Class_Piggybank_2

Write a class named **piggybank** that produce a piggybank object, that can insert any amount of coin by limit a number of coin in any forms inside piggybank to not be more than 100 coins (Will not receive anymore if insert more) as shown in class structure and example used below.

Structure of class **Complex**	Example used of class **Complex**
class piggybank: **def __init__(self):** # Has variable **self.coins** contains an empty dict at the start # Has *key* as a coin value, and *value* as a number of coins **def add(self, v, n):** # If a number of coin increase by **n** coins and exceed 100, # it will not increase, return **False** as add function has failed # Convert **v** into float first (5 -> 5.0) # If a bank never have any **v** coins, make **self.coins[v]=0** # Call function **self.coins[v] += n** # Return **True** as add function is a success **def __float__(self):** # Multiply coins value with a number of coin for every coin types # Must return only as float (0 -> 0.0) **def __str__(self):** # Return string that show a number of coin of every coin types # Arrange from the lowest to highest coin values	**p1 = piggybank()** **print(int(p1)) # 0** **p1.add(0.25, 4)** # Increase 25 satang by 4 coins **print(float(p1)) # 1.0** **p1.add(0.50, 1)** # Increase 50 satang by 1 coins **print(float(p1)) # 1.5** **p1.add(10, 1)** # Increase 10 bath by 1 coins **print(float(p1)) # 11.5** **print(p1) # {0.25:4, 0.5:1, 10.0:1}** **print(p1.add(10, 1)) # True** Can increase **print(float(p1)) # 21.5** **print(p1.add(1,94)) # False** Can't increase exceed 100 coins **print(float(p1)) # 21.5**

Method **__float__** is used when we call **float(p)** when **p** is **piggybank**, get an output as **float** represent a value of **p**

Method **__str__** is used when we call **str(p)** when **p** is **piggybank**, get an output as **string** represent a value of **p**

Grader Submission

Put these lines of codes below after **class piggybank** as shown above before submit to grader for checking

```
cmd1 = input().split(';')
cmd2 = input().split(';')
p1 = piggybank(); p2 = piggybank()
for c in cmd1: eval(c)
for c in cmd2: eval(c)
```

Input

Various function in order to test a class

Output

Return output from program above that rely on class **Piggybank**

Example

Input (from keyboard)	Output (on screen)
`p1.add(1.11,2); print(float(p1), p1)` `print(float(p2), p2)`	`2.22 {1.11:2}` `0.0 {}`
`p1.add(0.25,1);p1.add(5,1);p1.add(0.25,2);p1.add(5.0,1)` `print(float(p1), str(p1))`	`10.75 {0.25:3, 5.0:2}`

`p1.add(0.25,1); print(p1.add(0.25,100))`
`print(p1.add(0.25,99)); print(float(p1))`
	`False`
	`True`
	`25.0`