LES EXCEPTIONS

Mr. CHAOULID

1. Définition

- On distingue deux types d'erreur en programmation:

  - Les erreurs de syntaxe, qui sont détectées par le compilateur et peuvent être corrigées par le programmeur avant l'exécution du programme.

    Exemple : a=in(input())

  - Le deuxième type d'erreurs concerne les erreurs d'exécution qui ne sont pas signalées par le compilateur et apparaissent lorsque le programme est exécuté. Elles surviennent en mode Run ou lors de l'utilisation de l'exécutable, une instruction ne peut pas être effectuée. Le programme s'arrête brutalement, c'est très gênant!! Pour l'utilisateur c'est un 'BUG'.

    Exemple : division par zéro, saisie d'un caractère au lieu d'un nombre,……

- Ces erreurs sont connues par le mot **exception**. Les exceptions causent l'arrêt du programme lors de son exécution qui est une chose désagréable surtout pour un utilisateur non expérimenté.
- Pour gérer ce type des erreurs, il y a des instructions qui permettent de contourner les portions du code susceptibles de générer une exception.

2. Les instructions TRY…EXCEPT :

- La gestion d'une exception se fait selon le schéma suivant :

```
try:
    #Code susceptible de lever une exception
except:
    #Code exécuté si une exception se produit
```

- La machine essaye d'exécuter toutes les instructions du bloc try, si aucune des instructions ne génèrent une exception, elles s'exécutent toutes les unes après l'autre, cependant si une instruction produit une erreur, il saute au bloc except qui gère l'exception levée.

3. Exemple:

- **Sans gestion d'exception:**

```
print("Donner votre age")
age=int(input())
print("votre age est",age)
```

- **Résultats:**

```
Donner votre age
a
Traceback (most recent call last):
  File "c:\Users\User\Desktop\POO\chap4.py", line 3, in <module>
    age=int(input())
ValueError: invalid literal for int() with base 10: 'a'
```

3. Exemple:

- **Avec gestion d'exception:**

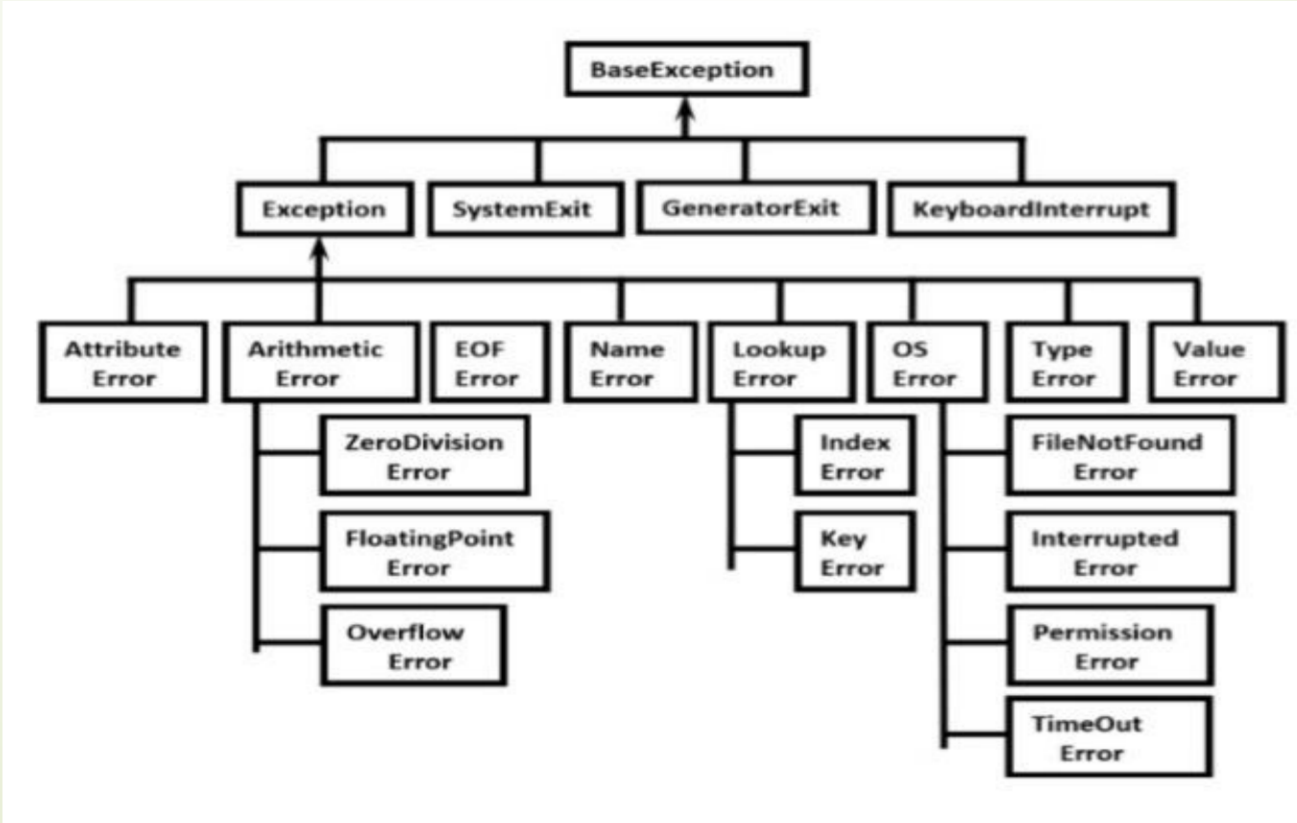
```
try:
    print("Donner votre age")
    age=int(input())
    print("votre age est",age)
except:
    print("Age invalide!!!")
```

- **Résultats:**

```
Donner votre age
b
Age invalide!!!
```

4. Les types d'exceptions :

- Python possède de nombreuses classes d'exceptions natives et toute exception est une instance (un objet) créée à partir d'une classe exception.
- La classe d'exception de base pour les exceptions natives est **BaseException**



4. Les types d'exceptions :

- En écrivant la clause *except* seulement, on indique qu'on veut gérer tous les types d'exceptions. Si le code de la clause **try** est susceptible de générer plusieurs types d'exceptions, on peut vouloir être plus précis en gérant les exceptions avec plusieurs clauses **excpet.** .
- **Exemple:**

```
try:
    print("Donner le premier nombre")
    a=float(input())
    print("Donner le deuxieme nombre")
    b=float(input())
    c=a/b
    print("Le résultat de la division est",c)
except ValueError:
    print("Erreur de conversion!!")
except ZeroDivisionError:
    print("Division par zéro impossible")
except :
    print("Erreur !!")
finally:
    print("Fin du calcul")
```

5. Générer une exception

- Générer une exception si une donnée est invalide.
- Il est possible de générer une erreur dans un programme grâce à l'instruction **raise**
- **Exemple: Générer une exception si le champ âge d'une personne est invalide**

```
class Personne:

    def __init__(self,n,p,a):
        self.__nom=n
        self.__prenom=p
        self.set_age(a)
……….
     def get_age(self):
         return self.__age
     def set_age(self,a):
         if(isinstance(a,int) and a>0):
             self.__age=a
         else:
             raise Exception("Age invalide")
……..
```

FIN