Structures de base [bs]
Exercices de cours

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 12 mai 2018

Table des matières

Python - Exercices de cours (Solution)

Mots-Clés Structures de base ■
Difficulté ● ○ ○

1 Appréhender le cours

1.1 Simplification de programme / pgsimplific

Soit le programme suivant :

```
def PGSimplific1a():
    a = int(input())
    b = a
    c = b + b * b
    x = int(input())
    e = 4 * x
    s = e + b
    s = s * s + c
    f = s
    print("Le resultat est ", f, sep="")

PGSimplific1a()
```

Déclarez les variables de type entier.

Quel est le résultat avec 3 et -2 ?
Justifiez.

Solution simple

Le résultat est 37. En effet :

- On saisit 3 dans `a`.
- `b` vaut `a` qui vaut 3.
- `c` vaut `3+3*3=12`.
- On saisit -2 dans `x`.
- `e` vaut `4*(-2)=-8`.
- `s` vaut `-8+3=-5` puis `(-5)*(-5)+12=25+12=37`.
- Finalement `f` vaut `s` qui vaut 37.

Les variables `b` et `f` étant inutiles,
Écrivez une première version simplifiée.

Validez votre script avec la solution.

Solution Python

```
def PGSimplific1b():
    a = int(input())
    c = a + a * a
    x = int(input())
    e = 4 * x
    s = e + a
    s = s * s + c
    print("Le resultat est ", s, sep="")

PGSimplific1b()
```

Quelles sont les variables incompressibles ?

Solution simple

Ce sont les données et les résultats :

- Données : `a` et `x`
- Résultats : `s`

En ne déclarant que les données et les résultats,
Écrivez une deuxième version simplifiée.

Validez votre script avec la solution.

Solution Python

```
def PGSimplific1c():
    a = int(input())
    x = int(input())
    s = 4 * x + a
    s = s * s + a + a * a
    print("Le resultat est ", s, sep="")

PGSimplific1c()
```

Vérifiez en recalculant le résultat.

Solution simple

Voici le résultat d'exécution :

```
3
-2
Le resultat est 37
```

1.2 Réutilisation d'une variable / pgaffect3

Déclarez deux entiers `x` et `y`.

Affectez 3 à `x`.

Affectez `x` à `y`.

Affectez `3 * x + 1` à `x`.

Affectez `2 * x - 5 * y` à `x`.

Affichez (où `[x]` désigne le contenu de `x`) :

```
x vaut [x] et y vaut [y]
```

Donnez la valeur des variables à l'issue de votre script.

Solution simple

On a :

- Initialement `x` vaut 3
- Puis `y` vaut `x` qui vaut 3
- Puis `x` vaut `3*3+1=10`
- Puis `x` vaut `2*10-5*3 = 20-15 = 5`
- Finalement :

```
x vaut 5 et y vaut 3
```

Testez.

Validez votre script avec la solution.

Solution Python @[pgaffect3.py]

```
def PGAffect3():
    x = 3
    y = x
    x = 3 * x + 1
    x = 2 * x - 5 * y
    print("x vaut ", x, " et y vaut ", y, sep="")

PGAffect3()
```

1.3 Exponentiation 16-ème / pgexponent1

Écrivez un script qui saisit un entier X dans `x`.
Affichez l'invite :

```
Valeur de X?
```

Calculez **le plus rapidement** X^{16} dans `x`.

Aide simple
Calculez X^2 puis X^4 puis X^8 puis X^{16}.

Affichez (où `[x]` désigne le contenu de `x`) :

```
La valeur de X^16 est [x]
```

Testez. Exemple d'exécution :

```
Valeur de X? 3
La valeur de X^16 est 43046721
```

Validez votre script avec la solution.

Solution Python @[pgexponent1.py]

```python
def PGExponent1():
    x = int(input("Valeur de X? "))
    x = x * x
    x = x * x
    x = x * x
    x = x * x
    print("La valeur de X^16 est ", x, sep="")

PGExponent1()
```

2 Appliquer le cours

2.1 Quelques calculs / pgcalcul1

Écrivez un script qui saisit deux nombres dans les variables `n1` et `n2` de type entier.

Calculez la somme, la différence, le produit et le quotient de `n1` et `n2` et affecte les résultats respectivement aux variables de type réel `a`, `b`, `c` et `d`.

Affichez les valeurs de `a`, `b`, `c` et `d`.

Compilez et exécutez le programme.
Testez avec 14 et 3 (par exemple).

Modifiez le programme en déclarant `n1` et `n2` avec le type réel.
Recompilez et exécutez.
Que constatez-vous ?

Même question avec `n1` de type entier et `n2` de type réel.
Expliquez.

Solution Python
Quel que soit les types des opérandes, l'opérateur `/` désigne la division réelle. Pour obtenir la division sur les entiers, il faut utiliser l'opérateur `//`.

Validez votre script avec la solution.

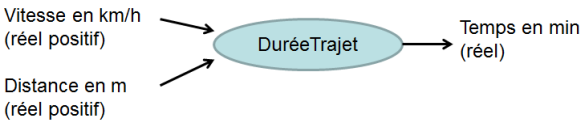
Solution Python @[pgcalcul1.py]

```
def PGCalcul1():
    print("Deux nombres? ", sep="", end="")
    n1 = int(input())
    n2 = int(input())
    a = n1 + n2
    b = n1 - n2
    c = n1 * n2
    d = n1 / n2
    print("La somme est ", a, sep="")
    print("La difference est ", b, sep="")
    print("Le produit est ", c, sep="")
    print("La division est ", d, sep="")

PGCalcul1()
```

2.2 Durée de trajet / pgtrajet1

Objectif
Cet exercice calcule la durée en minutes du trajet d'un véhicule :



Écrivez un script qui saisit la vitesse moyenne en **km/h** d'un véhicule dans `vmoy` (réel) et la distance parcourue en **m** par ce véhicule dans `dist` (réel). Affichez les invites :

```
Vitesse moyenne (en km/h)?
Distance en m?
```

La formule qui lit les trois éléments est :

$$vitesse = \frac{distance}{temps} \quad \text{d'où} \quad temps = \frac{distance}{vitesse}$$

pour autant que les unités soient compatibles. Dans notre cas, on peut d'abord convertir la distance en km d'où le temps sera exprimée en heure que l'on divisera par 60 pour l'exprimer en minutes. Par conséquent :

Calculez la distance en km dans `km` (réel) puis affichez :

```
==> Distance en km : ...
==> Duree en minutes : ...
```

Autre solution : Calculez la vitesse en m/s dans `vt` (réel) et la durée en secondes du trajet de ce véhicule dans `nsecs` (entier).

Affichez finalement :

```
==> Vitesse en m/s : ...
==> Duree en secondes : ...
==> Duree en minutes : ...
```

Testez. Exemple d'exécution :

```
Vitesse moyenne (en km/h)? 50
Distance en m? 5000
==> Distance en km : 5
==> Duree en minutes : 6
==> Vitesse en m/s : 13.8888888889
==> Duree en secondes : 360
==> Duree en minutes : 6
```

Validez votre script avec la solution.

Solution Python @[pgtrajet1.py]

```
def PGTrajet1():
    vmoy = float(input("Vitesse moyenne (en km/h)? "))
    dist = float(input("Distance en m? "))
    km = dist / 1000
    print("==> Distance en km : ", km, sep="")
    print("==> Duree en minutes : ", (km / vmoy * 60), sep="")
    vt = (vmoy * 1000.0) / 3600.0
    nsecs = int(dist / vt)
    print("==> Vitesse en m/s : ", vt, sep="")
    print("==> Duree en secondes : ", nsecs, sep="")
    print("==> Duree en minutes : ", (nsecs // 60), sep="")

PGTrajet1()
```

2.3 Un peu de trigo / pgtrigo1

Écrivez un script qui saisit un entier dans `n`.
Affichez l'invite :

```
Valeur de n?
```

Calculez et affichez les valeurs de $\cos(n * \pi/2)$ et de $\sin(n * \pi/2)$.

Outil Python
La constante `pi` est définie dans la bibliothèque `math`.

Testez avec plusieurs valeurs de `n`.
Concluez.

Solution simple
On remarque que les fonctions mathématiques sont des approximations.
Exemple d'exécution :

```
Valeur de n? 3
==> -1.8369701987210297e-16
==> -1.0
```

Validez votre script avec la solution.

Solution Python @[pgtrigo1.py]

```
import math

def PGTrigo1():
    n = int(input("Valeur de n? "))
    alpha = (n * math.pi) / 2
    print("==> ", math.cos(alpha), sep="")
    print("==> ", math.sin(alpha), sep="")

PGTrigo1()
```

2.4 Date de Pâques / pgpaques1

Objectif

Depuis le concile de Nicée (325 ap.J.-C.) et la mise en place du calendrier grégorien (en 1591), la date de Pâques est le premier dimanche après la première pleine lune qui suit l'équinoxe de printemps (du 21 mars). Il existe plusieurs formules permettant de calculer la date de Pâques pour une année donnée dans le calendrier grégorien.

Voici les calculs permettant de connaître cette date [$x \bmod p$ dénote le reste de x dans la division euclidienne par p] :

$$\begin{aligned} a &= annee \bmod 19 \\ b &= annee \bmod 4 \\ c &= annee \bmod 7 \\ d &= (19a + 24) \bmod 30 \\ e &= (2b + 4c + 6d + 5) \bmod 7 \\ n &= 22 + d + e \end{aligned}$$

où n est le numéro du jour du mois de mars (ou avril si n est supérieur à 31) correspondant au dimanche de Pâques.

Écrivez un script qui demande une année (qui est supposée comprise entre 1591 et 2099) puis calcule et affiche le « rang » de la date de Pâques correspondant.

Vérifiez qu'en 2002, Pâques est tombé le 31 mars, et qu'en 1996, le 7 avril.

Solution simple

Pour $annee = 2002$, on a :

$$\begin{aligned} a &= 2002 \bmod 19 = 7 \\ b &= 2002 \bmod 4 = 2 \\ c &= 2002 \bmod 7 = 0 \\ d &= (19 \times 7 + 24) \bmod 30 = 7 \\ e &= (2 \times 2 + 4 \times 0 + 6 \times 7 + 5) \bmod 7 = 2 \\ n &= 22 + 7 + 2 = 31 \end{aligned}$$

Donc Pâques est en mars et le quantième est le 31. Pour $annee = 1996$, les calculs indiquent que $n = 38$, ce qui signifie que le dimanche de Pâques est le 38è jour à partir du 1er mars, i.e. le 7 avril.

Validez votre script avec la solution.

Solution Python @[pgpaques1.py]

```
def PGPaques1():
    annee = int(input("Votre annee dans [1592..2099]? "))
    a = annee % 19
    b = annee % 4
    c = annee % 7
    d = (19 * a + 24) % 30
    e = (2 * b + 4 * c + 6 * d + 5) % 7
    n = 22 + d + e
    print("Numero de jour = ", n, sep="")

PGPaques1()
```

3 Approfondir le cours

3.1 Moyennes de deux entiers / pgmoyenne1

Écrivez un script qui saisit deux entiers dans `n1` et `n2`.
Affichez l'invite :

```
Deux entiers?
```

Calculez et affichez (où `[x]` désigne le contenu de `x`) :

```
Pour les entiers [n1] et [n2]:
  Moyenne decimale = ...
  Moyenne entiere  = ...
```

Testez. Exemple d'exécution :

```
Deux entiers? 3 8
Pour les entiers 3 et 8:
  Moyenne decimale : 5.5
  Moyenne entiere  : 5
```

Validez votre script avec la solution.

Solution Python @[pgmoyenne1.py]

```python
def PGMoyenne1():
    print("Deux entiers? ", sep="", end="")
    n1 = int(input())
    n2 = int(input())
    print("Pour les entiers ", n1, " et ", n2, ":", sep="")
    print("  Moyenne decimale = ", ((n1 + n2) / 2.0), sep="")
    print("  Moyenne entiere  = ", ((n1 + n2) // 2), sep="")

PGMoyenne1()
```

3.2 Codage d'une date / pgdate1

Objectif
On choisit de coder une date par un entier composé de la façon suivante :

$$a \times 10000 + m \times 100 + j$$

où a est l'année ($a \geq 1$), m le mois ($1 \leq m \leq 12$) et j le jour ($1 \leq j \leq 31$).
Exemple : La date « 14 juillet 1789 » sera codée par l'entier 17890714.
Cet exercice décode une telle date (dont on supposera qu'elle est valide).

Écrivez un script qui saisit une date codée dans un entier `dt`.
Affichez l'invite :

```
Votre date codee?
```

Calculez, les trois variables étant de type entier :

- Le numéro du jour dans `j`.
- Le numéro du mois dans `m`.
- Et l'année dans `a`.

Affichez (où `[x]` désigne le contenu de `x`) :

```
jour=[j] mois=[m] annee=[a]
```

Testez. Exemple d'exécution :

```
Votre date codee? 20161206
jour=6 mois=12 annee=2016
```

Validez votre script avec la solution.

Solution Python @[pgdate1.py]

```python
def PGDate1():
    dt = int(input("Votre date codee? "))
    j = dt % 100
    m = (dt // 100) % 100
    a = dt // 10000
    print("jour=", j, " mois=", m, " annee=", a, sep="")

PGDate1()
```