````
---

### # 01.00.00

```

from book import Book
from recipe import Recipe

```

---

### # 01.00.01

```

Recipe("cooki", 0, 10, ["dough", "sugar", "love"], "deliciousness incarnate", "dessert")

```

---

### # 01.00.02

```

Recipe("cooki", 1.5, 10, ["dough", "sugar", "love"], "deliciousness incarnate", "dessert")

```

---

### # 01.00.03

```

Recipe("cooki", 1, 10, [], "deliciousness incarnate", "dessert")

```

---

### # 01.00.04

```

Recipe("cooki", 1, 10, ["dough", "sugar", "love"], "deliciousness incarnate", "dessert")
print("Congratulations you finally made sime delicous cookies")

```

---

### # 01.00.05

```

b = Book("My seductive recipes")
print(b.creation_date)

# should be the current date and time
````
````

```
print(b.last_update)

# should be the same as the creation date or None
```

01.00.06

```
crumble = Recipe("Crumble" , 1, 25, ["apples", "flour", "sugar"], "delicious", "dessert")
b.add_recipe(crumble)
print(b.last_update)
```

01.00.07

```
b.get_recipe_by_name("Crumble")

# should print the recipe
# AND
# <Recipe object at x>

b.get_recipe_by_name("Liver Icecream")

# The recipe does not exist
# The error must be handled in a justifiable manner
# such as returning None, [], or printing an error message
```

01.00.08

```
print[b.get_recipes_by_types("dessert"](0))

# Should print the Crumble recipe

b.get_recipes_by_types("asdasd")

# The recipe type does not exist, error must be handled in a justifiable manner
# such as returning None, [], or printing an error message
```

01.02.00

```
print(Vector([1. , 2e-3, 3.14, 5.]).values)
```

````
----

### # 01.02.01

```

print(Vector(4).values)

```

----

### # 01.02.02

```

Vector(-1)

```

----

### # 01.02.03

```

print(Vector((10, 12)).values)

```

----

### # 01.02.04

```

print(Vector((3, 1)).values)

```

----

### # 01.02.05

```

v = Vector((1, 1))
print(v.values)

```

----

### # 01.02.06

```

Vector((4, 7.1))

```

----

### # 01.02.07
````

```

v = Vector(4)
print(v.values)

```

01.02.08

```

print(v * 4)

```

01.02.09

```

print(4.0 * v)

```

01.02.10

```

v * "hi"

```

01.02.11

```

v = Vector(4)
v2 = Vector([[1.0], [1.0], [1.0], [1.0]])
print((v + v2).values)

```

01.02.12

```

v + Vector([0.0, 0.0, 0.0, 0.0])

```

01.02.13

```
```

```
v + "hello"
```

01.02.14

```
v + None
```

01.02.15

```
print((v - v2).values != (v2 - v).values)
```

01.02.16

```
Vector(4) / 2
```

01.02.17

```
Vector(4) / 3.14
```

01.02.18

```
Vector(4) / 0
```

01.02.19

```
Vector(4) / None
```

01.02.20

```

None / Vector(4)

```

01.02.21

```

3 / Vector(3)

```

01.03.00

```

txt="This is a simple string for a basic test. Very simple."
for elem in generator(txt, sep=' '):
    print(elem)

for elem in generator(txt, sep='.'):
    print(elem)

for elem in generator(txt, sep='i'):
    print(elem)

for elem in generator(txt, sep='si'):
    print(elem)

```

01.05.00

```

python3 -i the_bank.py

```

01.05.01

```

from the_bank import Account, Bank

bank = Bank()
john = Account(
    'William John',
    zip='100-064',
    brother="heyhey",
    value=6460.0,
    ref='58ba2b9954cd278eda8a84147ca73c87',
```

```
    info=None,
    other='This is the vice president of the corporation',
    lol = "hihi"
)

bank.fix_account(john)

# OR

bank.fix_account('William John')
```

01.05.02

```
john = Account(
    'William John',
    zip='100-064',
    rother="heyhey",
    value=6460.0,
    ref='58ba2b9954cd278eda8a84147ca73c87',
    info=None,
    other='This is the vice president of the corporation',
)
```

01.05.03

```
john = Account(
    'William John',
    zip='100-064',
    rother="heyhey",
    ref='58ba2b9954cd278eda8a84147ca73c87',
    info=None,
    other='This is the vice president of the corporation',
    lol = "lolilol"
)
```

01.05.04

```
bank.add(
    Account(
        'Jane',
        zip='911-745',
        value=1000.0,
        ref='1044618427ff2782f0bbece0abd05f31'
    )
)

jhon = Account(
```

```
    'Jhon',
    zip='911-745',
    value=1000.0,
    ref='1044618427ff2782f0bbece0abd05f31'
)

bank.add(jhon)

print("testing a valid transfer")
print(jhon.value)
bank.transfer("Jane", "Jhon", 500)
print(jhon.value)
```

01.05.05

```
bank.transfer("Jane", "Jhon", 1000)
print(jhon.value)
```