

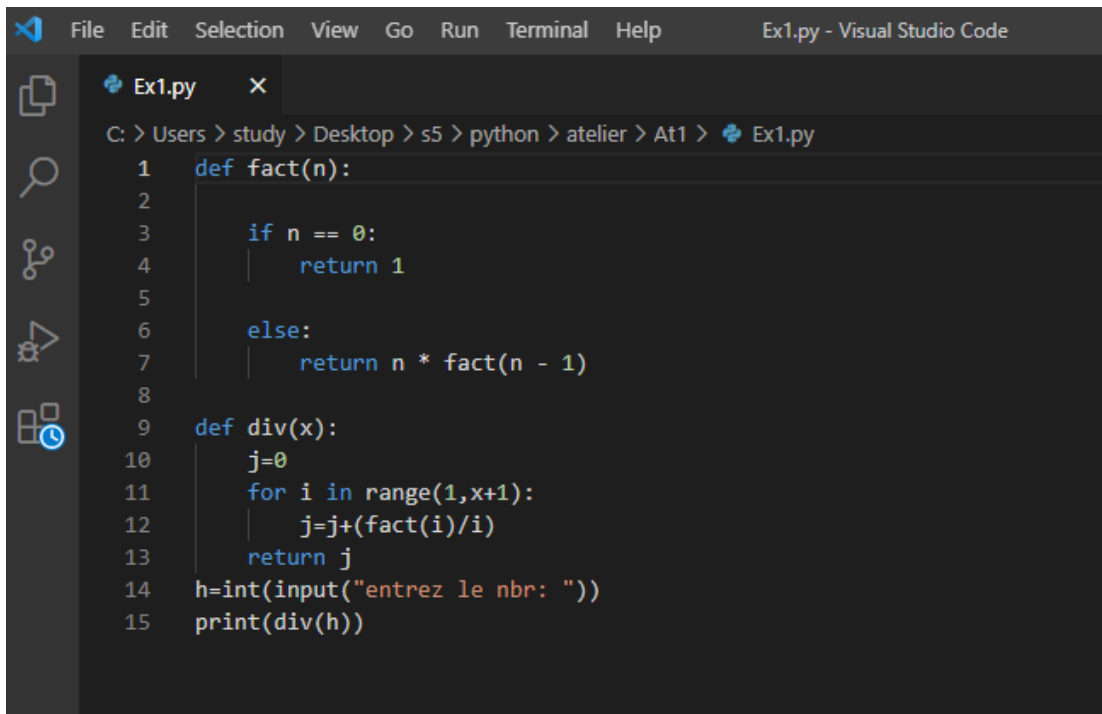
Atelier 1 & 2 : Programmation Python



Atelier 1

Ex1:

la solution

A screenshot of the Visual Studio Code editor interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar shows "Ex1.py - Visual Studio Code". The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The main editor area displays the following Python code:

```
1 def fact(n):
2
3     if n == 0:
4         return 1
5
6     else:
7         return n * fact(n - 1)
8
9 def div(x):
10     j=0
11     for i in range(1,x+1):
12         j=j+(fact(i)/i)
13     return j
14 h=int(input("entrez le nbr: "))
15 print(div(h))
```

- if n == 0:

return 1

Critères de fin de récursivité (cela arrêtera la récursivité)




- **return n * fact(n - 1)**

Détermination de la factorielle par récursivité

- `for i in range(1,x+1): j=j+(fact(i)/i)`

Déclaration de la boucle principale qui réduit au dernier

Exécution du code

| | | |
|---|---|---|
| main.py |    | Shell |
| <pre>1- def fact(n): 2 3- if n == 0: 4 return 1 5 6- else: 7 return n * fact(n - 1) 8 9- def div(x): 10 j=0 11- for i in range(1,x+1): 12 j=j+(fact(i)/i) 13 return j 14 h=int(input("entrez le nbr: ")) 15 print(div(h))</pre> | | <pre>entrez le nbr: 5 34.0 > </pre> |

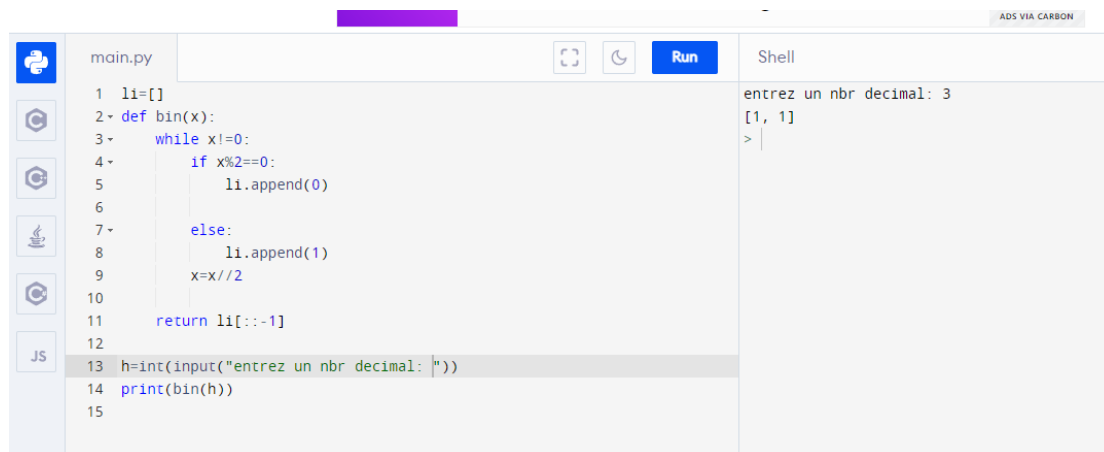
Ex2:

la solution

```
> Users > study > Desktop > s5 > python > atelier > At1 > Ex2.py > ...
1  li=[]
2  def bin(x):
3      while x!=0:
4          if x%2==0:
5              li.append(0)
6
7          else:
8              li.append(1)
9              x=x//2
10
11     return li[::-1]
12
13 h=int(input("dakhel nbr decimal"))
14 print(bin(h))
15 |
```

j'ai travaillé cet exercice avec des listes, (très facile)

Exécution du code



The screenshot shows a code editor with a file named 'main.py'. The code defines a function 'bin(x)' that uses a list 'li' to store the binary digits. It uses a while loop to divide the number by 2, appending the remainder (0 or 1) to the list. The list is then reversed and returned. The main part of the code prompts the user for a decimal number and prints the resulting binary list.

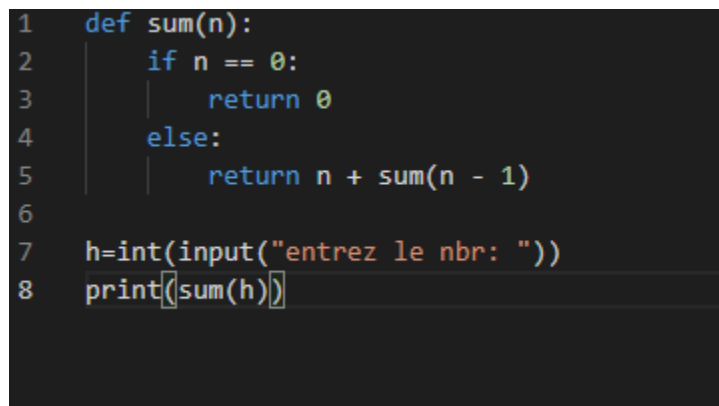
```
1 li=[]
2 def bin(x):
3     while x!=0:
4         if x%2==0:
5             li.append(0)
6
7         else:
8             li.append(1)
9             x=x//2
10
11     return li[::-1]
12
13 h=int(input("entrez un nbr decimal: "))
14 print(bin(h))
15
```

The Shell output shows the user entering '3' and the program outputting '[1, 1]'.

exemple de nbr 3 qui devient 11 en binaire

Ex3: la récursivité (somme de 1 à n)

la solution



The screenshot shows a code editor with a recursive function 'sum(n)'. The function has a base case where if 'n' is 0, it returns 0. Otherwise, it returns 'n' plus the result of 'sum(n - 1)'. The main part of the code prompts the user for a number and prints the result of the recursive sum function.

```
1 def sum(n):
2     if n == 0:
3         return 0
4     else:
5         return n + sum(n - 1)
6
7 h=int(input("entrez le nbr: "))
8 print(sum(h))
```

Boucle de récursivité

def sum(n):




if n == 0:

return 0

else:

return n + sum(n - 1)

Exécution du code




| | | |
|--|---|---------------------------------------|
| main.py |    | Shell |
| <pre>1 def sum(n): 2 if n == 0: 3 return 0 4 else: 5 return n + sum(n - 1) 6 7 h=int(input("entrez le nbr: ")) 8 print(sum(h))</pre> | | <pre>entrez le nbr: 4 10 > </pre> |

exemple valide (somme de 1à4=10)

Ex4: fibonacci

```
1  #fibonacci
2  def fibo(n):
3      if n==0 :
4          return 0
5      elif n==1:
6          return 1
7      else :
8          return fibo(n-2)+fibo(n-1)
9  n=int(input("entre nbr" ))
10 print(fibo(n))
```

Exécution du code

| | | |
|--|---|----------------------------------|
| main.py |    | Shell |
| <pre>1 def fibo(n): 2 if n==0 : 3 return 0 4 elif n==1: 5 return 1 6 else : 7 return fibo(n-2)+fibo(n-1) 8 n=int(input("entre nbr: ")) 9 print(fibo(n))</pre> | | <pre>entre nbr: 5 5 > </pre> |

Ex5: compter les chiffres d'un nombre

```

1  def compt(a):
2      if len(a) > 1:
3          return compt(a[0:len(a) - 1]) + 1
4      else:
5          return 1
6
7  n=input("entre nbr: " )
8  print(compt(n))

```

iterer dans la list (String) on retournant le 1 comme a chaque iteration successif

```

if len(a) > 1:
    return compt(a[0:len(a) - 1]) + 1
else:
    return 1

```

Exécution du code

| | |
|--|------------------------------------|
| <pre> 1 def compt(a): 2 if len(a) > 1: 3 return compt(a[0:len(a) - 1]) + 1 4 else: 5 return 1 6 7 n=input("entre nbr: ") 8 print(compt(n)) </pre> | <pre> entre nbr: 147 3 > </pre> |
|--|------------------------------------|

EX7:inverse les lettre d'une chaîne

```

1  # inverse les lettre d'une chaîne
2  def leb(a):
3      return a[::-1]
4  k=input("entrez des caracteres: \n")
5  print(leb(k))

```

j'ai travaillé avec des pas négatifs pour inverser la liste[::-1]

Exécution du code

| main.py | Shell |
|---|--|
| <pre> 1 # inverse les lettre d'une chaîne 2 def leb(a): 3 return a[::-1] 4 k=input("entrez des caracteres: \n") 5 print(leb(k)) </pre> | <pre> entrez des caracteres: mlk klm > </pre> |

EX8:

Fonction qui détermine le nombre d'occurrences d'éléments donnés

```

1
2  def occ(struct, i):
3      x = 0
4      for k in struct:
5          if k == i:
6              x = x + 1
7      return x
8
9  print(occ([1, 2, 1], 1))

```

for k in struct:

if k == i:

x = x + 1

return x

il parcourt les structures de données et s'il obtient une correspondance sur ces dernières (éléments donnés), il l'incrémentera et le renverra finalement

Exécution du code

```
main.py 2
1 def occ(struct, i):
2     x = 0
3     for k in struct:
4         if k == i:
5             x = x + 1
6     return x
7
8 print(occ([1, 2, 1], 1))
```

Atelier 2

ex1:

l1=[3, 6, 9, 12, 15, 18, 21]

l2=[4, 8, 12, 16, 20, 24, 28]

def imp_pair(l1, l2):

 x=[]

 y = []

 for i in range(0, len(l1)):

 if i%2 != 0:

 x.append(l1[i])

 for j in range(0, len(l2)):

 if j%2 == 0:

 y.append(l2[j])

 return x + y

print(imp_pair(l1,l2))

```

main.py  [Run]  Shell
1  l1=[3, 6, 9, 12, 15, 18, 21]
2  l2=[4, 8, 12, 16, 20, 24, 28]
3- def imp_pair(l1, l2):
4      x=[]
5      y = []
6-   for i in range(0, len(l1)):
7-       if i%2 != 0:
8-           x.append(l1[i])
9-   for j in range(0, len(l2)):
10-       if j%2 == 0:
11-           y.append(l2[j])
12   return x + y
13
14 print(imp_pair(l1,l2))

```

[6, 12, 18, 4, 12, 20, 28]
> |

le meme exemple de l'exercice

ex2:

li=[11, 45, 8, 23, 14, 12, 78, 45, 89]

x=len(li)

j=x//3

li1=li[0:j]

li2=li[j:2*j]

li3=li[2*j:3*j]

print(li1[::-1],li2[::-1],li3[::-1])

le meme exemple de l'exercice

```

main.py  [Run]  Shell
1  li=[11, 45, 8, 23, 14, 12, 78, 45, 89]
2  x=len(li)
3  j=x//3
4
5  li1=li[0:j]
6  li2=li[j:2*j]
7  li3=li[2*j:3*j]
8  print(li1[::-1],li2[::-1],li3[::-1])

```

[8, 45, 11] [12, 14, 23] [89, 45, 78]
> |

j ai utilisé (len) pour afficher le nombre des chiffres dans la liste

et ce nombre je vais le diviser /3 pour obtenir trois listes


```
x=len(li)
```

```
j=x//3
```

```
li1=li[0:j]
```

```
li2=li[j:2*j]
```

```
li3=li[2*j:3*j]
```

ex4:

```
1 set1={23, 42, 65, 57, 78, 83, 29}
2 set2={57, 83, 29, 67, 73, 43, 48}
3 set=set1.intersection(set2)
4 print(set)
5 set1.difference_update(set)
6 print("apres suppression:",set1)
```

intersection() ne conservera que les **éléments** présents dans les deux sets

difference() renverra un nouvel ensemble, qui contient uniquement les **éléments** qui ne sont PAS présents dans les deux sets.

Exécution du code

| main.py | Run | Shell |
|--|-----|--|
| <pre>1 set1={23, 42, 65, 57, 78, 83, 29} 2 set2={57, 83, 29, 67, 73, 43, 48} 3 set=set1.intersection(set2) 4 print(set) 5 set1.difference_update(set) 6 print("apres suppression:",set1)</pre> | | <pre>{57, 83, 29} apres suppression: {65, 42, 78, 23} > </pre> |

ex5:

```
1  l1=[47, 64, 69, 37, 76, 83, 95, 97]
2  dic={'Yassine':47, 'Imane':69, 'Mohammed':76, 'Abir':97}
3  def chang(l1, dic):
4      keys = list(dic.keys())
5      a= []
6      for i in keys:
7          for j in l1:
8              if dic[str(i)] == j:
9                  a.append(j)
10     return a
11
12  print(chang(l1, dic))
```

exemple de l'ex



```
main.py [Run] Shell
1  l1=[47, 64, 69, 37, 76, 83, 95, 97]
2  dic={'Yassine':47, 'Imane':69, 'Mohammed':76, 'Abir':97}
3  def chang(l1, dic):
4      keys = list(dic.keys())
5      a= []
6      for i in keys:
7          for j in l1:
8              if dic[str(i)] == j:
9                  a.append(j)
10     return a
11
12  print(chang(l1, dic))
```

[47, 69, 76, 97]
> |

