{% set num = 9 %} {% set titre = “Exercices Bilan Python”%} {% set theme = “python” %} {% set niveau = “premiere”%}

{{ titre_chapitre(num,titre,theme,niveau)}}

python_logo1.png{:.center width=100px}

{{ initexo(0) }}

!!! exo “{{ exercice()}} freinage ok”

```
*Remarque 1 :* Vous pouvez (et c'est même à cela que ça sert) réutiliser une fonction dans u

Une voiture roule sur route sèche lorsqu'un obstacle apparait sur la route devant le véhicul

Entre le moment où l'obstacle est apparu et le moment où la voiture s'arrête une certaine di
- `v` est donnée en kilomètres par heure,
- la valeur renvoyée correspond à la distance totale d'arrêt $d_{arret}$ en mètres.


```python
def distance_totale_arret(v):
    assert v >= 0
    d = 0.3*v + 0.005*v**2
    return d
```

Par exemple l'appel de fonction ci-dessous nous indique qu'à une vitesse de 120 km par heure


```
distance_totale_arret(120)
```
108.0

=== "Question 1 "
    Compléter la fonction `freinage_ok` qui prend en paramètres deux nombres :

    - une vitesse `v` correspondant à la vitesse du véhicule sur route sèche au moment où ap
    - un entier `distance_obstacle` désignant la distance à laquelle est apparu l'obstacle.

    Cette fonction doit renvoyer `True` si le véhicule a le temps de s'arrêter avant l'obsta

=== "Solution"
    ```python
    def freinage_ok(v, distance_obstacle):
```
```

```
        d = 0.3*v + 0.005*v**2
        if d < distance_obstacle:
            return True
        else:
            return False

    freinage_ok(135,140)
    ```

    True

=== "Question 2 : Jeu de test"
    Vérifier que votre fonction satisfait le jeu de tests suivant.

    ```python
    assert freinage_ok(20, 250)
    assert not freinage_ok(190, 40)
    assert not freinage_ok(20, 5)
    assert freinage_ok(30, 48)
    assert freinage_ok(40, 30)
    assert not freinage_ok(60, 30)
    assert freinage_ok(80, 60)
    ```

=== "Question 3 :"
    Un véhicule roule à 70 kilomètres par heure sur une route sèche. En utilisant votre fonc


    ```python
    freinage_ok(70,46)
    ```
    True
```

!!! exo “{{ exercice()}} réussir l’examen”

```
Dans tout ce qui suit les notes sont sur 20 points.

Voici un exemple de calcul de moyenne avec des coefficients :

- DS : 13 coefficient 2
- Evaluation : 18 coefficient 1
- TP : 16 coefficient 0.5
La moyenne est obtenue en faisant `(2*13 + 1*18 + 0.5*16)/(2+1+0.5)`

Un examen comporte 6 épreuves :
```
```

- Français coefficient 4,
- Anglais coefficient 3,
- Philosophie coefficient 1,
- Mathématiques coefficient 4,
- Physique coefficient 2,
- Informatique coefficient 6.

Un·e étudiant·e peut être admis·e de deux façons :

- si sa moyenne littéraire est supérieure ou égale à 11 et sa moyenne scientifique est supér
- ou bien si sa moyenne scientifique est supérieure ou égale à 16.

=== "Question 1"
    Programmer la fonction `reussir_examen` qui prend les six notes des épreuves comme param

=== "Solution"
    ```python
    def reussir_examen(fra, ang, phi, mat, phy, inf):
        moyenne_litt = (4*fra + 3*ang + 1*phi)/(4+3+1)
        moyenne_sci = (4*mat + 2*phy + 6*inf)/(4+2+6)
            if moyenne_litt>=11 and moyenne_sci>=11:
                return True
            elif moyenne_sci>=16:
                return True
            else:
                return False

    reussir_examen(17,12,9,19,2,11)
    ```
    True

=== "Question 2 : Jeu de test"
    Tester votre fonction grâce au jeu de tests ci-dessous :

    ```python
    assert reussir_examen(10, 13, 11, 15, 6, 12)
    assert not reussir_examen(8, 13, 11, 15, 6, 12)
    assert not reussir_examen(10, 13, 11, 11, 6, 12)
    assert reussir_examen(6, 3, 11, 15, 16, 20)
    ```

=== "Question 3 :"
    En utilisant votre fonction déterminer si un·e étudiant·e ayant eu les notes suivantes (

=== "Solution"

```python
reussir_examen(8,11,14,13,9,11)
False
```

!!! exo “{{ exercice()}} ballon ficelle” Dans une fête foraine un jeu de hasard consiste pour le joueur à réussir à crever un ballon parmi de nombreux ballons qui sont accrochés au plafond du stand grâce à des ficelles mesurant entre 30 centimètres et 99 centimètres.

Une fois crevé, le forain récupère la ficelle du ballon et l'enroule autour d'une bobine de

En fonction de la longueur, il arrive à faire un certain nombre de tours avec la ficelle. Pa

- Le joueur gagne 20 points si le nombre de tours de ficelle est égal au nombre de centimètr
- sinon il gagne 10 points si le nombre de centimètres de ficelle qui dépassent est supérieu
- sinon il gagne 5 points si le nombre de tours de ficelle est pair,
- sinon il ne gagne aucun point.

En faisant plusieurs lancers, le joueur cumule des points avec lesquels il peut s'offrir un

=== "Question 1"
    Compléter la fonction `score` ci-dessous qui prend en argument un nombre entier `l` corr

=== "Solution 1 "
    ```python
    def score(lg):
        nb_tours = lg // 10
        longueur_qui_depasse = lg % 10
        if nb_tours == longueur_qui_depasse:
            nbpoints= 20
        elif longueur_qui_depasse>=7:
            nbpoints= 10
        elif nb_tours % 2 == 0:
            nbpoints= 5
        else:
            nbpoints=0
        return nbpoints
    ```
=== "Solution 2"
    ```python
    def score(lg):
        nb_tours = lg // 10
        longueur_qui_depasse = lg % 10
        nb_points = 0
        if nb_tours == longueur_qui_depasse:
            return 20
    ```

```
        elif longueur_qui_depasse>=7:
            return 10
        elif nb_tours % 2 == 0:
            return 5
        else:
            return nb_points
    ```

=== "Question 2 : Jeu de test"
    Vérifier que votre fonction satisfait le jeu de tests suivant :

    ```python
    assert score(55) == 20
    assert score(33) == 20
    assert score(57) == 10
    assert score(49) == 10
    assert score(88) == 20
    assert score(43) == 5
    assert score(85) == 5
    assert score(56) == 0
    assert score(34) == 0
    ```

=== "Question 3"
    En utilisant votre fonction, déterminer le nombre de points lorsque la ficelle mesure 64

=== "Solution"
    ```python
    score(64)
    5
    ```
```

!!! exo “{{ exercice()}} annee bissextile”

```
=== "Question 1"
    D'après [Wikipédia](https://fr.wikipedia.org/wiki/Ann%C3%A9e_bissextile) :

    *«Depuis l'ajustement du calendrier grégorien, l'année n’est bissextile (comportant 366

    - *si l'année est divisible par 4 et non divisible par 100 ;*
    - *si l'année est divisible par 400 (« divisible » signifie que la division donne un nom
    - *Sinon, l'année n'est pas bissextile : elle a la durée habituelle de 365 jours (elle e

    *Ainsi, 2021 n'est pas bissextile. L'an 2008 était bissextil suivant la première règle (

    Programmer la fonction `annee_bissextile` qui prend en paramètre un nombre entier corres
```
```

```
=== "Solution"
    ```python
    def annee_bissextile(a):
        if a % 4 == 0 and a % 100 != 0:
            return True
        elif a % 400 ==0:
            return True
        else:
            return False
    ```

=== "Question 2 : Jeu de test"
    Tester votre fonction grâce au jeu de tests ci-dessous :

    ```python
    assert annee_bissextile(2008)
    assert not annee_bissextile(1900)
    assert not annee_bissextile(2021)
    assert annee_bissextile(2028)
    assert annee_bissextile(2400)
    assert not annee_bissextile(2100)
    ```

=== "Question 3"
    En utilisant votre fonction déterminer si 1792 est une année bissextile.

=== "Solution"
    ```python
    annee_bissextile(1792)
    True
    ```

=== "Question 4"
    rogrammer ci-dessous la fonction `nb_jours` qui prend en argument une année `a >= 1583`

    Cette fonction `nb_jours` réutilisera la fonction `annee_bissextile`.

=== "Solution"
    ```python
    def nb_jours(a):
        assert a >= 1583
        nb_jours = 0
        for i in range(a):
            if annee_bissextile(a) == True:
                nb_jours = 366
            else:
```
```

```
                nb_jours = 365
        return nb_jours
    ```

=== "Question 5"
    En utilisant votre fonction déterminer combien de jours comportait l'année 1594 et combi

=== "Solution"
    ```python
    print(nb_jours(1594))
    print(nb_jours(1748))
    365
    366
    ```

!!! exo “{{ exercice()}} nombre de bits” On sait que sur `k` bits on peut coder `2**k` mots différents. Par exemple sur `k = 10` bits on peut coder `2**10 = 1024` mots différents.

Un exemple classique est le codage ASCII sur `k = 7` bits qui permet de représenter `n = 2**

Mais si l'on souhaite coder `n` éléments différents, peut-on déterminer le nombre `k` de bit
Par exemple si l'on souhaite coder `n = 3 789 147` caractères, combien de bits sont nécessai

=== "Question 1"
    Compléter la fonction `nombre_de_bits` ci-dessous qui prend en paramètre un nombre `n >=

    Pour cela on teste différentes valeurs de `2**k` (en augmentant `k` de un en un) jusqu'à

=== "Solution"
    ```python
    def nombre_de_bits(n):
        k = 0
        while 2**k < n:
            k += 1
        return k

    nombre_de_bits(1025)
    11
    ```

=== "Question 2 : Jeu de test"
    Tester votre fonction en utilisant le jeu de tests ci-dessous.

    ```python
    from math import log, ceil, floor
    ```

```
    for n in  [2, 3, 4, 5, 6, 7, 8, 15, 16, 17, 1024, 1025, 2**57 - 1, 2**57]:
        assert nombre_de_bits(n) == (n-1).bit_length()
    ```
```

!!! exo “{{ exercice()}} Millionnaire”

```
Depuis ses huit ans, Super-avare économise des klipoks (monnaie locale).

Le jour de ses huit ans il avait `8**3 = 512` klipoks dans sa tirelire.
Le jour de ses neuf ans il avait `512 + 9**3 = 1241` klipoks dans sa tirelire.
Le jour de ses dix ans il avait `1241 + 10**3 = 2241` klipoks dans sa tirelire.
Le jour de ses onze ans il avait `2241 + 11**3 = 3572` klipoks dans sa tirelire.

=== "Question 1"
    Compléter la fonction `calculer_tirelire` qui prend en paramètre un entier `n` supérieur

    Remarque : on est ici en présence d'un algorithme d'accumulation (l'accumulateur est `

=== "Solution"
    ```python
    def calculer_tirelire(n):
        somme = 0
        for age in range(8, n+1):
            somme += age**3
        return somme

    calculer_tirelire(45)
    1070441
    ```

=== "Question 2: Jeu de test"
    Tester votre fonction en utilisant le jeu de tests ci-dessous.

    ```python
    assert calculer_tirelire(8)  == 512
    assert calculer_tirelire(9)  == 1241
    assert calculer_tirelire(10)  == 2241
    assert calculer_tirelire(11)  == 3572
    assert calculer_tirelire(18) == 28457
    assert calculer_tirelire(75) == 8121716
    ```

=== "Question 3"
    Compléter la fonction `age_millionnaire` afin qu'elle renvoie l'âge à partir duquel Supe

=== "Solution"
    ```python
```

```
    def age_millionnaire():
        tirelire = 0
        age = 0
        while tirelire < 1000000 :
            tirelire += age**3
            age += 1
        return age-1

    age_millionnaire()
    45
    ```
```

!!! exo "{{ exercice()}} Kangourou" Un kangourou se déplace en ligne droite en faisant des bonds dont la longueur est aléatoire. On modélise la longueur d'un bond en utilisant la fonction `randint(2, 13)` du module `random` qui renvoie un nombre entier aléatoire compris entre 2 et 13.

```
=== "Question 1"
    Compléter la fonction `distance_totale(n)` ci-dessous qui prend en paramètres un nombre

    Remarque : on est ici en présence d'un algorithme d'accumulation (l'accumulateur est `
=== "Solution"
    ```python
    from random import randint

    def distance_totale(n):
        total = 0
        for _ in range(n):
            longueur_saut = randint(2, 13)
            total += longueur_saut
        return total

    ```

=== "Question 2"
    Effectuer plusieurs appels à cette fonction pour voir quelle distance parcourt approxima

=== "Solution"
    ```python
    moyenne = 0
    for i in range(10000):
        distance = distance_totale(10)
        moyenne += distance
    moyenne = moyenne / 10000
    print(moyenne)
    75.0162
    ```
```
```

````
=== "Question 3"

    On souhaite désormais savoir combien de sauts aléatoires va faire le kangourou pour parc

=== "Solution"
    ```python
    from random import randint

    def nb_sauts_100_metres():
        total = 0
        nb_sauts = 0
        while total < 100 :
            longueur_saut = randint(2, 13)
            total += longueur_saut
            nb_sauts += 1
        return nb_sauts

    nb_sauts_100_metres()

    12
    ```
=== "Question 4"
    Effectuer des appels de fonctions pour vérifier si les résultats fournis sont vraisembla

=== "Solution"
    ```python
    for _ in range(50):
        print(nb_sauts_100_metres())
    ```
````