UE "Informatique S3" : Programmation 2

Résponsable: Florian Bridoux (florian.bridoux@lis-lab.fr)

Objectifs :

- Approfondir la programmation en Python 3
- Interfaces graphiques et dessins avec TkInter
- Programmation d'algorithmes orientés mathématiques
- Voir des choses utiles pour la suite (travail sur ordi, enseignement,...)

UE "Informatique S3" : Programmation 2

CM,TD,TP disponibles sur:
https://pageperso.lis-lab.fr/ florian.bridoux/cours/prog3/

Ressources supplémentaires:
https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python

Volume horaire:

- CM: 12 heures
- TD: 6 heures
- TP: 12 heures

Notations:

- session 1 : Max(0.5*note examen + 0.5* note projet, note examen)
- Session 2 : note examen rattrapage

Présentation du langage Python

Histoire:

- Créé par Guido van Rossum (informaticien hollandais) depuis 1989. Objectif: être facile à apprendre et puissant.
- Langage de plus en plus utilisé : scientifiques, applis web, administration système, prototypage... enseigné partout (lycées, prépas, écoles, licences, masters...) idéal pour débutants et dev confirmés

Version de python:

- v 2.7 encore utilisé dans de nombreux programmes sera obsolète en 2020 : https://pythonclock.org/
- v >= 3.4 conseillée versions les plus récentes : améliorations sur des points techniques

Pour ce cours :

- utilisez le terminal + gedit
- évitez les environnements de développement intégrés (EDI) sauf si maîtrisés

Utilisation python

Deux modes:

1. interactif (calculette): taper *python3* dans un terminal.
2. script (programme): taper *python3 exo.py* pour exécuter le script exo.py.

Types primitifs

1. *int*: nombre entier
2. *float*: nombre à virgule
3. *bool* (True,False) : valeur logique
4. *string*: chaîne de caractère
5. *tuple*: *n*-uplet
6. *None*: similaire à NULL en c

Types primitifs: type *int* (entier)

```
>>> 12 + 13 # addition
25
>>> 14 - 12 # soustraction
2
>>> 3 * 5 # multiplication
15
>>> 7 // 3 # quotient division entière
2
>>> 7 % 3 # reste division entière
1
>>> 3**4 # 3 puissance 4
81
```

Types primitifs: type *int* (entier)

```
>>> 2+3*100 # multiplication prioritaire!
302
>>> (2+3)*100
500
>>> 2*3**2 # puissance prioritaire!
18
>>> (2*3)**2
36
>>> 5//0 # division par 0 interdite
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
>>> 2 % 0 # division par 0 interdite
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
```

Types primitifs: type *int* (entier)

```
>>> 123**12
1199116384871690629707272l
>>> max(23,29,36,12)
36
>>> min(23,28)
23
>> abs(-2)
2
>>> type(27)
<class 'int'>
>>> type(-5)
<class 'int'>
>>> type(0)
<class 'int'>
>>> type(25*62)
<class 'int'>
```

Types primitifs: type *float* (nombre à virgule)

```
>>> type(1.5)
<class 'float'>
>>> type(1.0)
<class 'float'>
>>> 1 / 3
0.3333333333333333
>>> 1.0 * 2
2.0
>>> 1 * 2.0
2.0
>>> 1 + 2.0
3.0
>>> 2.5 * 3.75
9.375
>>> 2.3**2.3
6.791630075247877
```

Types primitifs: type *float* (nombre à virgule)

```
>>> import math
>>> math.log(10,2)
3.3219280948873626
>>> math.ceil(6.5) # arrondie supérieur
7
>>> math.ceil(6.5) # arrondie inférieur
6
>>> math.sqrt(26)  # racine carrée
5.099019513592784 5
>>> help(math) # liste les fonctions du module math
```

Regardez la liste des fonctions de math.

Types primitifs: type Bool (vrai ou faux)

```
>>> type(True) #Vrai
<class 'bool'>
>>> type(False) #Faux
<class 'bool'>
>>> 1==1 # égalité
True
>>> 1!=1 # inégalité
False
>>> 2 < 2 # inférieur strict
False
>>> 2 <= 2 # inférieur ou égal
True
>>> 3 > 2 # supérieur
True
>>> 1.5 >= 2 # supérieur ou égal
False
```

Types primitifs: type Bool (vrai ou faux)

```
>>> 3 % 2 == 1
True
>>> False or False
False
>>> True or False
True
>>> True and False
False
>>> True and True
True
>>> 3 == 4 or 2 < 3
True
>>> not True
False
>>> not False
True
```

Types primitifs: type string (chaine de caractère)

```
>>> "cheval"
'cheval'
>>> 'cheval'
'cheval'
>>> type("cheval")
<class 'str'>
>>> '' #chaîne vide
''
>>> len("cheval")
6
```

Types primitifs: type string (chaine de caractère)

```
>>> "cheval"[0] # première lettre
'c'
>>> "cheval"[1] # deuxième lettre
'h'
>>> "cheval"[-1] # dernière lettre
'l'
>>> "cheval"[0:3] # première à troisième lettre
'che'
>>> "cheval"[2:-1] # troisième à avant dernière lettre
'eva'
>>> "cheval"[2:] # troisième à dernière lettre
'eval'
>>> "cheval"[:4] #sous-chaîne: première à troisième lettre
'che'
```

Types primitifs: type string (chaine de caractère)

```
>>> 'un '+'cheval' #concaténation
'un cheval'
>>> "des " + "cheval"[:-1] + "ux"
'des chevaux'
>>> "abc "*3 #répétition
'abc abc abc '
>>> "abc" + 12
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not int
>>> "abc" + str(12)
'abc12'
>>> int('123') + 12
135
```

Types primitifs: type tuple (n-uplet)

```
>>> type( (12,13,14) ) #tuple de taille 3
<class 'tuple'>
>>> len( (12,13,14) )
3
>>> type( (12,) ) #tuple de taille 1
<class 'tuple'>
>>> type( (), ) # tuple de taille 0
<class 'tuple'>
>>> (12,25) + (13,) + (45,297) # concaténation
(12, 25, 13, 45, 297)
>>> (12,25) * 2 # répétition
(12,25,12,25)
>>> (12,23,34)[-1] #dernier élément
34
```

Types primitifs: None

```
>>> type(None)
<class 'NoneType'>
```

Variables

```
>>> abeille
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>> abeille = 34
>>> abeille
34
>>> ma_variable_73 = "Paris" #convention de nommage
>>> être = "verbe" #pas recommandé
>>> 7_angelina = True
  File "<stdin>", line 1
    7_angelina = 32
     ^
SyntaxError: invalid token
```

Variables

```
>>> a = 10
>>> a
10
>>> type(a)
<class 'int'>
>>> a = "hola "
>>> a
'hola '
>>> type(a)
<class 'str'>
>>> b = a*2
>>> b
'hola hola '
```

Variables

```
>>> combien = 0
>>> combien
0
>>> combien = combien + 1
>>> combien
1
>>> combien = combien +1
>>> combien
2
>>> a = 10
>>> b = 10 * 2
>>> b
20
>>> a = 1
>>> b
20
```

Variables

Conclusion :

- On ne déclare pas les variables en python: on les initialise juste.
- Le type d'une variable est *dynamique* (*i.e.* il change avec le temps). C'est le type de sa valeur courante.
- On a vu les types primitifs: int, float, bool, string, tuple, None, mais il y a beaucoup d'autres types (que l'on verra plus tard): listes, ensembles, dictionnaires, voir des types personnalisés ...

Entrées-sorties

```
>>> a = input("Entrée utilisateur:")
Entrée utilisateur:123
>>> a
'123'
>>> type(a)
<class 'str'>
>>> a*3
'123123123'
>>> a = int(a)
>>> a
123
>>> type(a)
<class 'int'>
>>> a*3
369
```

Entrées-sorties

```
>>> print("bonjour")
bonjour
>>> print(129)
129
>>> a = "au revoir"
>>> print(a)
au revoir
>>> a = 5
>>> b = 7
>>> print("a=",a,"et b=",b,"donc a+b=",a+b)
a= 5 et b= 7 donc a+b= 12
>>> a=3
>>> print("a=",a,"et b=",b,"donc a+b=",a+b)
a= 3 et b= 7 donc a+b= 10
```

Entrées-sorties

Exercice:

1. Faites un programmes python qui demande 3 notes à l'utilisateur et en affiche la moyenne.

Entrées-sorties

Exercice:

1. Faites un programmes python qui demande 3 notes à l'utilisateur et en affiche la moyenne.

Correction:

```
note1 = int(input("note 1?"))
note2 = int(input("note 2?"))
note3 = int(input("note 3?"))

moyenne = (note1+note2+note3) / 3
print("La moyenne est de:",moyenne)
```

Bloc conditionnel

```
>>> if True:
...     print("bonjour")
...
bonjour
>>> if False:
...     print("bonjour")
... else:
...     print("pas bonjour")
pas bonjour
>>> if False:
...     print("bonjour")
... elif True:
...     print("bonjour quand même")
... else:
...     print("pas bonjour")
bonjour quand même
```

Bloc conditionnel

```
>>> a = 10
>>> if a == 1:
...     print("égale")
... elif a > 2:
...     print("plus grand")
... else:
...     print("plus petit")
...
plus grand
>>> a = 1
>>> if a == 1:
...     print("égale")
... elif a > 2:
...     print("plus grand")
... else:
...     print("plus petit")
...
égale
```

Bloc conditionnel

```
>>> a = 0.5
>>> if a == 1:
...     print("égale")
... elif a > 2:
...     print("plus grand")
... else:
...     print("plus petit")
...
plus petit
```

Bloc conditionnel

Exercice:

1. Faites un programmes python qui demande à l'utilisateur 2 notes et indique qu'elles sont égales si elles le sont ou la plus petite des deux notes si elles ne le sont pas.

Bloc conditionnel

Exercice:

1. Faites un programmes python qui demande à l'utilisateur 2 notes et indique qu'elles sont égales si elles le sont ou la plus petite des deux notes si elles ne le sont pas.

Correction:

```
note1 = int(input("note 1?"))
note2 = int(input("note 2?"))

if note1 == note2:
    print("Les deux notes sont égales.")
elif note1 < note2:
    print("La première note est la plus petite.")
else:
    print("La deuxième note est la plus petite.")
```

Boucle

```
>>> i = 0
>>> while i < 7:
...     print(i)
...     i = i + 1
...
0
1
2
3
4
5
6
```

Boucle

```
>>> for i in range(3,7):
...     print(i)
...
3
4
5
6
>>> for i in range(4): # = range(0,4)
...     print(i)
...
0
1
2
3
```

Boucle

Exercice:

1. Faites un programmes python qui demande à l'utilisateur combien de note il a reçu, lui demande ces n notes et en affiche la moyenne.

Boucle

Exercice:

1. Faites un programmes python qui demande à l'utilisateur combien de note il a reçu, lui demande ces n notes et en affiche la moyenne.

Correction (boucle for):

```
nombre_notes = int(input("Combien de notes?"))
somme = 0
for i in range(nombre_notes):
    note_i = int(input("note?"))
    somme = somme + note_i
print("La moyenne est de",somme/nombre_notes)
```

Boucle

Exercice:

1. Faites un programmes python qui demande à l'utilisateur combien de note il a reçu, lui demande ces n notes et en affiche la moyenne.

Correction (boucle while):

```
nombre_notes = int(input("Combien de notes?"))
somme = 0
i = 0
while i < nombre_notes:
    note_i = int(input("note?"))
    somme = somme + note_i
    i = i + 1
print("La moyenne est de",somme/nombre_notes)
```

Boucle

Exercice:

1. Faites un programmes python qui demande à l'utilisateur combien de note il a reçu, lui demande ces n notes et en affiche la moyenne.

Correction (boucle while):

```
nombre_notes = int(input("Combien de notes?"))
somme = 0
i = 0
while i < nombre_notes:
    note_i = int(input("note?"))
    somme = somme + note_i
    i = i + 1
print("La moyenne est de",somme/nombre_notes)
```

Fonction

```
>>> def ma_fonction(param1,param2):
...     print(param1,param2)
...     return param1+param2
...
>>> a = ma_fonction(12,40)
12
>>> a
52
>>> def ma_fonction_sans_return(param1,param2):
...     print(param1,param2)
...
>>> a = ma_fonction_sans_return(1,2)
1 2
>>> print(a)
None
```

Fonction

```
>>> def fact(n):
...     if n == 1:
...         return 1
...     else :
...         return fact(n-1)*n
...
>>> fact(7)
5040
```

Fonction

Exercice:

1. Faites une fonction $is\_prime(n)$ qui dit si un nombre entier $n > 1$ est premier (divisible seulement par 1 et par lui même). Afficher tous les nombre premiers entre 2 et 100.

Fonction

Exercice:

1. Faites une fonction *is_prime*(n) qui dit si un nombre entier $n > 1$ est premier (divisible seulement par 1 et par lui même). Afficher tous les nombre premiers entre 2 et 100.

Correction:

```
def is_prime(n):
    for i in range(2,n):
        if n % i == 0:
            return False
    return True

for n in range(2,101):
    if is_prime(n):
        print(n)
```