Wang Pengchao G1 TDA

Compte rendu TP3

Gestion d'un parking

Class Parking :

Tout d'abord , on crée la classe Parking et sa construction __init__ avec les objets (adresse,capaccueil et nbreportail).

```
class Parking:
  def __init__(self,adresse,capaccueil,nbreportail):
    assert isinstance(adresse,str) and len (adresse)>0
    assert isinstance(capaccueil,int) and len (capaccueil) >0
    assert isinstance(nbreportail,int) and len (nbreportail) >0
    self.adresse=adresse
    self.capaccueil=capaccueil
    self.nbreportail=nbreportail
```

Puis, on fait les test assert pour adresse ,capaccueil et nbreportail :

```
assert isinstance(adresse,str) and len (adresse)>0
assert isinstance(capaccueil,int) and len (capaccueil) >0
assert isinstance(nbreportail,int) and len (nbreportail) >0
```

Ensuite, on entre les nombres de portail avec "for i in range":
On utilise la méthode append pour ajouter l'élément Portail.

```
  self.ports = []
  for i in range(nbreportail):
    self.ports.append(Portail,(i,self))
```

Class Portail :
Tout d'abord , on crée la classe Portail et sa construction __init__ avec les objets (num,parking,nbcentre,nbsortie).

```
class Portail:

  def __init__(self,num,parking,nbrentre,nbsortie):
    assert isinstance(num,int) and num>=0
    assert isinstance(parking)
    self.num=num
    self.parking=parking
    self.nbentre=0
    self.nbsortie=0
```

Puis, on définit la méthode "entre" et "sortie" :
On utilise try/exception pour utiliser une exception, on ajoute +1 pour déclarer si une place a été prise , ou sinon -1 .

```
  def entre(self):
    try:
      self.nbentre=self.nbentre+1
      self.parking.entre()
    except:
      self.nbentre=self.nbentre-1

  def sortie(self):
    try :
      self.nbsortie=self.nbsortie+1
      self.parking.sortie()
    except:
      self.nbsortie=self.nbsortie-1
```

Class Allen

On crée une classe Allen avec la construction (d ,f)
On fait un test avec assert pour d et f avec int

```
class Allen:
  def __init__(self,d,f):
    assert isinstance(d,int)
    assert isinstance(f,int)
    self.d=d
    self.f=f
```

Puis , on definit la méthode avant avec les paramètre other.
On définit une condition où d et plus petit que f pour True, sinon False

```
  def avant(self,other):
    if self.d < self.f :
      return True
    else:
      return False
```

On remarque que p1 est inférieur à p2, sa affiche “True”

```
> p1=Allen(2,4)
> p2=Allen(6,8)
> p1.avant(p2)
True
>
```