Exercices fonctions correction

Exercices fonctions correction

Christophe Viroulaud

Première - NSI

Lang 06

xercice 1

ercice 2

xercice .

ercice 4

xercice 5

.....

. .

1. Exercice 1 Exercice 2

2. Exercice 2

ce 3

. .

5 Evercice 5

- F

8 Exercice

Exer

xercice 7 xercice 8

kercice 9

rcice 10

Exercices fonctions

```
def est_pair(x: int) -> bool:
1
        11 11 11
2
3
        vérifie si x est pair
4
5
        Args:
             x (int): entier
6
7
8
        Returns:
9
             bool: True si x est pair
        11 11 11
10
11
        if x % 2 == 0:
             return True
12
13
        else:
             return False
14
15
16
    # appel de la fonction
17
   print(est_pair(5))
```

ercice 2

ercice 3

ercice 4

xercice 5

kercice 0

ercice r

. .

cercice 10

- 2. Exercice 2

Exercices fonctions

correction

```
def valeur absolue(x: int) -> int:
1
        11 11 11
        renvoie la valeur absolue de x
3
        11 11 11
        if x < 0:
5
6
             return -x
        else:
8
             return x
9
10
   print(valeur_absolue(-4))
11
```

Remarque

La docstring est moins détaillée mais toujours présente.

Exercice 2

kercice 4

exercice 5

varcica 0

ercice 9

- 3. Exercice 3

Exercice 3

Exercices fonctions













```
def surface(r: int) -> float:
    """
    renvoie la surface du disque de rayon r
    """
    return 3.14*r**2
    print(surface(5))
```

```
1
   from math import pi
2
   def surface2(r: int) -> float:
        11 11 11
4
5
        renvoie la surface du disque de rayon r
6
        utilise la bibliothèque math
        11 11 11
8
        return pi*r**2
9
   print(surface2(5))
10
```

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercice 7
Exercice 8

- 4. Exercice 4

- Exercice 4

Exercices fonctions

```
def est_majeur(age: int) -> bool:
1
2
        11 11 11
        vérifie si la personne est majeur
3
4
5
        Args:
6
             age (int): âge de la personne
        Returns:
8
             bool: True si majeur
9
        11 11 11
10
11
        if age >= 18:
12
             return True
        else:
13
             return False
14
```

```
def est_majeur2(age: int) -> bool:
1
        11 11 11
2
        vérifie si la personne est majeur
3
4
        Parameters
5
6
        _____
        age : int
8
             âge de la personne.
        Returns
9
10
11
        boolean
12
        11 11 11
13
14
        return age >= 18
```

- rcice 1
- 2. Exercice 2
- 3. Exercice 3
- . -
- T. Exercice
- 5. Exercice 5
- 6. Exercice
- - R Evercice

Exercice 3

Exercice 4

Exercice 5

E

Exc Exc

Exercices fonctions

```
def puissance(x: int, n: int) -> int:
1
3
        élève x à la puissance n
4
5
        Parameters
6
        x : int
8
             entier.
9
        n : int
10
             exposant.
11
12
        Returns
13
14
        res : int
        11 11 11
15
16
        res = 1
        for i in range(n):
17
18
             res *= x
19
        return res
```

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercice 7
Exercice 8
Exercice 9

- 6. Exercice 6

Exercices fonctions

correction

```
1
   from random import randint
2
3
   def lancer_des() -> int:
4
        11 11 11
5
6
        renvoie la somme de deux dés
        11 11 11
        de1 = randint(1, 6)
8
        de2 = randint(1, 6)
9
        return de1+de2
10
```

rcice 3

ercice 4

xercice 5

Exercice 6

ercice /

vercice 10

-
- I. EXCICICC
- 2. Exercice 2
- 3. Exercice 3
- 1 Eversion
- 5 Evereice F
- ____
- 6. Exercice
- 7. Exercice 7
 - cice

- 4
- Exercice 5

 Exercice 7

Exercices fonctions

- ercice 8
- xercice 9
 - rcice 10

```
def pythagore(a: int, b: int, c: int) -> bool:
1
2
3
       vérifie si le triangle a, b, c est
      rectangle
        11 11 11
4
        cotes = a**2+b**2
5
6
        hyp = c**2
7
        if cotes == hyp:
            return True
8
        else:
9
            return False
10
```

ercice 3

ercice 4

kercice 5

xercice 6

Exercice 7

ercice 8

tercice 9

ercice 10

```
def pythagore2(a: int, b: int, c: int) -> bool:
 1
         11 11 11
 2
 3
        vérifie si le triangle a, b, c est rectangle
4
        Parameters
 5
 6
 7
        a, b, c: int
8
             mesures des côtés.
9
10
        Returns
11
12
        Boolean.
13
         11 11 11
14
        return a**2 + b**2 == c**2
15
```

xercice 2

ercice 3

cercice 4

xercice 5

xercice 0

Exercice 7

Exercice 9

- iuii
- 1. Exercice
- 2. Exercice 2
- 3. Exercice 3
- 4 = :
- 4. Exercice
- 5. Exercice
- 6 Exercice
- 8. Exercice 8

- 3
- e 7
- e 8

ice 2 ice 3 ice 4

Exercices fonctions

- xercice 4

 xercice 5

 xercice 6
- Exercice 7

 Exercice 8

 Exercice 9
 - xercice 9
 - ercice 10

```
def somme(n: int) -> int:
    """

renvoie la somme des entiers de 1 à n

"""

somme = 0

for i in range(n+1):
    somme = somme + i

return somme
```

exercice 2

Exercice 8

. . .

exercice 9

ercice 10

- Z. Exercice Z
- 3. Exercice 3
- 4. Exercice 4
- 1. Exercise 1
- 5. Exercice
- 6 Evereice
- 6. Exercice
- /. Exercice
 - rcice

- - E
 - Exercice 8

 Exercice 9

 Exercice 1
 - Exercic

Exercices fonctions

correction

20 / 26

```
1
    def est premier(x: int) -> bool:
        11 11 11
 2
 3
        renvoie True si x est un nombre premier
        11 11 11
 4
        diviseur = 2
 5
 6
        # si le reste est nul, c'est que nous avons un
        diviseur
        while diviseur < x and not(x \% diviseur == 0):
7
            diviseur += 1
8
        # On a divisé par tous les nombres < x
9
        if diviseur == x:
10
11
            return True
        else: # on s'est arrêté avant
12
13
            return False
```

Exercice 10

22 / 26

Exercices fonctions

```
def valeur_proche(a: int) -> int:
1
       11 11 11
       renvoie l'entier inférieur le plus proche
3
4
       de racine(a)
       11 11 11
5
       entier = 1
6
       while entier**2 <= a:
7
            entier += 1
8
       return entier-1
9
```

Exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

ercice 8

```
def racine(a: int) -> float:
1
       11 11 11
2
       calcule une approximation de
3
4
       racine carrée de a
       11 11 11
5
       x = valeur_proche(a)
6
       for i in range(20):
7
            x = 0.5*(x+a/x)
8
9
       return x
```

exercice 2

Exercice 3

Exercice 4

Exercice 5

Exercice 6

kercice 7

ercice 9

```
def triangle(c: int) -> None:
1
        11 11 11
2
3
       Trace un triangle noir
        11 11 11
4
       t.begin_fill()
5
       for _ in range(3):
6
            t.forward(c)
7
            t.left(120)
8
       t.end_fill()
9
```

kercice 2

ercice 3

kercice 4

xercice 5

. .

Xercice 9

```
# programme principal
1
  t.up()
3
  for _ in range(3):
       t.left(90)
4
       t.forward(50)
5
       t.right(90)
6
       triangle(100)
7
  t.done()
8
```