```
'''
#################################################################################################
#####################################TP-TD 13: Exceptions#########################################
#################################################################################################
------Exemple des exceptions en python------
>>> 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> if (1>10:print('bjr')
  File "<stdin>", line 1
    if (1>10:print('bjr')
           ^
SyntaxError: invalid syntax
>>> 'bjr'+16
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
>>> q
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'q' is not defined
>>> from math import sqrt,log
>>> sqrt(-1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: math domain error
>>> log(-1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: math domain error
>>> l
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'l' is not defined
>>> l=[1,2,3]
>>> l[10]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> d={'a':10}
>>> d['b']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'b'
'''
# ------1er exemple en ténant comùpte des exceptions en python------

 try:
    print('awel star fel try')
    x=10
    x=x*2+20
    i=int(input('donner un entier'))
```

```
    print(i)
    x=x/i
    print(x+1)
    print('akher star fel try')
 except:
    print(x)
    print('saret ghalta qq part')
    print('el ghalta elli fel try fana star?')

 try:
    print('awel star fel try')
    x=10
    x=x*2+20
    i=int(input('donner un entier'))
    print(i)
    x=x/i
    print(x+1)
    print('tout va bien')
    print('akher star fel try')
 except:
    print('saret mochkla')
    print('el mochkla mna3roufhech')
    print('amma metakkdin elli saret mochkla')
    print('wedalil, elli a7na fel blok mta3 l except')

'''
Exercice:
ecrire une fonction 'saisie' qui permet
de saisir et retourner un entier >10.
(Ténir compte des exceptions)
'''
# --- D'une façon générale: On peut avoir des exceptions ----
def saisie():
  while True:
    x=int(input('donner un entrier >10\n'))
    if x>10:break
  return(x)

# --- En ténant compte des exceptions ----
def saisie1():
  try:
    while True:
      x=int(input('donner un entrier >10\n'))
      if x>10:break
  except:
    print('utilisateur n est pas serieux\n')

  return x


# --- selon le besoin, on
# Dans saisie1:Puisque le x à la sortie du boucle while va être directement retourné par l
saisie1, donc on peut les combiner comme suit:
def saisie2():
  try:
    while True:
```

```
      x=int(input('donner un entrier >10\n'))
      if x>10:
        return x
  except:
    print('utilisateur n est pas serieux\n')
    print('pour cette raison bach n3a9bek, tu auras 0\n')
    return 'waw'

'''-------
La fonction saisie3 retourne NoneType puisque
l'instruction "return x" ne s'execute jamais
à cause de "break"
--------'''
def saisie3():
  try:
    while True:
      x=int(input('donner un entrier >10\n'))
      if x>10:
        break
        return x
  except:
    print('utilisateur n est pas serieux\n')
    print('pour cette raison bach n3a9bek, tu auras 0\n')

'''
Rappel:
Dans une fonction, on peut avoir plusieur "return".
UNE SEULE return qui s'exécute
'''
def g(x):
  if x%3==0:return 1
  if x%5==0: return 2
  if x%7==0: return 'yet9sam 3ala 7'
'''
>>> type(g(1))
<class 'NoneType'>
>>> g(3)
1
>>> g(5)
2
>>> g(15)
1
>>> g(105)
1
'''
'''
Exercice:
Ecrire une fonction "operations" de parametre x et y et retourne un dictionnaire dont les
'les operations' et leurs valeurs associées.
exemple:'''
>>> operations(1,2)
{'addition': 3, 'division': 0.5, 'soustraction': -1, 'racine': 1.0, 'multiplication': 2, '
1, 'log': 0.6931471805599453}

# --- Sans ténir compte des exceptions ----
from math import sqrt,log
def operations(x,y):
```

```
  d={}
  d['division']=x/y
  d['addition']=x+y
  d['soustraction']=x-y
  d['racine']=sqrt(x)
  d['multiplication']=x*y
  d['puissance']=x**y
  d['log']=log(3-x)
  return d

# --- En ténant compte des exceptions en générale ----
def operations1(x,y):
  d={}
  try:
    d['division']=x/y
    d['addition']=x+y
    d['soustraction']=x-y
    d['racine']=sqrt(x)
    d['multiplication']=x*y
    d['puissance']=x**y
    d['log']=log(3-x)
  except:
    print('il y a eu une erreur')
  return d

# --- En ténant compte des exceptions détaillées----
from math import sqrt,log
def operations2(x,y):
  d={}
  try:
    d['addition']=x+y
    d['division']=x/y
    d['soustraction']=x-y
    d['racine']=sqrt(x)
    d['multiplication']=x*y
    d['puissance']=x**y
    d['log']=log(3-x)
  except ZeroDivisionError:
    print('ta9ra prepa wta9sem 3ala 0')
  except ValueError:
    print('ta9ra prepa wmata3rafch domaine de definition')
  except TypeError:
    print('ta9ra prepa wmata3rafch le types')
  except:
    d['okhra']='ghalta ma 9rinech 7sabha'
    print('ghalta okhra')
  return d
```