formation

## Programme

- Présentation Scratch/Python ($\approx$ 1h)
- Fiche d'activité permettant ($\approx$ 2h)
  - soit d'appréhender l'un ou l'autre des deux langages
  - soit de donner des idées de fiches d'activités pour les élèves

# Plan

# Les types de données

## Scratch

Deux principaux types

- les nombres: `12`, `14.2`
- les chaînes de caractères: `bla bla`

les constantes s'écrivent telles quelles et le type est déduit

## Python

Quatre types de base

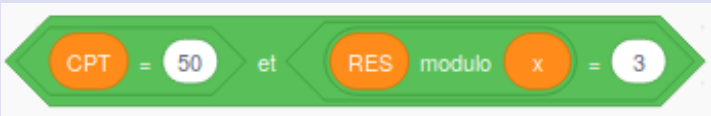
- `int` les entiers: `12`
- `float` représente les réels (Attention approximation) `14.2`
- `bool` les booléens: `True`,`False`
- `str` les chaînes de caractères: `'coucou'`, `'hello'`

Le type est déduit de la syntaxe utilisée: `'12'` est une chaîne de caractères

# Les expressions


## Scratch

- Liste d'opérateurs limitée
- Construction graphique des expressions
- Typage entre booléens et non booléens par formes géométriques



# Les expressions

## Python

Une expression a un type et une valeur (résultat du calcul)

- les constantes et les variables sont des expressions
- Les parenthèses:
  - si `exp` est une expression de type `T` alors `(exp)` est une expression du type `T`
  - Exemples: (4+5)  ('cou'+'cou')
- Les opérateurs unaires:
  - si `exp` est une expression de type `T` et `op` un opérateur unaire du type `T` alors `op exp` est une expression du type `T`
  - Exemples: not True  - 4
- Les opérateurs binaires:
  - si `exp1` et `exp2` sont des expressions de type `T1` et `T2` respectivement; si `op` un opérateur du type `T` acceptant `T1` et `T2` comme arguments alors `exp1 op exp2` est une expression du type `T`
  - Exemples: 12<14.6  (12+5)*2

# Évaluation des expressions (Python)

## Ordre d'évaluation

- Comme en math, les expressions sont évaluées suivant les priorités des opérateurs et des parenthèses

## opérateurs et priorité

1. `or`
2. `and`
3. `not`
4. `in, not in`
5. `==, <, <=, >, >=, !=`
6. `+, -`
7. `*, /, //, %`
8. `**`

# Évaluation des expressions (Python)

## Typage des expressions

- Lors de l'évaluation, on détermine aussi le type de l'expression
- Attention: un même opérateur peut correspondre à différentes opérations suivant le type des opérandes

## Opérateurs et types

- Les nombres
  - `+, -, *, **` Pas de surprise, si au moins une opérande est de type `float` le résultat sera un `float`.
  - `/` Attention retourne toujours un `float` même si le résultat tombe juste
  - `//` partie entière par défaut de la division ($\approx$ division entière). Le typage est `float` si une des deux opérandes est `float`
  - `%` $\approx$ reste de la division entière. Même typage que `//`

# Évaluation des expressions (Python)

## Typage des expressions

- Lors de l'évaluation, on détermine aussi le type de l'expression
- Attention: un même opérateur peut correspondre à différentes opérations suivant le type des opérandes

## Opérateurs et types

- Les booléens
  - `and, or, not` Pas de surprise si les deux opérandes sont des `bool` le résultat est de type `bool`

# Évaluation des expressions (Python)

## Typage des expressions

- Lors de l'évaluation, on détermine aussi le type de l'expression
- Attention: un même opérateur peut correspondre à différentes opérations suivant le type des opérandes

## Opérateurs et types

- Les chaînes de caractères
  - `+` entre deux `str` retourne un `str`, c'est la concaténation
  - `*` si une des deux opérandes est un `int` et l'autre un `str` le résultat est un `str`. C'est la duplication.

# Évaluation des expressions (Python)

## Typage des expressions

- Lors de l'évaluation, on détermine aussi le type de l'expression
- Attention: un même opérateur peut correspondre à différentes opérations suivant le type des opérandes

## Opérateurs et types

- Les comparaisons
  - `==`, `<`, `<=`, `>`, `>=`, `!=` On peut comparer des nombres entre eux et des chaînes de caractères entre elles. Le résultat est de type `bool`
  - `in`, `not in` On peut tester si une valeur est dans une liste de valeurs. Le résultat est de type `bool`

# Les variables

## Rappel

La principale activité d'un programme est de faire des calculs et de stocker ses résultats dans la mémoire pour effectuer d'autres calculs.

## Variable: Définition

Une variable est un nom associé à une zone de la mémoire. Ce nom permet de stocker une valeur ou de consulter la valeur qui se trouve dans la zone concernée. La valeur contenue dans la zone mémoire de la variable est appelée valeur de la variable.

# Les variables

## Scratch

- Création par menu 

- Nom des variables quelconque
- Portée des variables: globale ou locale à un sprite 

## Python

- Création lors de la première affectation
- Nom: une suite de caractères composée
  - de lettres alphabétiques (Éviter les caractères accentués!),
  - de chiffres et
  - du caractère _

  et qui commence par une lettre ou par _
  Exemples: `x y1 y2 ma_var _hello`
- Attention! 33 noms sont réservés pour le langage Python
- Portée: globale ou locale à une fonction

# L'affectation (donner une valeur à une variable)

## Scratch


- `mettre var à exp` donne la valeur de `exp` à `var`
- `ajouter exp à var` modifie la valeur de la variable `var` en y ajoutant `exp`

## Python

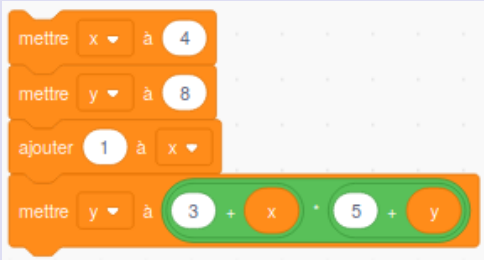
- `var = exp` donne la valeur de `exp` à `var`
  - Si `var` n'existe pas elle est créée sinon elle est modifiée
  - Le signe `=` se lit prend pour valeur et n'est pas symétrique
- `var += exp` modifie la valeur de la variable `var` en y ajoutant `exp`
  `var` doit exister pour utiliser cette instruction
- Il existe aussi `-=`,`*=` et `/=`

# La séquence

Les instructions sont exécutées dans l'ordre où elles sont écrites

## Scratch

- Les instructions s'emboîtent les unes dans les autres
- **Attention!** en Scratch, exécution parallèle de code assez naturelle



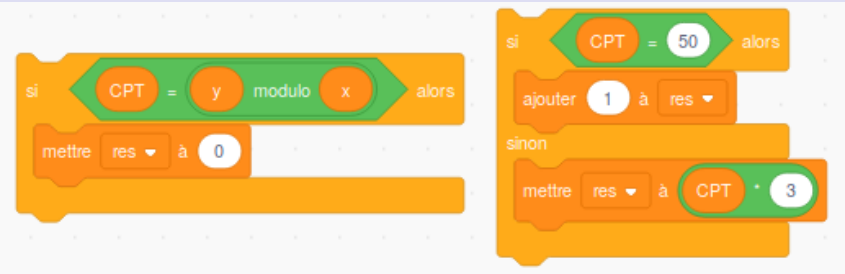

## Python

```
x=4
y=8
x=x+1
y=(3+x)*(5+y)
```

- Une instruction par ligne
- Pas de parallélisme naturel

# Conditionnelle

## Scratch




- Les conditionnelles peuvent s'imbriquer

## Python

```
if CPT==y%x:
    res=0
if CPT==50:
    res+=1
else:
    res=CPT+3
```

- les `:` sont très importants tout comme l'indentation
- le `else` est facultatif

# La boucle bornée

## Scratch

- Le nombre de tours de boucle est connu à l'avance
- Si le nb de tours est une expression, elle est évaluée qu'une fois


## Python

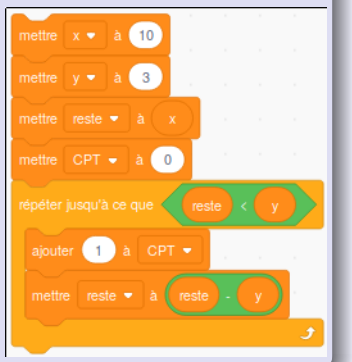
```
x=5
res=0
for i in range(10):
    res+=x
```

- la variable `i` est créée et prendra successivement les valeurs de 0 à 9.
- si le nb de tours est une expression, elle est évaluée qu'une fois

# Boucle générale

## Scratch

- On boucle jusqu'à ce que la condition soit vérifiée
- Le nb de tours de boucle n'est pas connu à l'avance. Il peut être infini



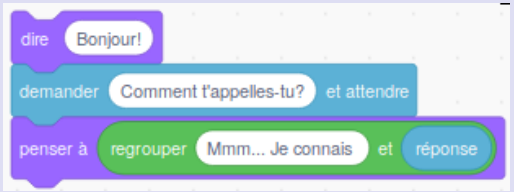

## Python

```
x=10
y=3
reste=x
cpt=0
while reste>=y:
    cpt+=1
    reste-=y
```

- On boucle tant que la condition **n'est pas vérifiée**
- Le nombre de tours de boucle peut être infini

# Les interactions avec l'utilisateur

## Scratch




- `demander` affiche un message et attend la réponse de l'utilisateur
- Le résultat est stocké dans une variable réponse
- D'autres interactions (souris, clavier) mais pas traité ici
- `dire` ou `penser` affiche un message sous forme de bulle

## Python

```
print("bonjour") # affiche un message
reponse=input("Comment t'appelles-tu? ")
#reponse est de type str
print("Mmm... Je connais",reponse)
print("Mmm... Je connais "+reponse)
```

# Plan

# Un premier point

## Scratch vs Python

- On a vu un sous ensemble de Scratch et de Python qui permet de travailler sur les algorithmes de base
- On peut identifier quelques difficultés dans le passage d'un langage à l'autre
  - L'affectation = en Python qui est le signe du test d'égalité en Scratch
  - Le typage des variables (plus de types en Python et moins implicites)
  - Les boucles ne sont pas strictement identiques dans les deux langages
  - Le mode d'écriture des programmes: graphique en Scratch, texte en Python

$\Leftarrow$ Il faut travailler sur ces sous ensembles en cycle 4 de collège et début de seconde

$\Leftarrow$ La notion de pseudo-code peut permettre de faire le passage de l'un à l'autre.

# Un exemple

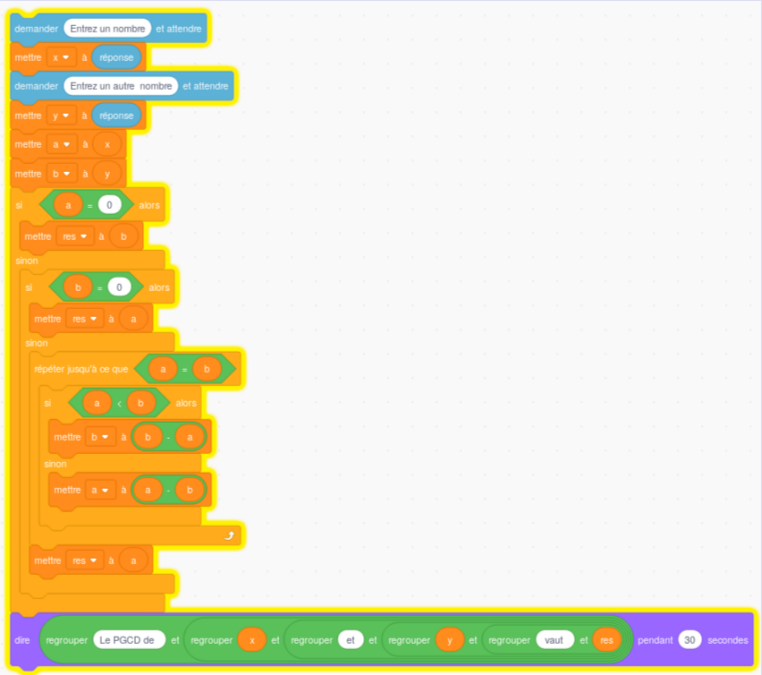
## Le PGCD par la méthode des soustractions

L'algorithme peut s'écrire sous la forme suivante:

```
paramètres:  x et y deux nombres entiers
résultat:  un nombre entier PGCD de x et y
début:
    a ← x
    b ← y
    si a vaut 0 alors
        res ← b
    sinon
        si b vaut 0 alors
            res ← a
        sinon
            tant que a et b sont différents faire
                si a<b alors
                    b ← b-a
                sinon
                    a ← a-b
            res ← a
        retourner res
```

# Le PGCD par la méthode des soustractions

## Scratch



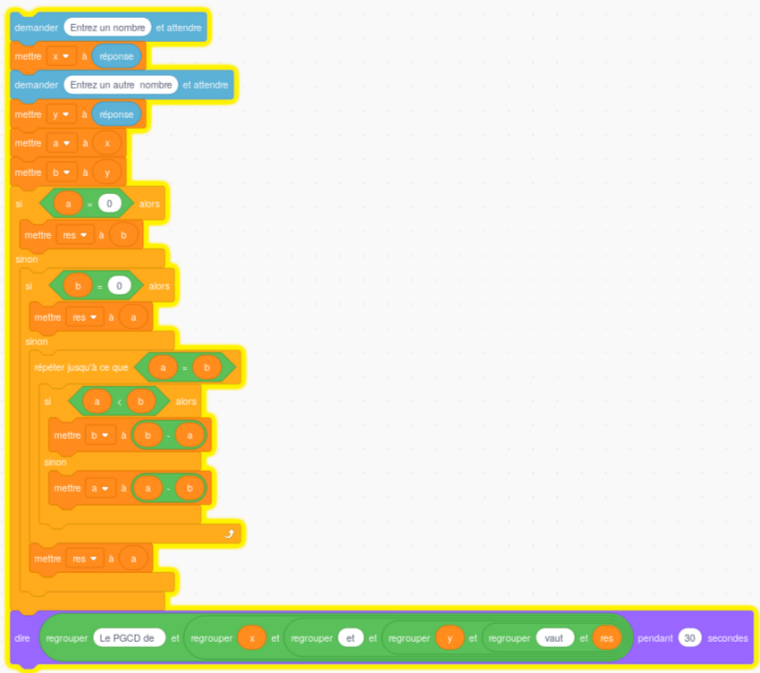

## Python (v1)

```
x=int(input("Entrez un nombre "))
y=int(input("Entrez un autre nombre "))
a=x
b=y
if a==0:
    res=b
elif b==0:
    res=a
else:
    while a!=b:
        if a<b:
            b-=a
        else:
            a-=b
    res=a
print("le PGCD de",x,"et",y,"est",res)
```

- Pour les lycéens on peut imaginer de partir
  - d'un énoncé
  - du Scratch
  - du pseudo-code

# Le PGCD par la méthode des soustractions

## Scratch



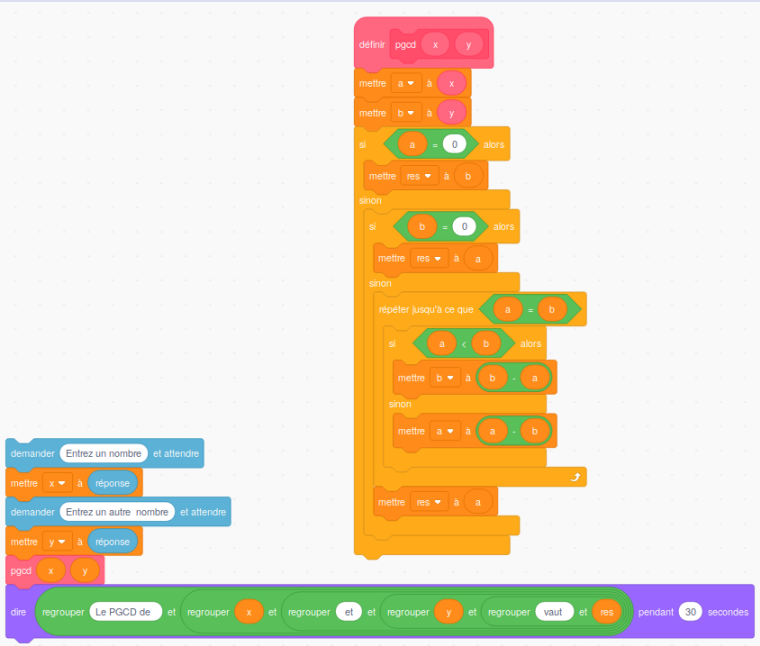

## Python (v2) avec fonction

```
def pgcd(a,b):
    if a==0:
        res=b
    elif b==0:
        res=a
    else:
        while a!=b:
            if a<b:
                b-=a
            else:
                a-=b
        res=a
    return res
x=int(input("Entrez un nombre "))
y=int(input("Entrez un autre nombre "))
print("le PGCD de",x,"et",y,"est",pgcd(x
    ,y))
```

- En Python on écrit des fonctions
- Pas de vrai équivalent en Scratch (les blocs ne sont pas des fonctions)

# Le PGCD par la méthode des soustractions

## Scratch




## Python (v2) avec fonction

```
def pgcd(a,b):
    if a==0:
        res=b
    elif b==0:
        res=a
    else:
        while a!=b:
            if a<b:
                b-=a
            else:
                a-=b
        res=a
    return res
x=int(input("Entrez un nombre "))
y=int(input("Entrez un autre nombre "))
print("le PGCD de",x,"et",y,"est",pgcd(x
    ,y))
```

- En Scratch: pas de variables locales
- En Scratch: pas de possibilité de retourner une valeur

# Plan

# Le dessin

## Scratch: le stylo


## Python: turtle

```
from turtle import *
clear()
pendown()
a=200
while a>0:
    forward(a)
    left(90)
    a-=5
turtle.done()
```

# Créer une liste

## La liste des diviseurs d'un nombre

### Scratch


### Python

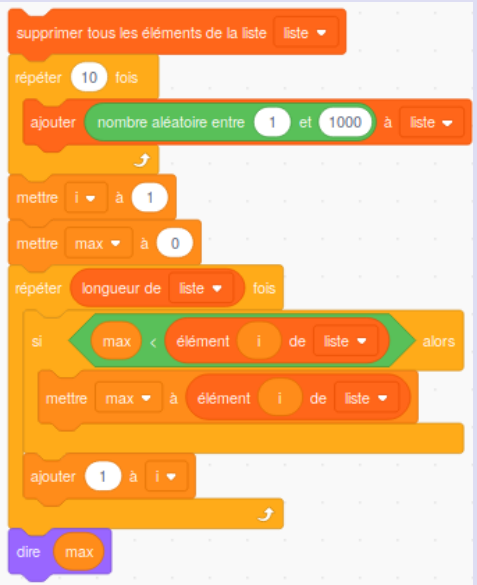
```
def liste_chiffres(nb):
    res=[nb%10]
    decomp=res//10
    while decomp!=0:
        res.append(decomp%10)
        decomp//=10
    return res
x=int(input("Entrez un nombre "))
print("la liste des chiffre de",x,"
    est",liste_chiffres(x))
```

- En Python [] désigne la liste vide

# Consulter une liste

## Recherche du maximum d'une liste

### Scratch




### Python: turtle

```
def maximum(liste):
    maxi=None
    for i in range(len(liste)):
        if maxi==None or liste[i]>
            maxi:
            maxi=liste[i]
    return maxi
liste=[12,4,5,46,55,43,2]
print("le max dans",liste,"est",
    maximum(liste))
```

- En Python les indices d'une liste vont de `0` à `len(liste)-1`
- En Scratch les indices d'une liste vont de `1` à `len(liste)`
- En Python, la notation `liste[i]` désigne l'élément numéro `i` de la liste

# Conclusion

- Un balayage des éléments communs aux deux langages
- Noyau permettant
  - de préparer les collégiens au programme de lycée
  - de fournir des points de repère aux lycéens pour débuter Python
- Il faut s'approprier ces éléments

# La suite

- Fiches de travail permettant
  - De travailler en Scratch ou en Python ou les 2 pour se familiariser aux langages
  - De travailler sur comment présenter ces notions aux collégiens/lycéens
  - Trois thèmes
    - Les structures de base
    - Le dessin
    - Les listes