corrigé-TP1

September 3, 2020

1 Utiliser le mode interactif

1.1 Exercice 1

1. Écrire les expressions suivantes avec un parenthésage explicite permettant d'indiquer l'ordre de priorité des opérations :

```
>>> 2 - 3 - 4
-5
>>> 1 / 2 ** 3
0.125
>>> 1/ 2 / 4
0.125
>>> 1/ 2 * 4
2.0
>>> 2 * 3 ** 2
18
>>> 2 ** 3 ** 2
512
>>> 34 // 3 % 4
3
>>> 5 % 3 ** 4
5
```

avec parenthésage :

```
>>> (2 - 3) - 4
-5
>>> 1 / (2 ** 3)
0.125
>>> (1/ 2) / 4
0.125
>>> (1/ 2) * 4
2.0
>>> 2 * (3 ** 2)
18
>>> 2 ** (3 ** 2)
512
>>> (34 // 3) % 4
3
```

```
>>> 5 % (3 ** 4)
5
```

2. Écrire les expressions suivantes avec le moins de parenthèses possibles :

```
>>> 2 + (8 * (4 - 5))
>>> (3 - 5) - ((6 * 2) / (5 ** 2))
```

avec moins de parenthèses :

```
>>> 2 + 8 * (4 - 5)
>>> 3 - 5 - 6 * 2 / 5 ** 2
```

1.2 Exercice 2

Prédire la valeur affichée dans l'interprète [Python][Python] après les séquences d'instructions suivantes.

1. Séquence 1 :

```
>>> a = 5
>>> a = a + 1
>>> b = a
>>> b = b ** 2 - a
>>> print(b)
```

[1]:
```
a = 5
a = a + 1
b = a
b = b ** 2 - a
print(b)
```

```
30
```

2. Séquence 2 :

```
>>> a = 5
>>> b = 6
>>> a = a - b
>>> b = b + a
>>> a = b - a
>>> print(a, b)
```

[2]:
```
a = 5
b = 6
a = a - b
b = b + a
a = b - a
print(a, b)
```

```
6 5
```

3. Séquence 3 :

```
>>> from random import randint
>>> a = randint(1, 100)  #entier aléatoire entre 1 et 100
>>> b = randint(1, 100)  #entier aléatoire entre 1 et 100
>>> a = a - b
>>> b = b + a
>>> a = b - a
>>> print(a, b)
```

[3]:
```
from random import randint
a = randint(1, 100)  #entier aléatoire entre 1 et 100
b = randint(1, 100)  #entier aléatoire entre 1 et 100
a = a - b
b = b + a
a = b - a
print(a, b)
```

```
83 6
```

2 Premiers programmes et premières erreurs

2.1 Exercice 3

1. Dans son espace personnel, créer un répertoire `Chapitre1` avec un sous-répertoire `TP1`.
2. Le prix d'une matière première est de 873 euros la tonne au début de l'année. Ce prix subit des variations saisonnières : au premier trimestre il augmente de 347 euros, au second trimestre il augmente de 25 %, au troisième trimestre il subit une baisse de 50 % et enfin il diminue de 100 euros.

- Créer un nouveau programme avec l'éditeur d'Idle et l'enregistrer dans `Chapitre1/TP1` sous le nom `prix.py`.
- Saisir dans ce fichier le code ci-dessous en le complétant afin qu'il calcule les valeurs successives de la variable `prix`.

```
prix = 873        #prix au début de l'année
prix = ....       #prix à la fin du premier trimestre
prix = ....       #prix à la fin du second trimestre
prix = ....       #prix à la fin du troisième trimestre
prix = ....       #prix à la fin de l'année
print("Prix final :", prix)
```

[4]:
```
prix = 873        #prix au début de l'année
prix = prix + 347       #prix à la fin du premier trimestre
prix = prix * 1.25       #prix à la fin du second trimestre
prix = prix * 0.5       #prix à la fin du troisième trimestre
prix = prix - 100       #prix à la fin de l'année
print("Prix final :", prix)
```

```
Prix final : 662.5
```

2.2 Exercice 4

La température f en degrés Fahrenheit s'obtient à partir de la température c en degrés Celsius par la formule de conversion $f = 1,8 * c + 32$.

On veut écrire un programme qui réponde à la **spécification** suivante : *convertir une mesure de température de l'échelle Celsius vers l'échelle Fahrenheit.*

1. Dans l'éditeur d'Idle, créer un programme `temperature.py` et saisir le code ci-dessous :

```
f = input("Température en degrés Fahrenheit ? "
d = 1,8 * f + 32
print("La température en degrés Celsius est de ", d)
```

2. Exécuter le code, on doit obtenir un message d'erreur indiquant une erreur de Syntaxe. Un curseur indique dans le code la position où l'interpréteur [Python][Python] s'est interrompu dans la lecture du code. Si le curseur est en début de ligne, il faut souvent chercher l'erreur à la fin de la ligne précédente …

   Corriger l'erreur de syntaxe.

Corrigé : la première erreur est une erreur de syntaxe est l'oubli de la fermeture de la parenthèse de la fonction `input`.

[5]:
```
f = input("Température en degrés Fahrenheit ? "
d = 1,8 * f + 32
print("La température en degrés Celsius est de ", d)
```

```
  File "<ipython-input-5-fb4fd8a27b6e>", line 2
    d = 1,8 * f + 32
    ^
SyntaxError: invalid syntax
```

Corrigé : la seconde erreur est une erreur de type : on ne peut pas additionner la valeur de `f` qui est de stype `str` (car renvoyée par `input`) et 32 de type `int`.

[10]:
```
f = input("Température en degrés Fahrenheit ? ")
print(type(f))
d = 1,8 * f + 32
print("La température en degrés Celsius est de ", d)
```

```
Température en degrés Fahrenheit ? 45
<class 'str'>
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)
```

```
<ipython-input-10-494235267baf> in <module>
  1 f = input("Température en degrés Fahrenheit ? ")
  2 print(type(f))
----> 3 d = 1,8 * f + 32
  4 print("La température en degrés Celsius est de ", d)


TypeError: can only concatenate str (not "int") to str
```

5. Remplacer la première instruction par `float(input("Température en degrés Fahrenheit ? "))` puis exécuter.

   Quel est l'effet de la fonction `float` ? Afficher sa documentation dans l'interpréteur avec l'instruction `help(float)`.

Corrigé : la troisième erreur est plus subtile : Python retourne un `tuple` car on a utilisé la virgule au lieu du point comme séparateur décimal.

[13]:
```
f = float(input("Température en degrés Fahrenheit ? "))
print(type(f))
d = 1,8 * f + 32
print("La température en degrés Celsius est de ", d)
```

```
Température en degrés Fahrenheit ? 44
<class 'float'>
La température en degrés Celsius est de  (1, 384.0)
```

Corrigé : une version correcte du code :

[12]:
```
f = float(input("Température en degrés Fahrenheit ? "))
d = 1.8 * f + 32
print("La température en degrés Celsius est de ", d)
```

```
Température en degrés Fahrenheit ? 44
La température en degrés Celsius est de  111.2
```

3 Effets de bord et erreur

3.1 Exercice 5

On veut écrire un programme vérifiant la **spécification** suivant : *le programme doit permuter les valeurs de deux variables* `a` *et* `b` *de type entier saisies en entrée.*

Avec l'éditeur Idle, créer dans le répertoire `TP1` un nouveau programme `permutation.py` et recopier le code ci-dessous

```
# entrées
a = int(input('a ?'))
b = int(input('b ?'))
```

```
# traitement
a = b
b = a
# sorties
print("a = ", a, " et b = ", b)
```

1. Tester ce programme pour les entrées 605 et 506. La spécification du programme est-elle satisfaite ?

[15]:
```
# entrées
a = int(input('a ?'))
b = int(input('b ?'))
# traitement
a = b
b = a
# sorties
print("a = ", a, " et b = ", b)
```

```
a ?605
b ?506
a =  506  et b =  506
```

Le programme ne permute pas les valeurs des variables `a` et `b`, il affecte la valeur initiale de `a` à `a` puis à `b`.

2. Pour représenter l'exécution du programme, compléter le tableau d'état ci-dessous, qui affiche pour chaque ligne d'instruction, les valeurs des variables `a` et `b` et les éventuelles interactions avec l'utilisateur.

Ligne	Variable a	Variable b	Interactions
a = int (input ('a ?'))	734		**affichage :** 'a ?' **saisie** : 734
b = int (input ('b ?'))	734	437	**affichage :** 'b ?' **saisie :** 437
a = b	437	437	
b = a	437	437	
print ("a =" , a, " et b = " , b)			**affichage :** a = 437 et b = 437

3. Proposer une modification du programme qui permute les valeurs des variables `a` et `b` saisies en entrée.

   Démontrer que le programme est correct en complétant un tableau d'état de ce nouveau programme qui utilise des valeurs indéterminées `x` et `y` pour les variables `a` et `b` en entrée.

On peut utiliser une variable supplémentaire pour stocker la valeur initiale de `a`.

[18]:
```
# entrées
a = int(input('a ?'))
b = int(input('b ?'))
# traitement
c = a
a = b
```

```
b = c
# sorties
print("a = ", a, " et b = ", b)
```

```
a ?437
b ?734
a =  734  et b =  437
```

4. Proposer une modification du programme qui permute les valeurs des variables `a` et `b` saisies en entrée, sans utiliser de variable supplémentaire.

Permutation de variables, sans utiliser une variable supplémentaire pour stocker la valeur initiale de `a`.

[19]:
```
# entrées
a = int(input('a ?'))
b = int(input('b ?'))
# traitement
a = a + b
b = a - b
a = a - b
# sorties
print("a = ", a, " et b = ", b)
```

```
a ?437
b ?734
a =  734  et b =  437
```

Ligne	Variable a	Variable b	Interactions
a = int (input ('a ?'))	x		**affichage :** 'a ?' **saisie** : x
b = int (input ('b ?'))	x	y	**affichage :** 'b ?' **saisie :** y
a = a + b	x + y	y	
b = a - b	x + y	x	
a = a - b	y	x	
print ("a =" , a, " et b = " , b)			**affichage :** a = y et b = x

4 Utiliser une bibliothèque

[28]:
```
from turtle import *
```

[29]:
```
from ipyturtle import Turtle
```

[23]:
```
help(circle)
```

```
Help on function circle in module turtle:

circle(radius, extent=None, steps=None)
```

```
Draw a circle with given radius.

Arguments:
radius -- a number
extent (optional) -- a number
steps (optional) -- an integer

Draw a circle with given radius. The center is radius units left
of the turtle; extent - an angle - determines which part of the
circle is drawn. If extent is not given, draw the entire circle.
If extent is not a full circle, one endpoint of the arc is the
current pen position. Draw the arc in counterclockwise direction
if radius is positive, otherwise in clockwise direction. Finally
the direction of the turtle is changed by the amount of extent.

As the circle is approximated by an inscribed regular polygon,
steps determines the number of steps to use. If not given,
it will be calculated automatically. Maybe used to draw regular
polygons.

call: circle(radius)                  # full circle
--or: circle(radius, extent)          # arc
--or: circle(radius, extent, steps)
--or: circle(radius, steps=6)         # 6-sided polygon

Example:
>>> circle(50)
>>> circle(120, 180)  # semicircle
```

4.1 Exercice 6

1. Avec l'éditeur d'Idle, créer dans le répertoire `TP1` un nouveau programme `tortue-polygones.py` et importer toutes les fonctions du module `turtle` avec `from turtle import *`.
2. Écrire un programme qui trace un carré de côté 100 pixels en utilisant les instructions `forward` et `left`.

[39]:
```
from turtle import *
up()
goto(0,0)
down()
forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
```

```
forward(100)
left(90)
exitonclick()
```

2. Écrire un programme qui trace un triangle équilatéral de côté 100 pixels.

[43]:
```
from turtle import *
up()
goto(0,0)
down()
forward(100)
left(120)
forward(100)
left(120)
forward(100)
left(120)
exitonclick()
```

3. Écrire un programme qui trace un hexagone régulier de côté 100 pixels.

[37]:
```
from turtle import *
angle = 360 / 6
up()
goto(0,0)
down()
forward(100)
left(angle)
forward(100)
left(angle)
forward(100)
left(angle)
forward(100)
left(angle)
forward(100)
left(angle)
forward(100)
left(angle)
exitonclick()
```

4.2 Exercice 7

1. Avec l'éditeur d'Idle, créer dans le répertoire `TP1` un nouveau programme `tortue-clef.py` et importer toutes les fonctions du module `turtle` avec `from turtle import *`.
2. Exécuter puis dans le du mode interactif la documentation de l'instruction `circle` avec `help(circle)`.
3. Saisir dans le programme la séquence d'instructions suivante, exécuter puis observer.

```
from turtle import *
```

```
forward(60)
left(120)
forward(60)
right(90)
circle(60,150)
exitonclick()
```

1. Compléter le programme pour afficher une figure en forme de clef.

[3]:
```
from turtle import *
forward(60)
left(120)
forward(60)
right(90)
circle(60,300)
right(90)
forward(60)
left(120)
forward(60)
right(90)
exitonclick()
```

4.3 Exercice 8

1. Avec l'éditeur d'Idle, créer dans le répertoire `TP1` un nouveau programme `tortue-triangles.py`.
2. Le programme ci-dessous permet de tracer un triangle équilatéral noir de côté 100 pixels. Compléter le code pour tracer la figure de droite avec la pyramide de triangles.

[48]:
```
begin_fill()
fillcolor("black")
forward(100)
left(120)
forward(100)
left(120)
forward(100)
left(120)
end_fill()
fillcolor("white")
forward(50)
begin_fill()
left(60)
forward(50)
left(120)
forward(50)
left(120)
forward(50)
left(120)
```

```
end_fill()
exitonclick()
```

4.4 Exercice 8

1. Avec l'éditeur d'Idle, créer dans le répertoire `TP1` un nouveau programme `tortue-drapeau.py`.
2. Écrire un programme qui dessine le drapeau français.

[14]:
```
begin_fill()
up()
goto(-100,0)
down()
fillcolor("blue")
forward(100)
left(90)
forward(200)
left(90)
forward(100)
left(90)
forward(200)
left(90)
end_fill()
up()
down()
forward(100)
begin_fill()
fillcolor("white")
forward(100)
left(90)
forward(200)
left(90)
forward(100)
left(90)
forward(200)
left(90)
end_fill()
up()
down()
forward(100)
begin_fill()
fillcolor("red")
forward(100)
left(90)
forward(200)
left(90)
forward(100)
```

```
left(90)
forward(200)
left(90)
end_fill()
exitonclick()
```