

spécialité
NSI

```
document.getElementById(div).innerHTML += "NSI";  
else if (i==2)  
{  
  divs[i];  
  var atpos=inputs[i].indexOf("@");  
  var dotpos=inputs[i].lastIndexOf(".");  
  if (atpos<1 || dotpos<atpos+2 || dotpos>inputs[i].length-1)  
  document.getElementById('errEmail').innerHTML += "use @.com";  
  else  
  document.getElementById(div).innerHTML += "NSI";  
  var atpos=inputs[i].indexOf("@");  
  var dotpos=inputs[i].lastIndexOf(".");  
  if (i==5) {  
    if (atpos<1 || dotpos<atpos+2 || dotpos>inputs[i].length-1)  
    document.getElementById('errEmail').innerHTML += "use @.com";  
  }  
}
```

```
#partie 1 du projet

def bac(moyenne): #je definis la variable 'bac'
    if moyenne>=10:
        return True
    else :
        return False

def mention(moyenne):
    if moyenne<=8 and moyenne>=0:
        print("Recalé")
    elif moyenne >=8 and moyenne<10:
        print ("second groupe")
    elif moyenne >=10 and moyenne<12:
        print ("Recu")
    elif moyenne>=12 and moyenne<14:
        print ("assez bien")
    elif moyenne >=14 and moyenne<16 :
        print ("bien")
    elif moyenne >=16 and moyenne <=20:
        print ("Tres bien")
    else :
        print("valeur incohérente")

"""assert mention((9))=="second groupe)"""
"""assert mention((-1))=="valeur incohérente)"""
```

```
#partie 1 du projet

def bac(moyenne): #je definis la variable 'bac'
    if moyenne>=10:
        return True
    else :
        return False
```

```
In [73]: bac(9)
Out[73]: False
```

```
In [72]: bac(11)
Out[72]: True
```

```
def mention(moyenne):
    if moyenne<=8 and moyenne>=0:
        print("Recalé")
    elif moyenne >=8 and moyenne<10:
        print ("second groupe")
    elif moyenne >=10 and moyenne<12:
        print ("Recu")
    elif moyenne>=12 and moyenne<14:
        print ("assez bien")
    elif moyenne >=14 and moyenne<16 :
        print ("bien")
    elif moyenne >=16 and moyenne <=20:
        print ("Tres bien")
    else :
        print("valeur incohérente")
```

```
"""assert mention(9)=="second groupe)"""
"""assert mention(-1)=="valeur incohérente)"""
```

```
In [78]: mention(9)
second groupe
```

```
In [79]: mention(-1)
valeur incohérente
```

```
nombre=(int(input("nombre de notes? :"))) #je definis
for i in range(nombre) :
    if nombre<=0 :
        (int(input("il faut un nombe plus grand que 0
        print(int(input("nombre de notes? :")))
    elif nombre>0:
        break

somme=0
for i in range(nombre) :
    note=float(input("note ? "))
    somme+=note
print("moyenne = ",somme/nombre)
```

```
nombre de notes? : 5
```

```
note ? 14
```

```
note ? 16
```

```
note ? 12
```

```
note ? 17
```

```
note ? 11
```

```
moyenne = 14.0
```

A photograph of a computer monitor displaying Python code in a dark-themed IDE. The code is for a class named 'AdmissionExtensionOnlineController' which inherits from 'http.Controller'. It defines a method 'type_wise_program' that takes 'self' and '**kwargs' as arguments. The method checks the length of 'kwargs' for a 'types' key. If it's empty, it returns 'None'. Otherwise, it extracts 'types' from 'kwargs' and initializes 'program_list' and 'domain' as empty lists. It then uses a series of 'if' and 'elif' statements to populate 'domain' based on the 'types' value. The 'domain' is a list of course IDs. Finally, it appends the 'state' to 'admission_register_list' and returns it. The code is as follows:

```
1 from odoo import http
2 from odoo.exceptions import ValidationError
3 from datetime import datetime
4 import calendar
5 import math
6 import pytz
7 import io, base64
8
9
10 class AdmissionExtensionOnlineController(http.Controller):
11
12     @http.route('/get/type_wise_program', website=True, auth='none',
13               def type_wise_program(self, **kwargs):
14         if len(kwargs['types']) <= 0:
15             return "None"
16
17         types = kwargs['types']
18         program_list = []
19         domain = []
20
21         if types == 'local_bachelor_program_hsc':
22             domain = [('course_id.is_local_bachelor_program_hsc', '=',
23                       'local_bachelor_program_a_level')]
24         elif types == 'local_bachelor_program_a_level':
25             domain = [('course_id.is_local_bachelor_program_a_level',
26                       '=', 'local_bachelor_program_diploma')]
27         elif types == 'local_bachelor_program_diploma':
28             domain = [('course_id.is_local_bachelor_program_diploma',
29                       '=', 'local_masters_program_bachelor')]
30         elif types == 'local_masters_program_bachelor':
31             domain = [('course_id.is_local_masters_program_bachelor',
32                       '=', 'international_bachelor_program')]
33         elif types == 'international_bachelor_program':
34             domain = [('course_id.is_international_bachelor_program',
35                       '=', 'international_masters_program')]
36         elif types == 'international_masters_program':
37             domain = [('course_id.is_international_masters_program',
38                       '=', 'international_masters_program')]
39
40         domain.append(('state', '=', 'application'))
41         admission_register_list = http.request._request('get',
42                                                         'admission_register_list',
43                                                         {'program_id': program_id})
44         for program in admission_register_list:
45             if program['course_id'] in domain:
46                 program_list.append(program)
```