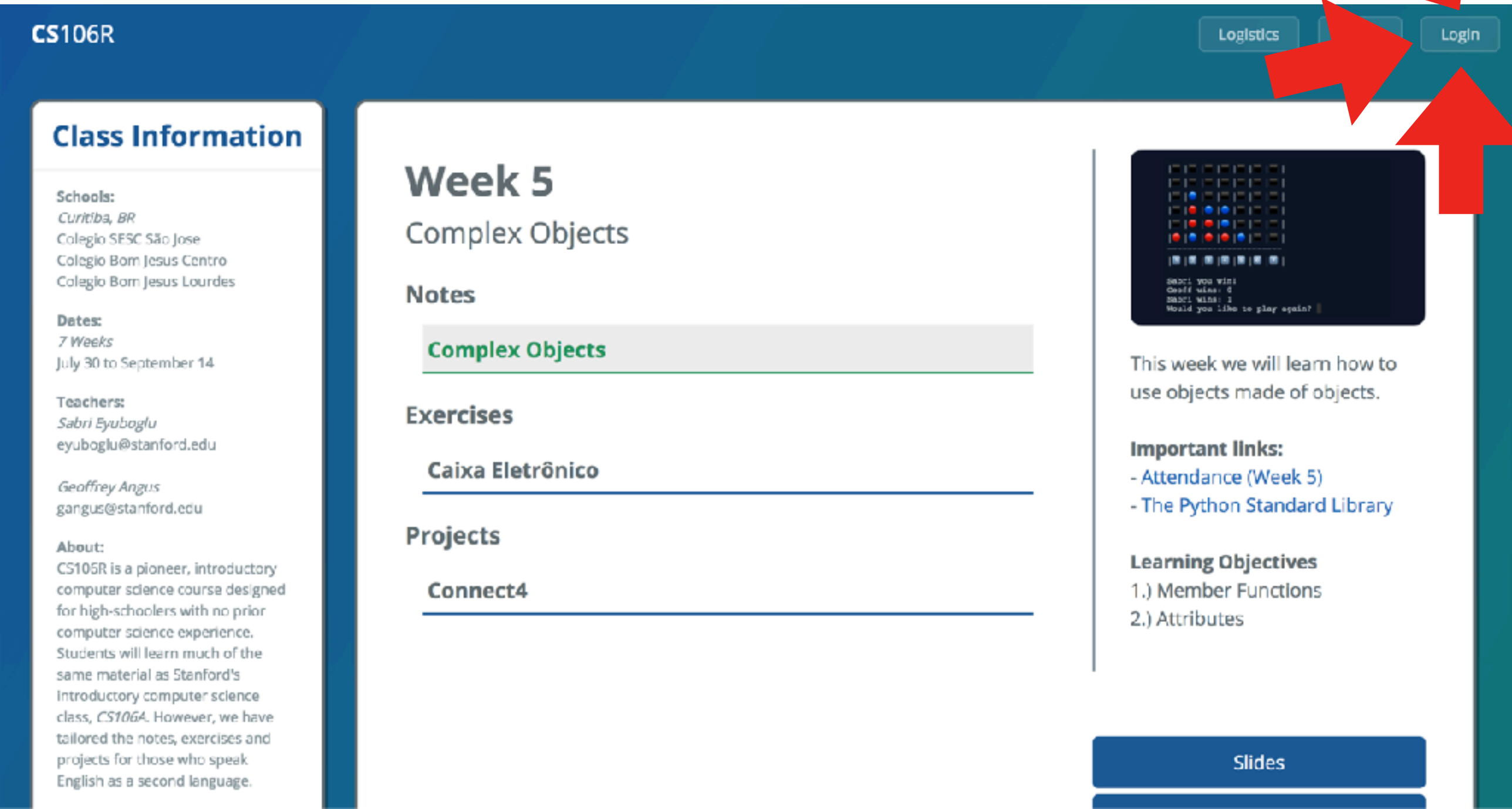
geoffreyangus.github.io/CS106R/



Week 5

CS106R

Sabri **Eyuboglu** & Geoffrey **Angus**

Last week on CS106R...

Scope

Code

```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

Output

```
>
```

Memory

Variables

Objects

Operators

Code

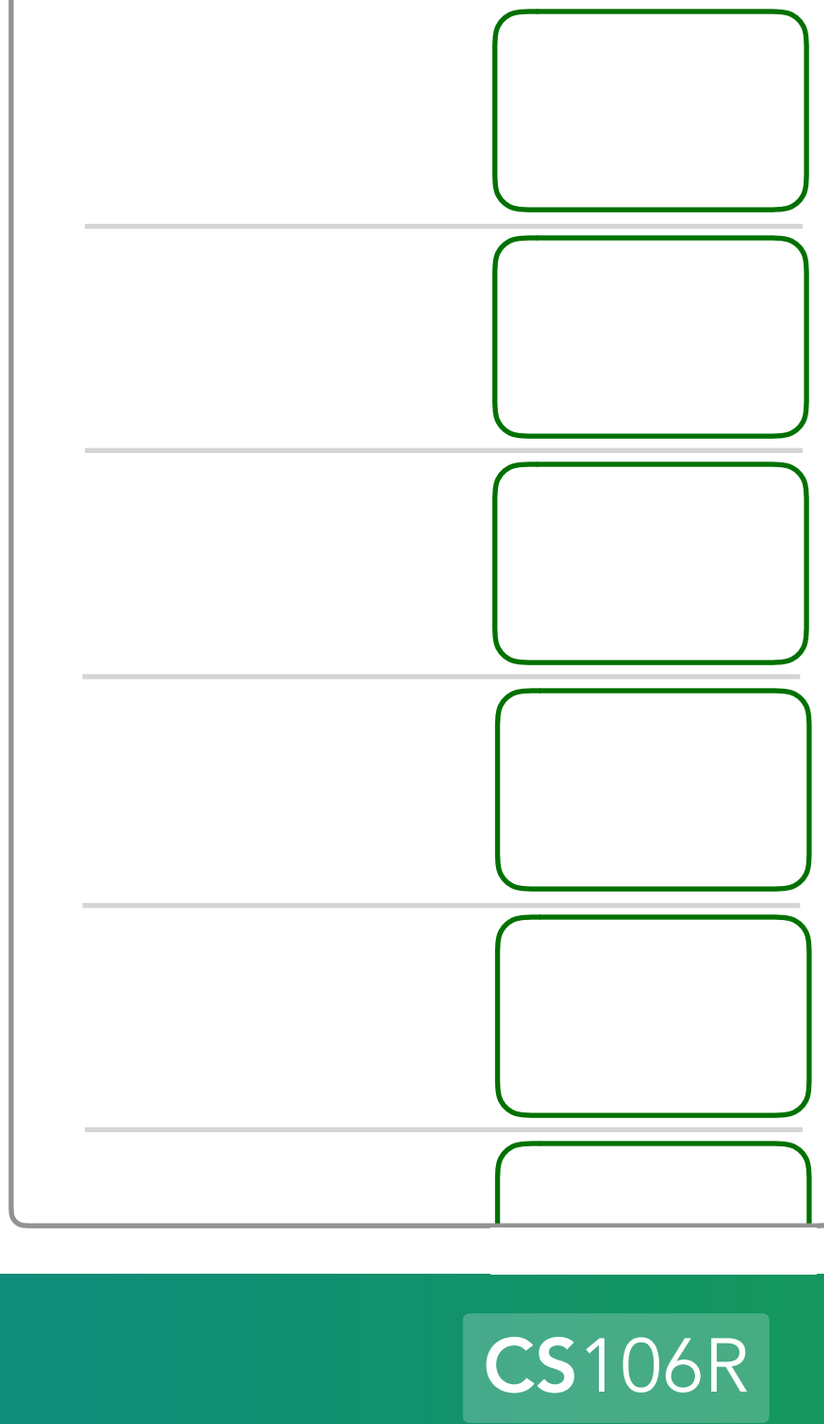
```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

Output

```
>
Enter side one:
```

Memory

Variables | Objects



Operators

Code

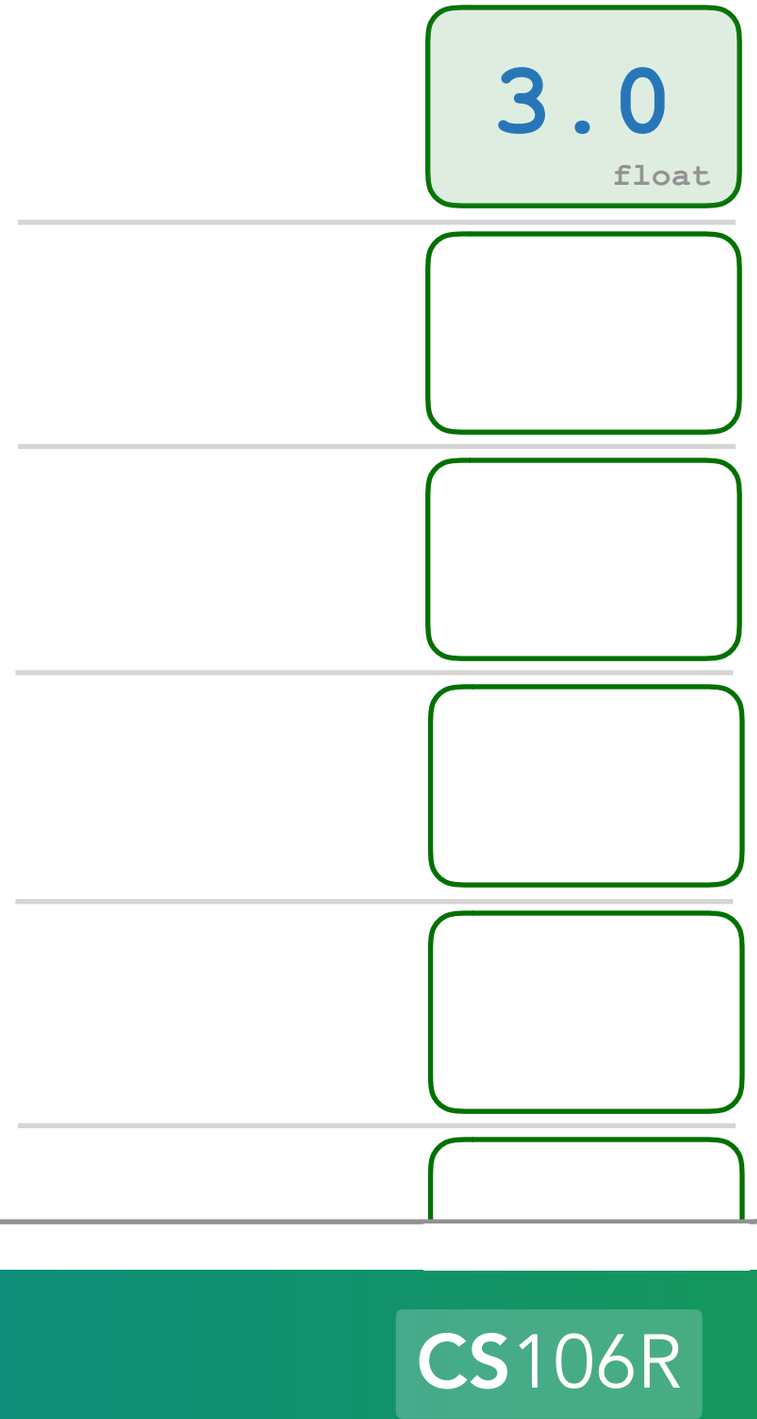
```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

Output

```
>
Enter side one: 3
```

Memory

Variables | Objects



Operators

Code

```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

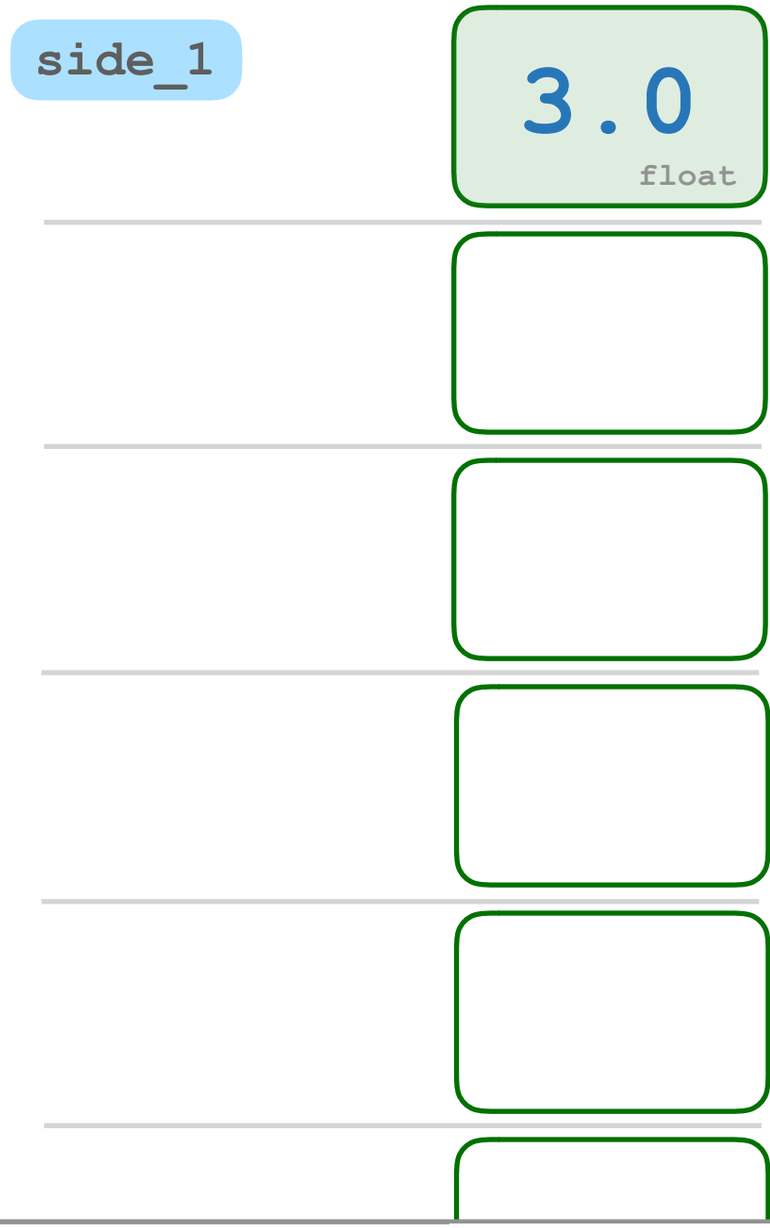
Output

```
Enter side one: 3
```

Memory

Variables

Objects



Operators

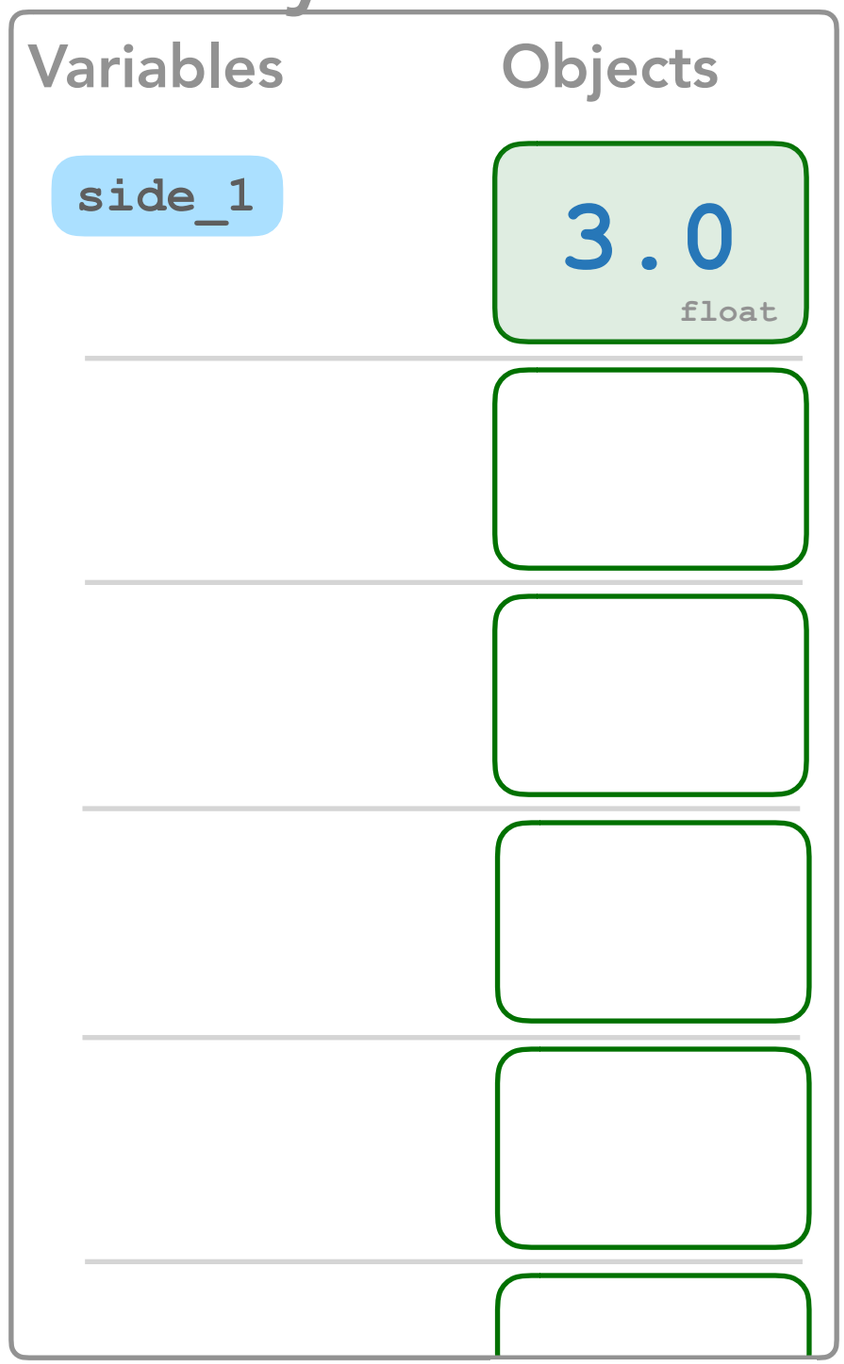
Code

```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two:
```

Memory



Operators

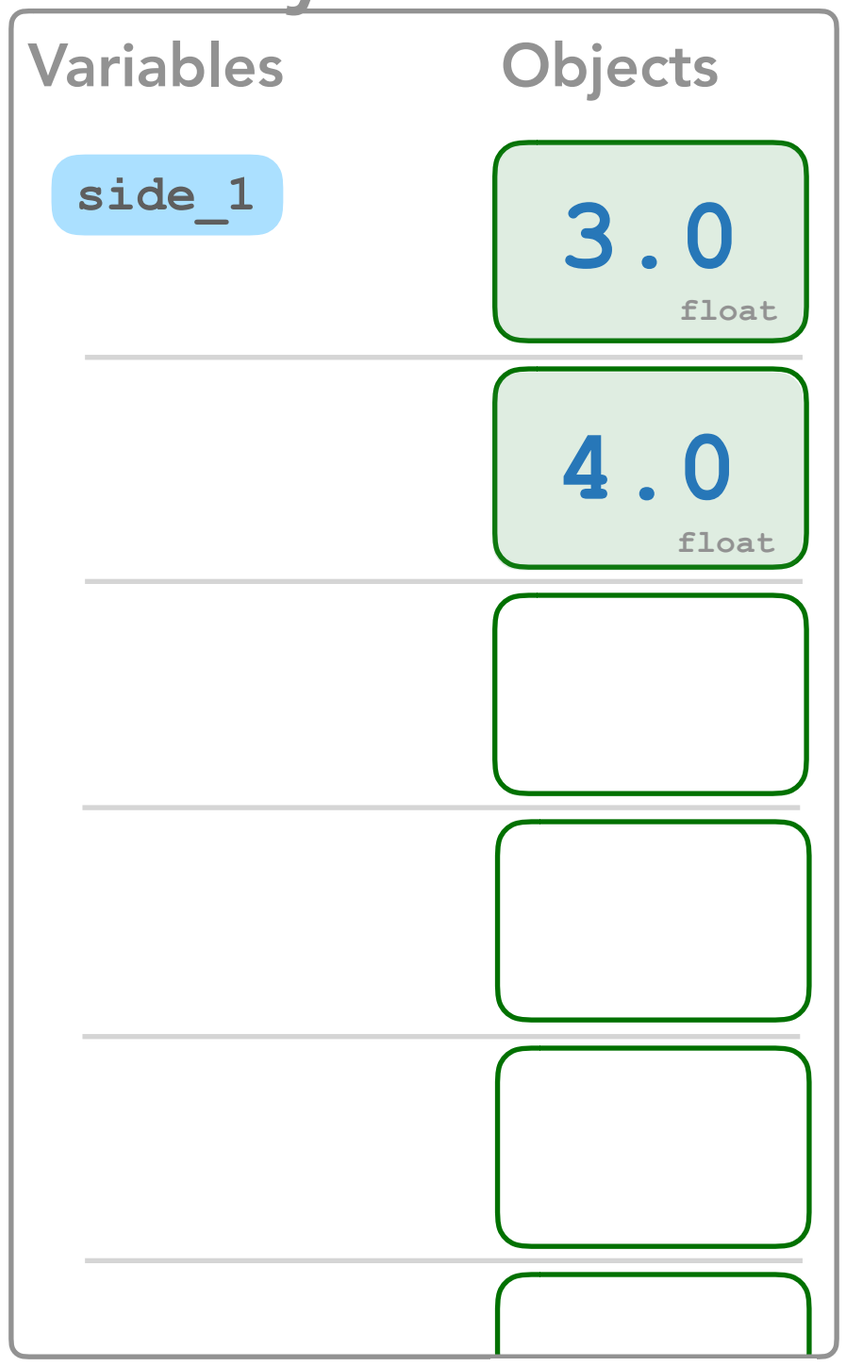
Code

```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

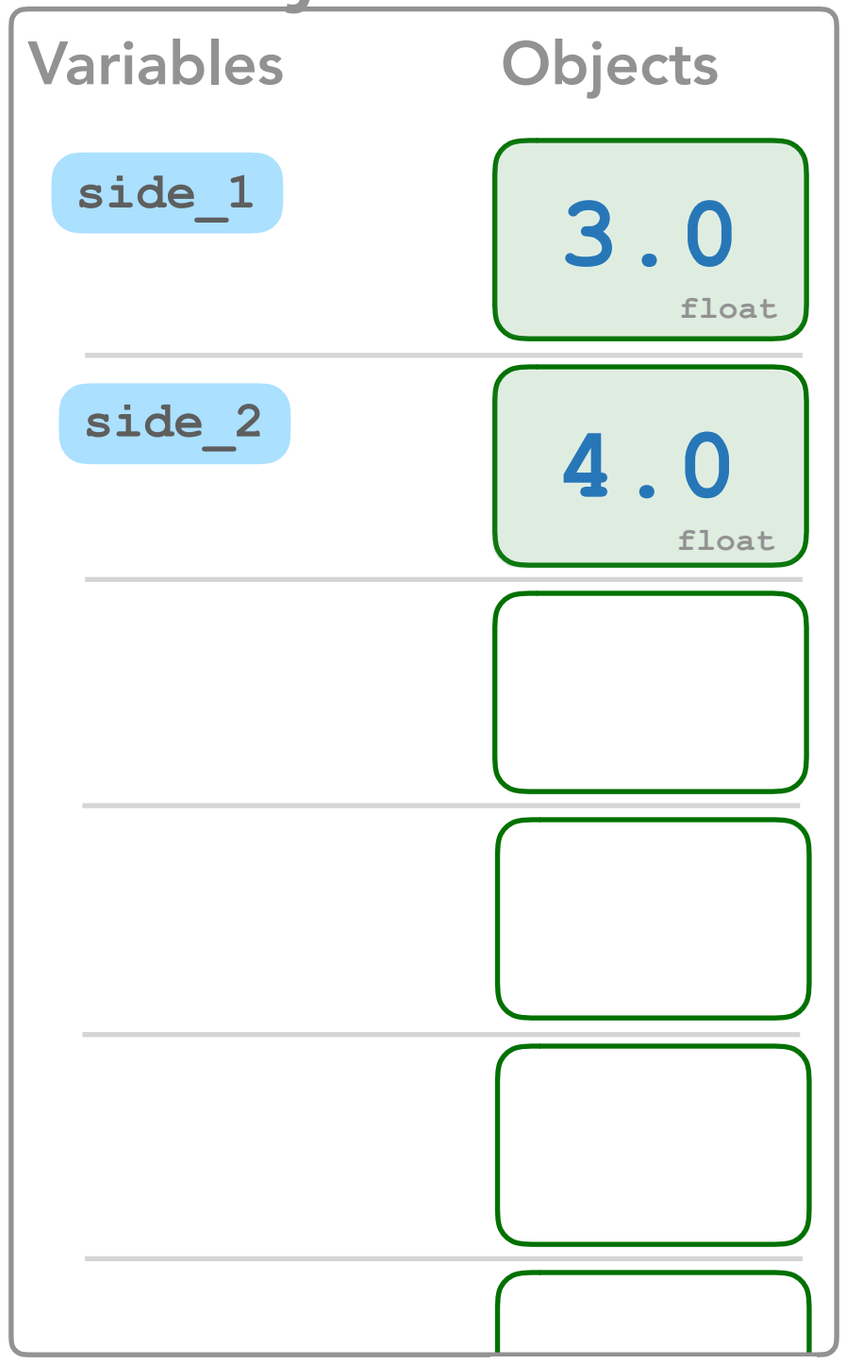
Code

```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

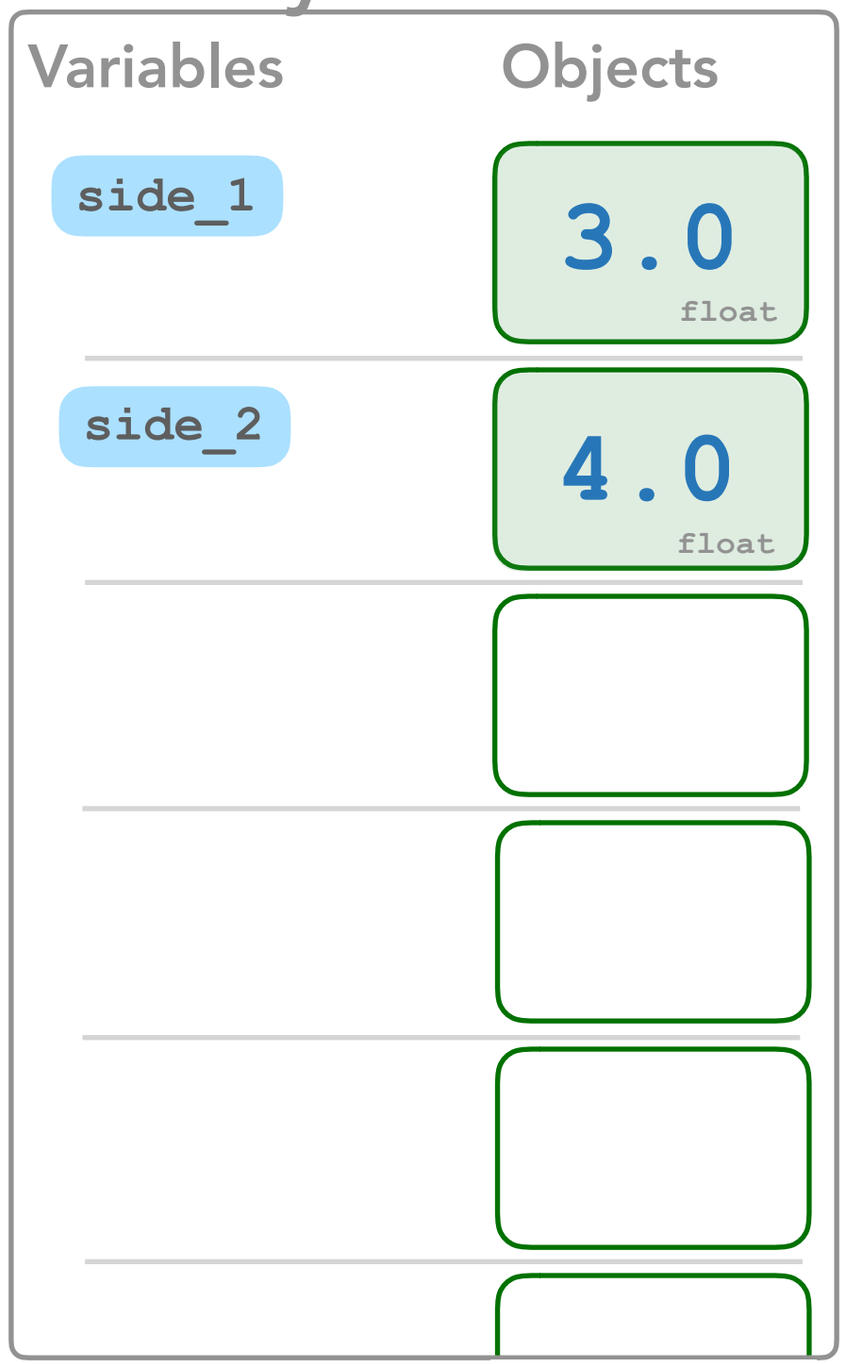
Code

```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

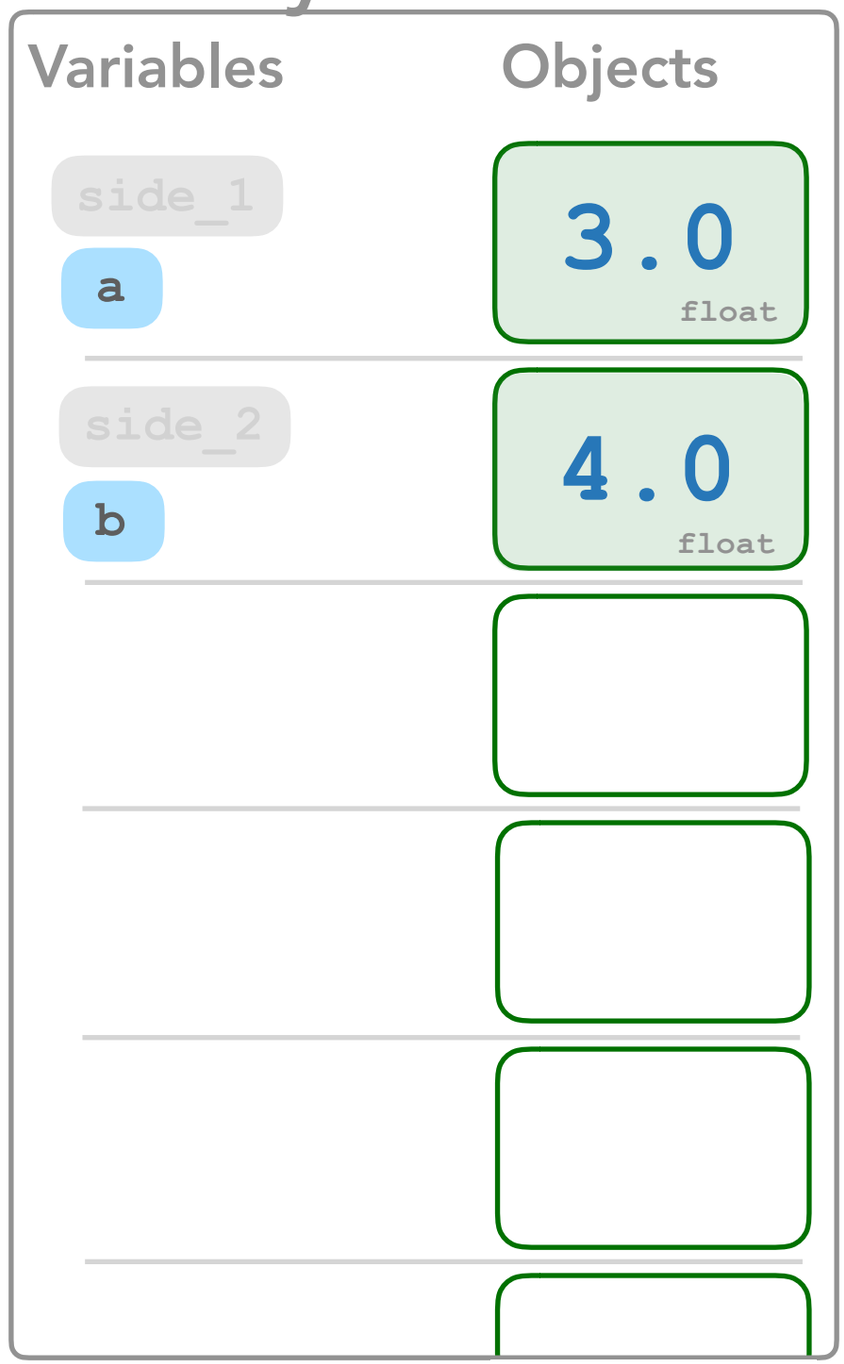
Code

```
def compute_pythag(a, b):
    c_squared = a*a + b*b
    c = square_root(c_squared)
    return c

def main():
    side_1 = input_float("Enter side one:")
    side_2 = input_float("Enter side two:")
    hypotenuse = compute_pythag(side_1, side_2)
    print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

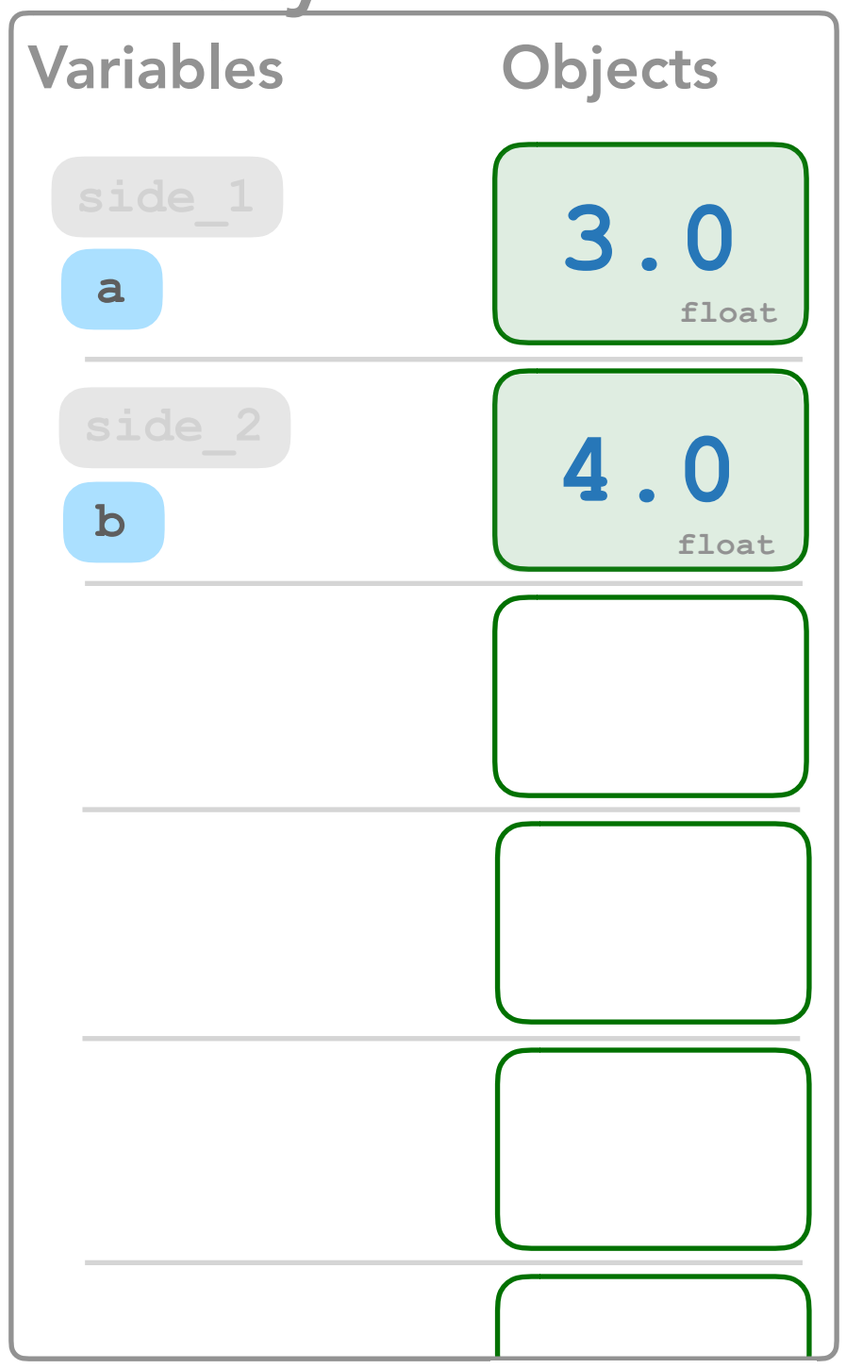
Code

```
def compute_pythag(a, b):
    c_squared = a*a + b*b
    c = square_root(c_squared)
    return c

def main():
    side_1 = input_float("Enter side one:")
    side_2 = input_float("Enter side two:")
    hypotenuse = compute_pythag(side_1, side_2)
    print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

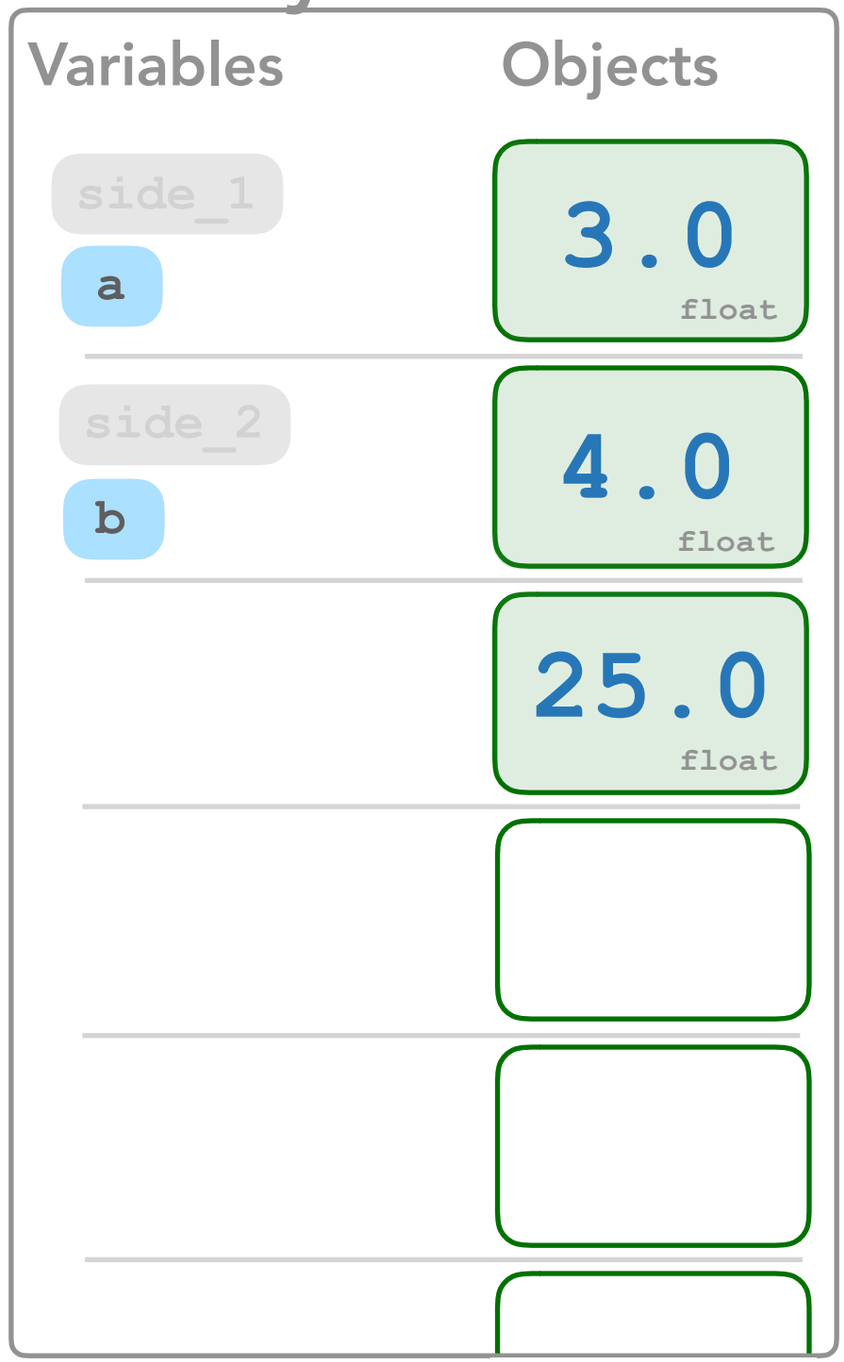
Code

```
def compute_pythag(a, b):
    c_squared = a*a + b*b
    c = square_root(c_squared)
    return c

def main():
    side_1 = input_float("Enter side one:")
    side_2 = input_float("Enter side two:")
    hypotenuse = compute_pythag(side_1, side_2)
    print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

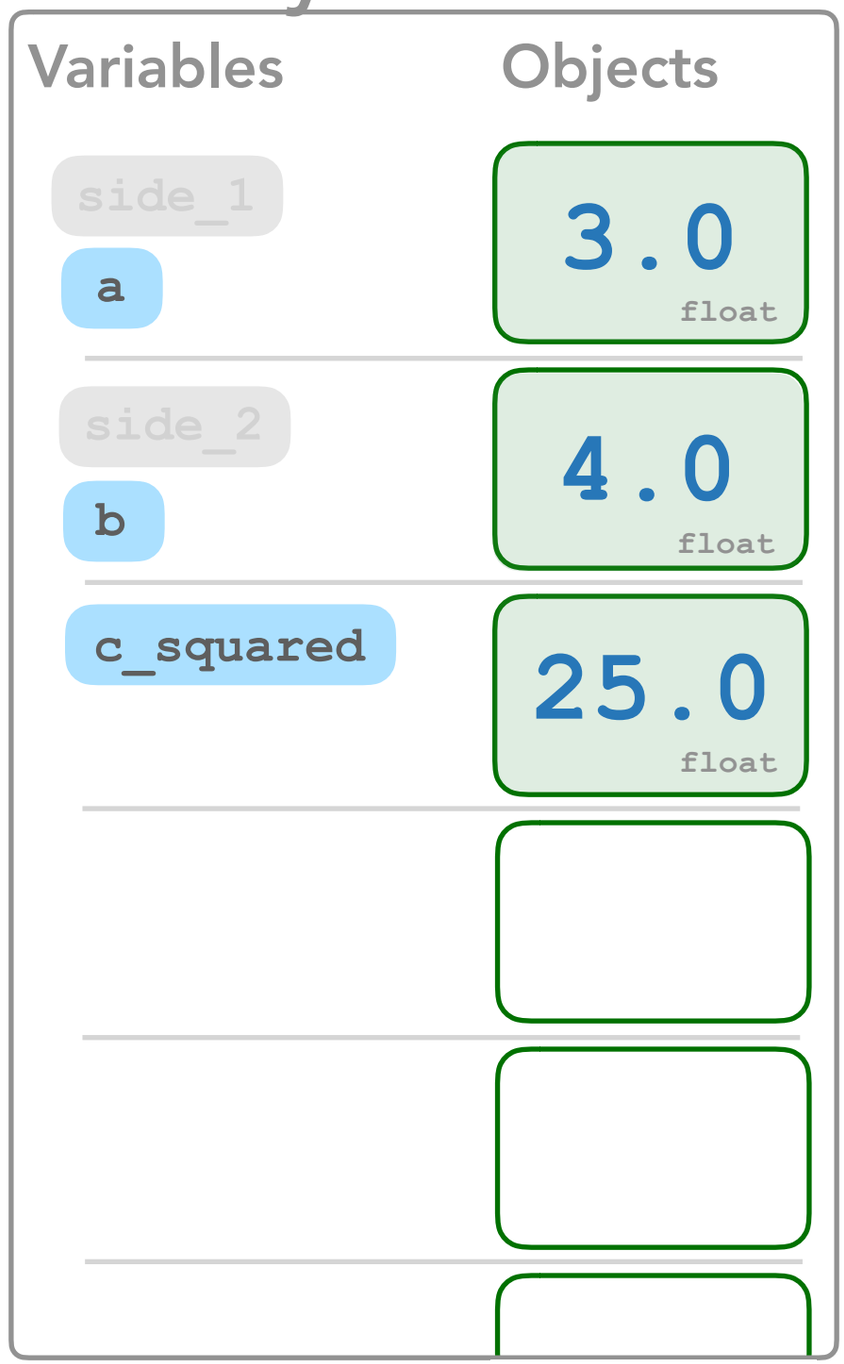
Code

```
def compute_pythag(a, b):
    c_squared = a*a + b*b
    c = square_root(c_squared)
    return c

def main():
    side_1 = input_float("Enter side one:")
    side_2 = input_float("Enter side two:")
    hypotenuse = compute_pythag(side_1, side_2)
    print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

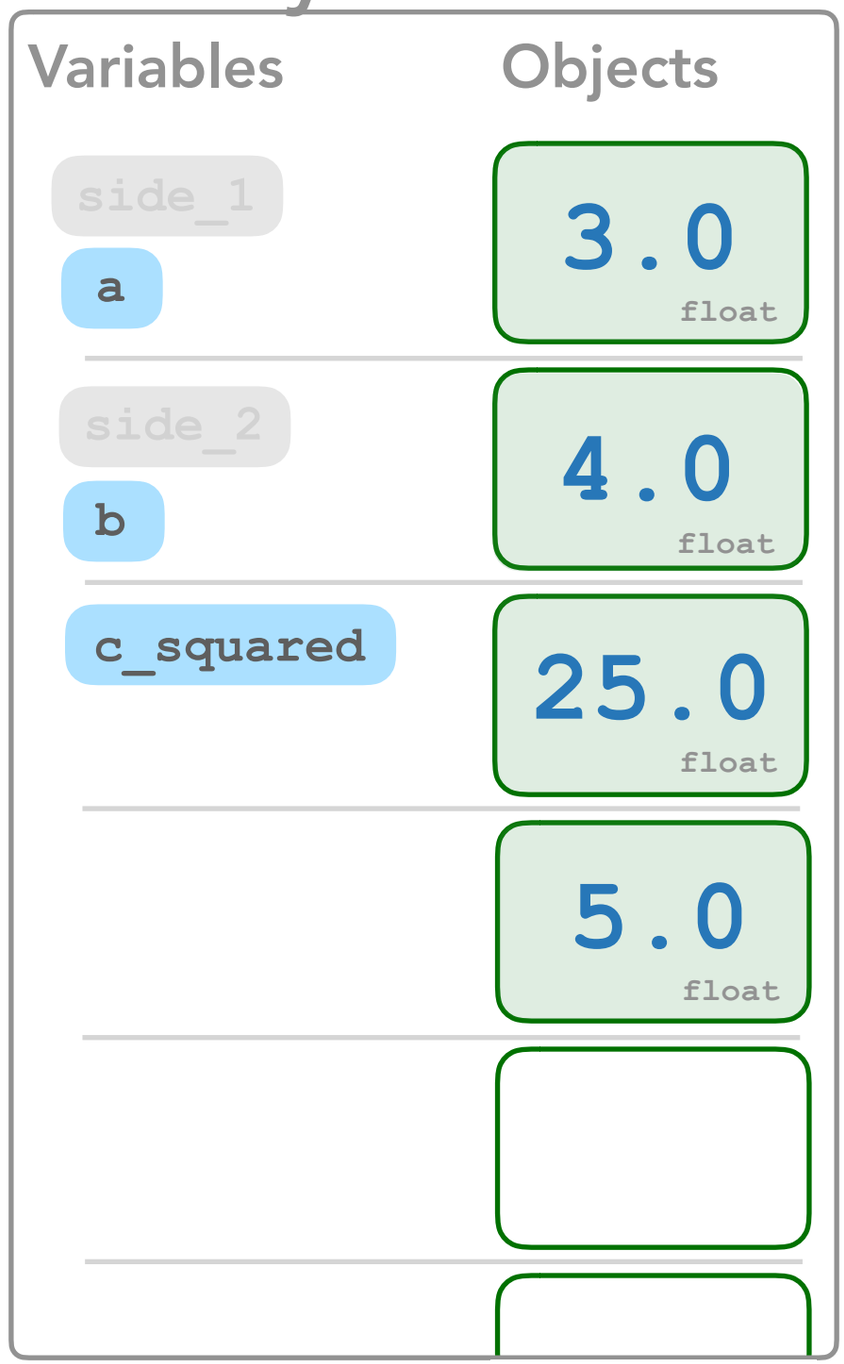
Code

```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

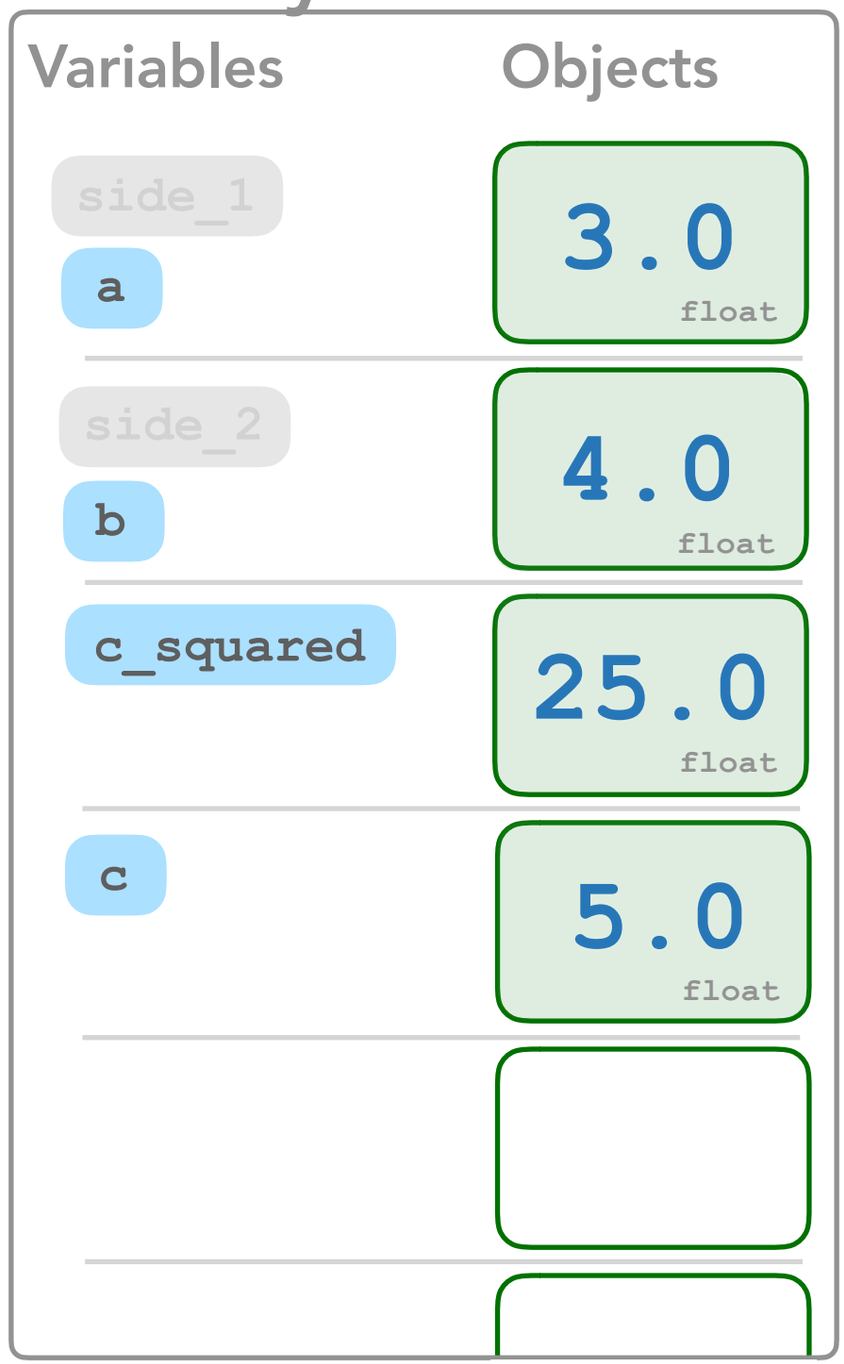
Code

```
def compute_pythag(a, b):
    c_squared = a*a + b*b
    c = square_root(c_squared)
    return c

def main():
    side_1 = input_float("Enter side one:")
    side_2 = input_float("Enter side two:")
    hypotenuse = compute_pythag(side_1, side_2)
    print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

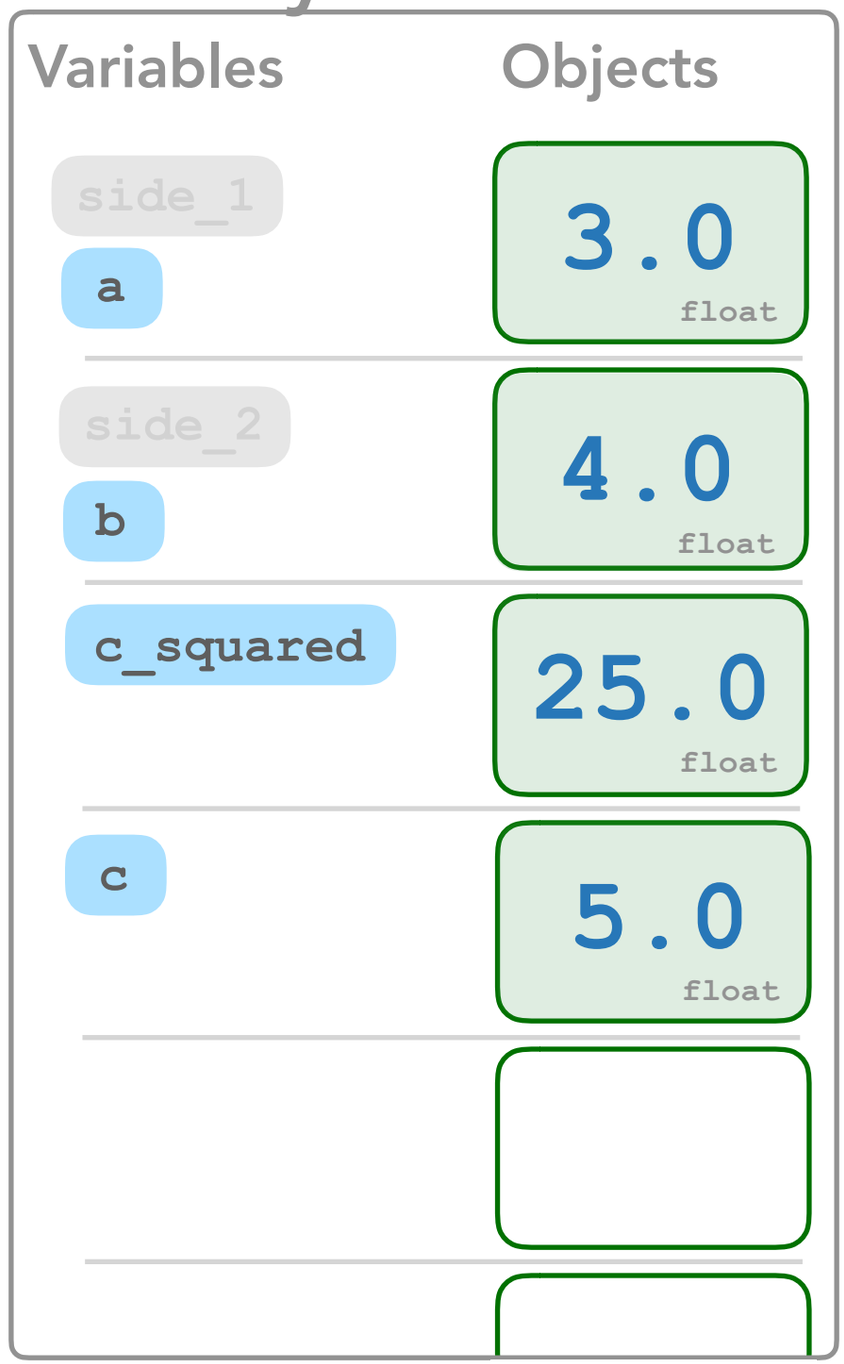
Code

```
def compute_pythag(a, b):
    c_squared = a*a + b*b
    c = square_root(c_squared)
    return c

def main():
    side_1 = input_float("Enter side one:")
    side_2 = input_float("Enter side two:")
    hypotenuse = compute_pythag(side_1, side_2)
    print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

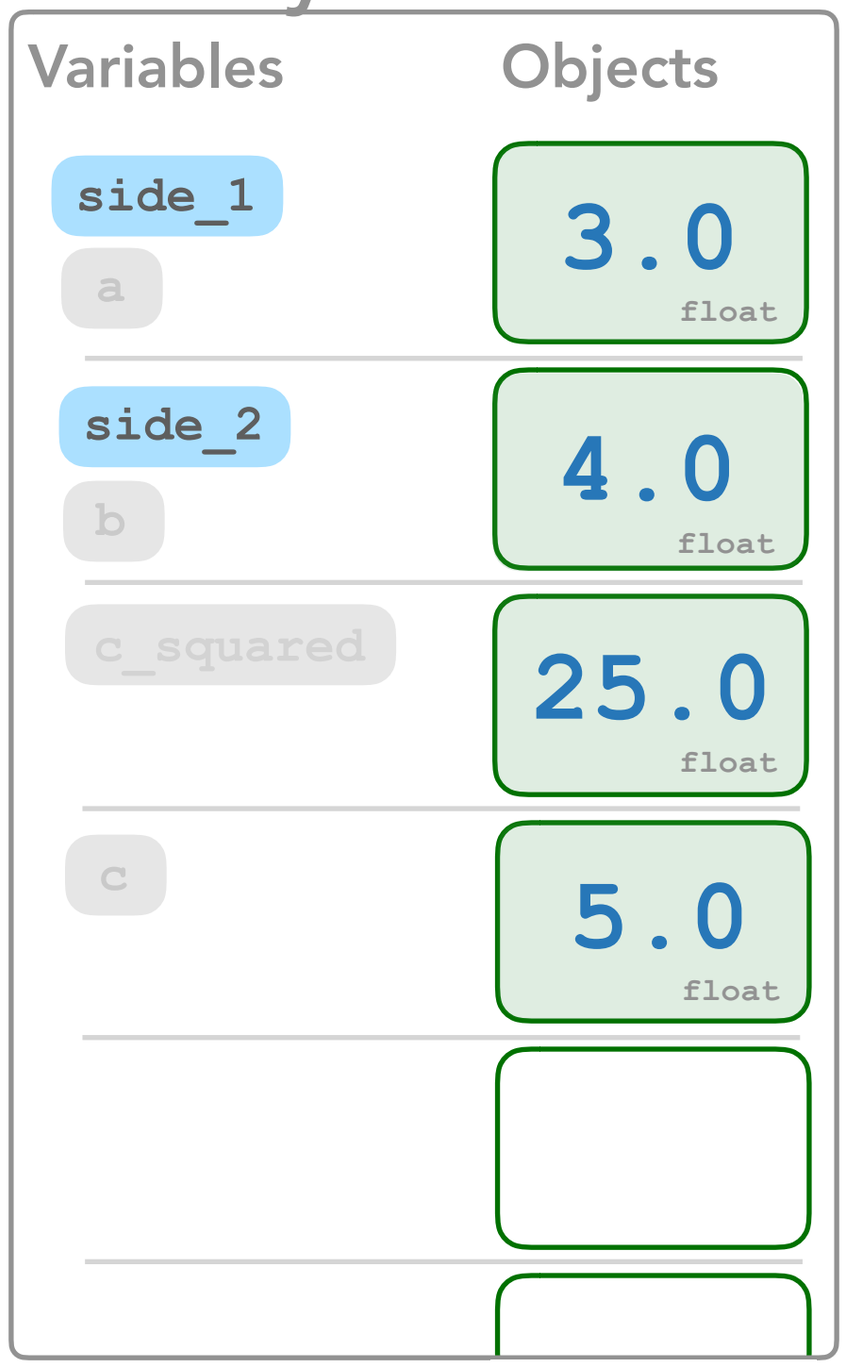
Code

```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

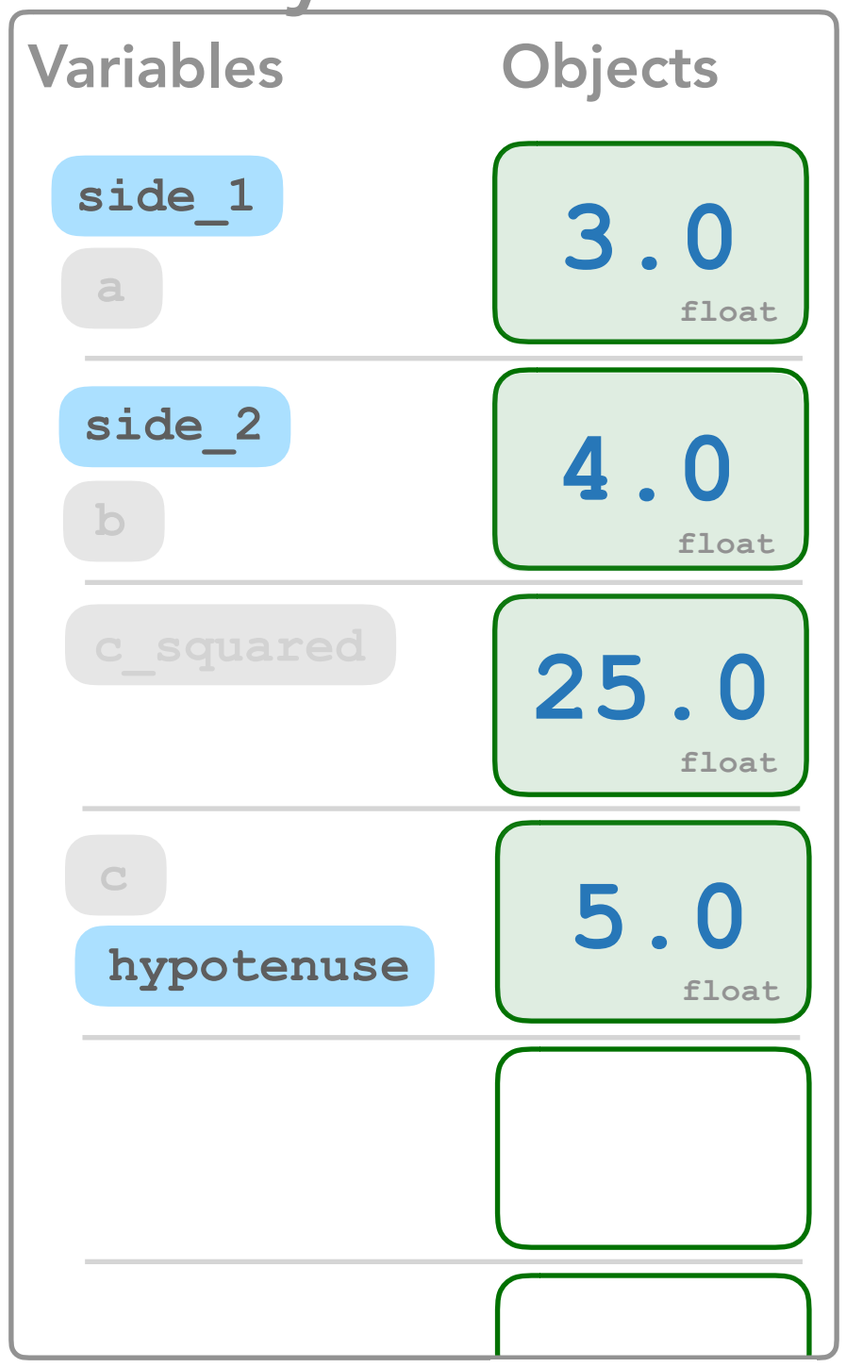
Code

```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

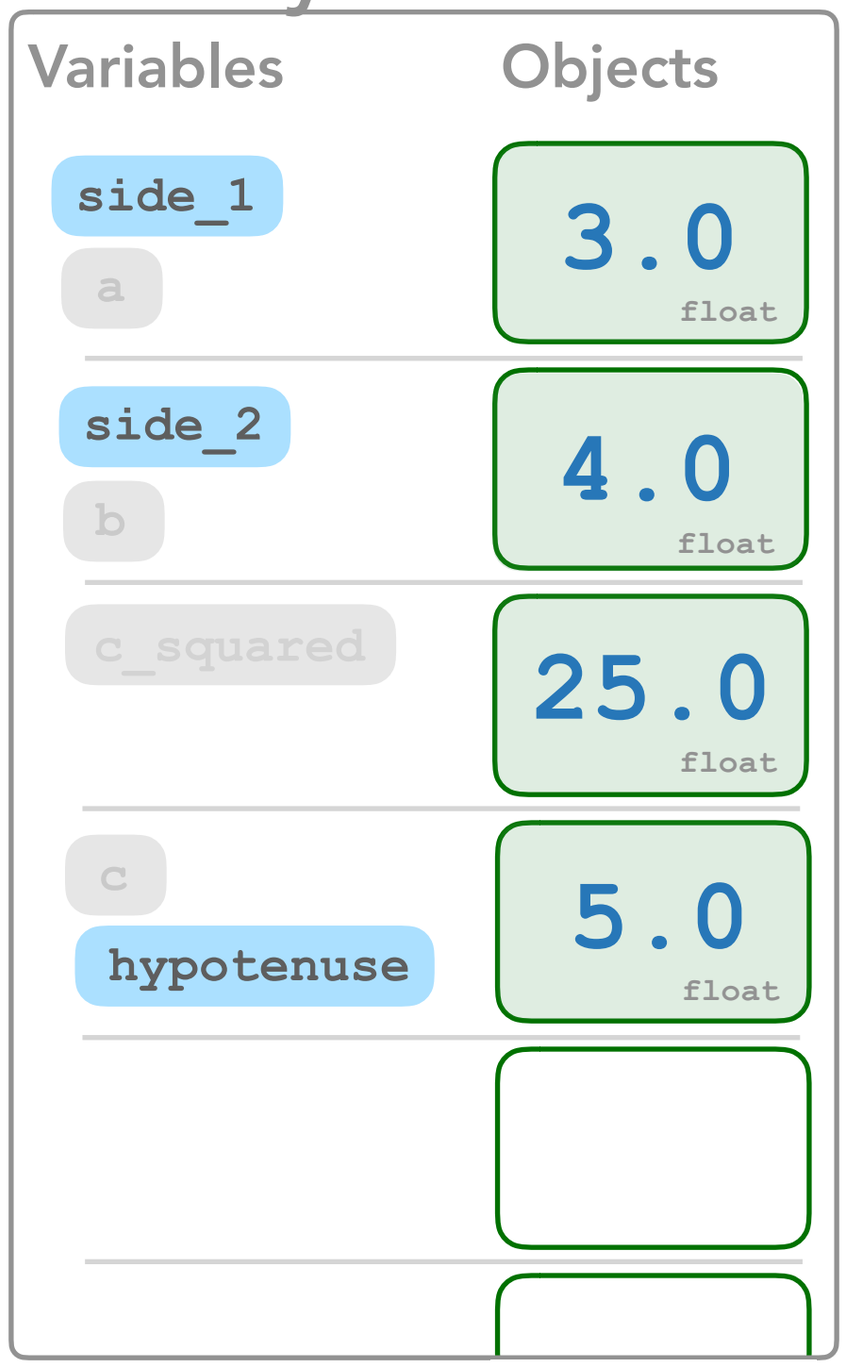
Code

```
def compute_pythag(a, b):
    c_squared = a*a + b*b
    c = square_root(c_squared)
    return c

def main():
    side_1 = input_float("Enter side one:")
    side_2 = input_float("Enter side two:")
    hypotenuse = compute_pythag(side_1, side_2)
    print(hypotenuse)
```

Output

```
>
Enter side one: 3
Enter side two: 4
```

Memory



Operators

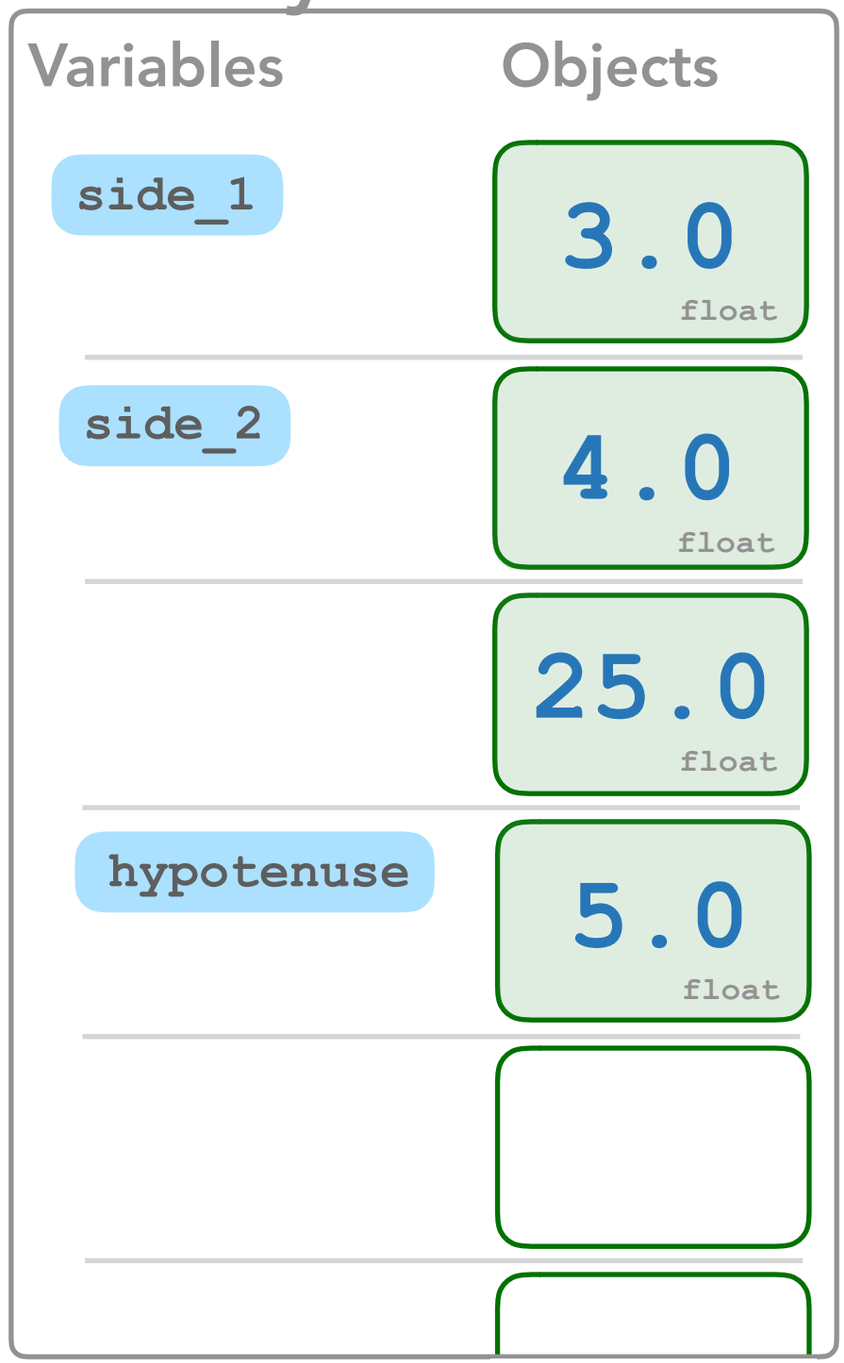
Code

```
def compute_pythag(a, b):
  c_squared = a*a + b*b
  c = square_root(c_squared)
  return c

def main():
  side_1 = input_float("Enter side one:")
  side_2 = input_float("Enter side two:")
  hypotenuse = compute_pythag(side_1, side_2)
  print(hypotenuse)
```

Output

```
Enter side one: 3
Enter side two: 4
```

Memory



Operators

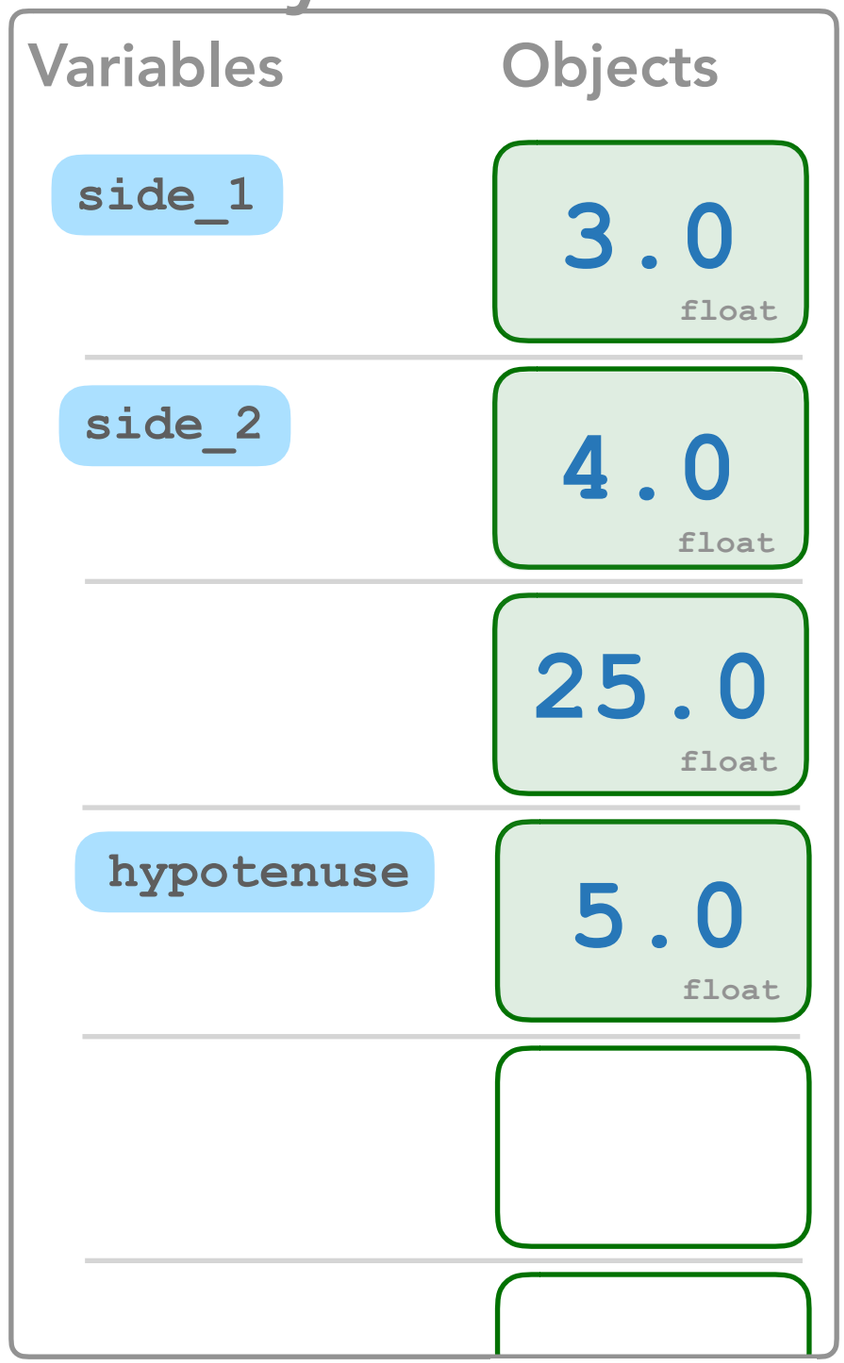
Code

```
def compute_pythag(a, b):
    c_squared = a*a + b*b
    c = square_root(c_squared)
    return c

def main():
    side_1 = input_float("Enter side one:")
    side_2 = input_float("Enter side two:")
    hypotenuse = compute_pythag(side_1, side_2)
    print(hypotenuse)
```

Output

```
Enter side one: 3
Enter side two: 4
5.0
```

Memory



This week on CS106R...

Objects

4 Basic Object Classes

string

Sequences of characters – text

Example

```
"Hello, World!"
```

int

Integers – whole numbers

Examples

```
5   3450   0   -17   1
```

float

Fractional numbers

Examples

```
-5.0   0.174   3.14
```

bool

True or false

Examples

```
True   False
```

There are **more complex** objects out there…

Introducing the

`Bot` class

Let's make GeoffBot **3.0** ...

```
Bot("GeoffBot")
```

```
geoff_bot = Bot("GeoffBot")
```

```
geoff_bot =
```

```
"GeoffBot" is name
0 Years Old
100 percent charge
```

Bot

```
Bot("GeoffBot")
```

```
"GeoffBot" is name

 0 Years Old

100 percent charge
```

Bot

name

"GeoffBot"

string

age

0

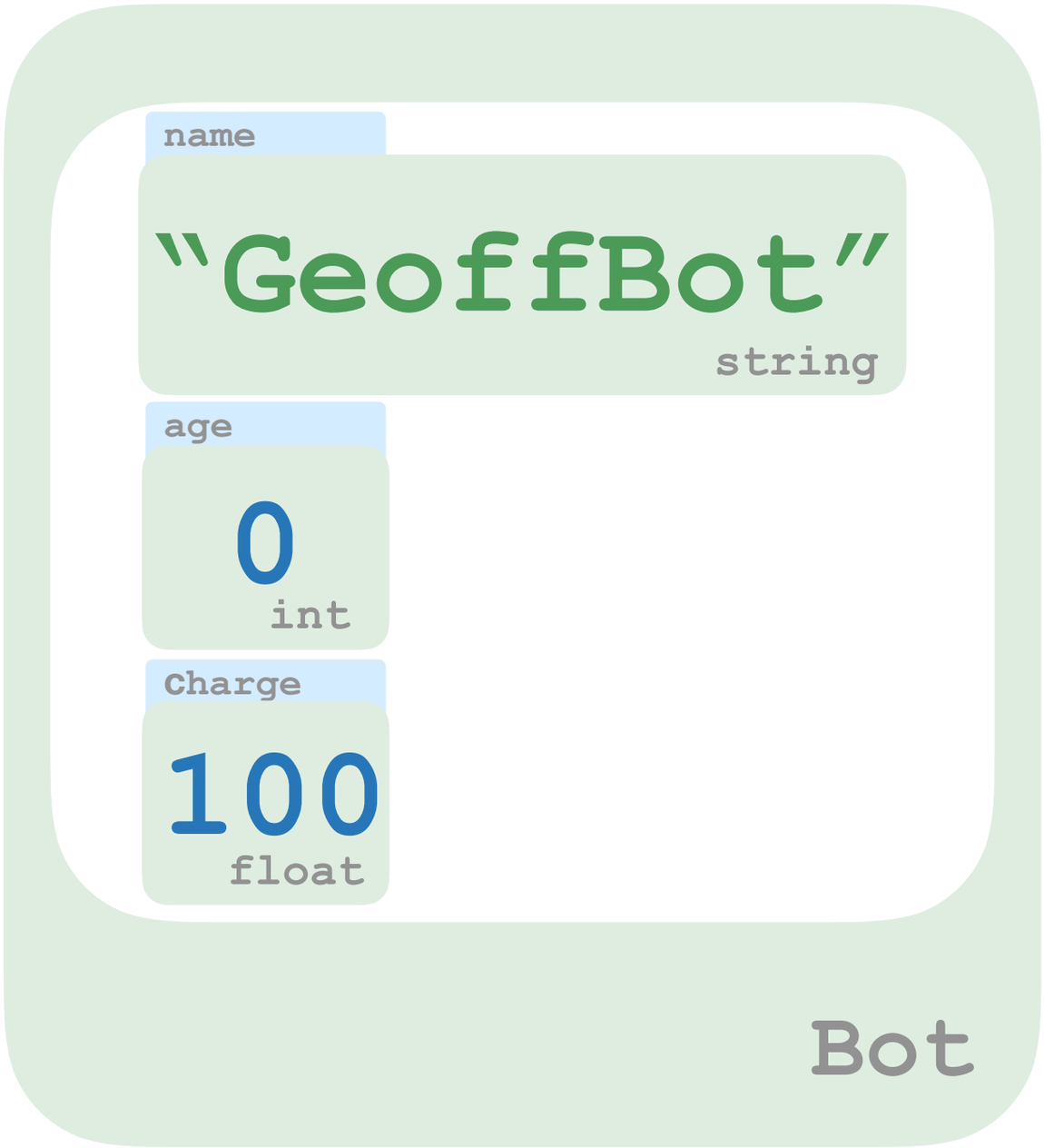
int

Charge

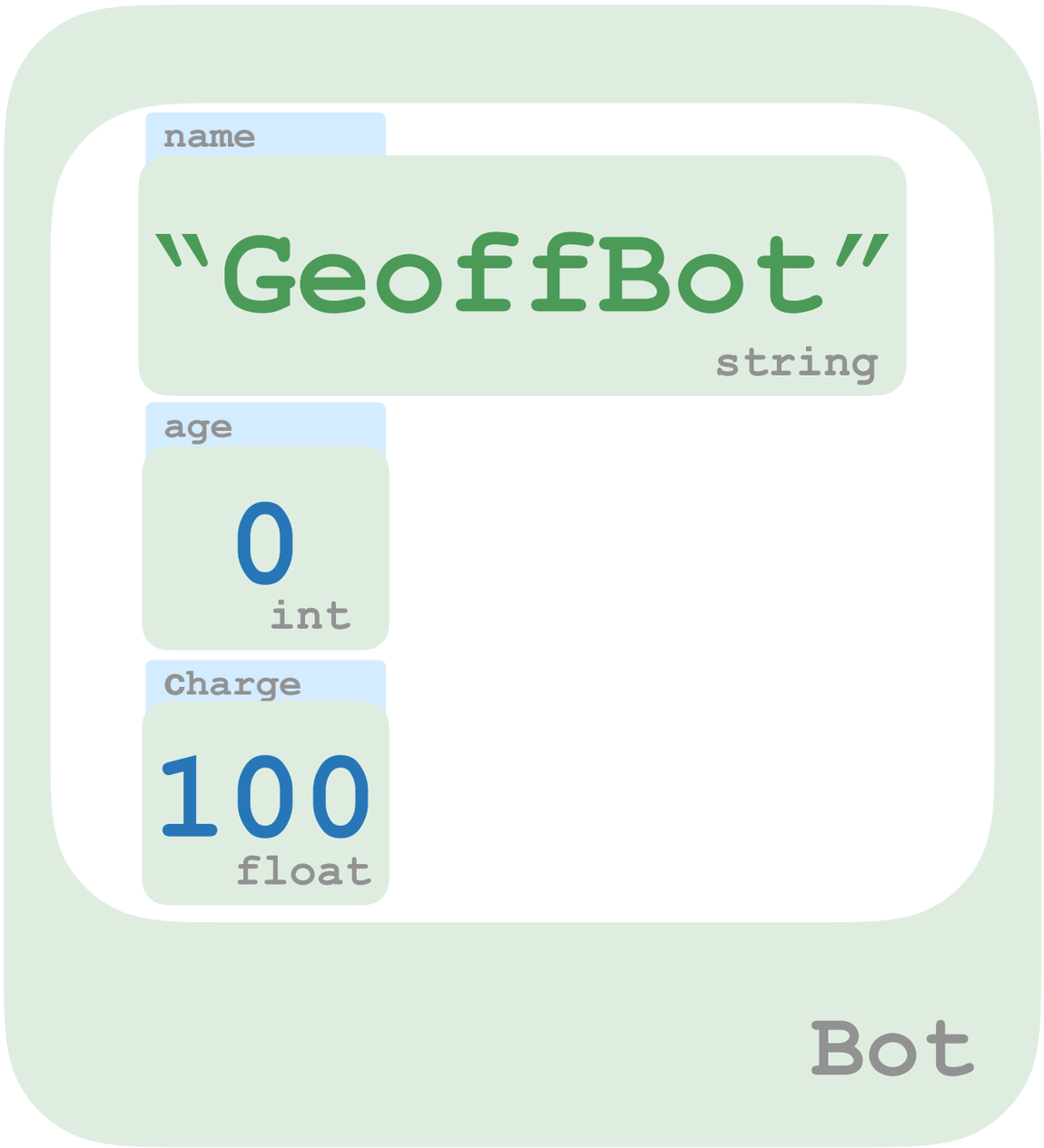
100

float

Bot

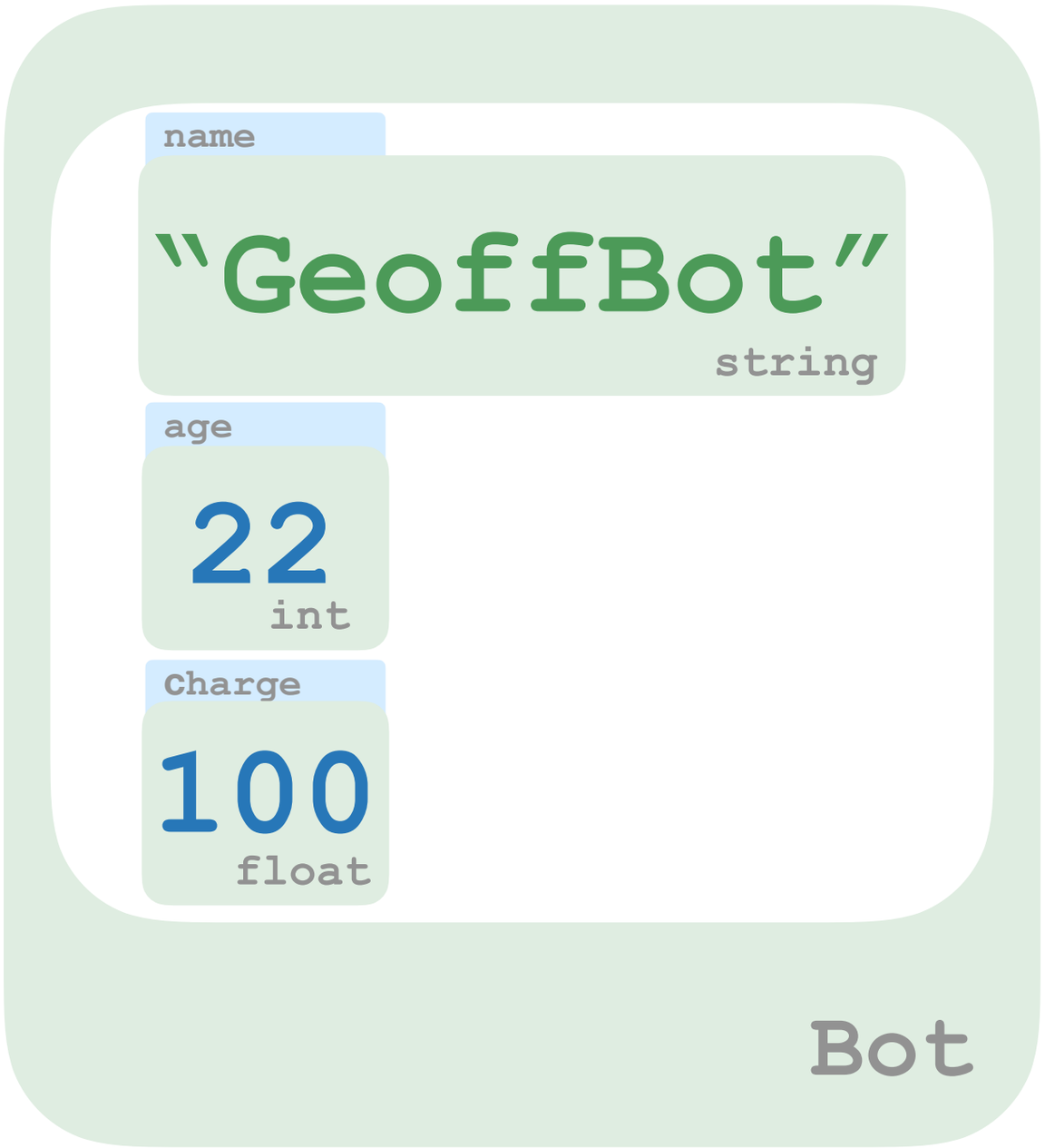



```
geoff_bot = Bot("GeoffBot")
```




```
geoff_bot = Bot("GeoffBot")
```

```
geoff_bot.age = 22
```
int




```
geoff_bot = Bot("GeoffBot")
```

```
geoff_bot.age = 22
```
int

name

"GeoffBot"

string

age

22

int

Charge

100

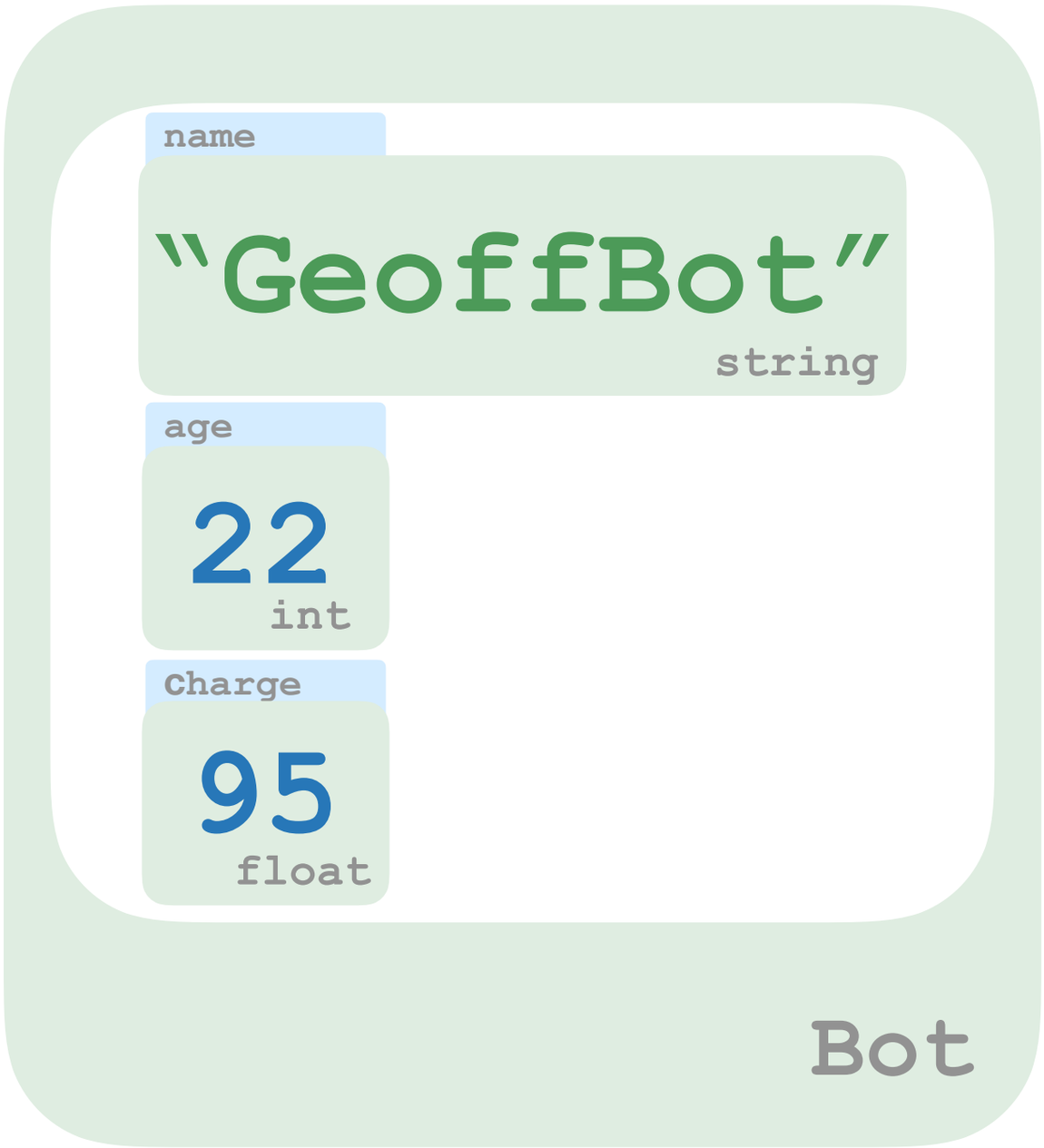
float

Bot

```
geoff_bot = Bot("GeoffBot")
```

```
geoff_bot.age = 22
```
int

```
geoff_bot.charge = 95
```
float




```
geoff_bot = Bot("GeoffBot")
```

```
geoff_bot.age = 22
```
int

```
geoff_bot.charge = 95
```
float

Introducing the `BankAccount` class

```
account_1 = BankAccount("Geoff")
```

```
account_1 =
```

```
0 Reais Remaining
Owner is "Geoff"
```

BankAccount

```
BankAccount("Geoff")
```

```
0 Reais Remaining

"Geoff" is owner
```

BankAccount

0 int Reais Remaining

"Geoff" string is owner

BankAccount

balance

0
int

Reais Remaining

name

"Geoff"
string

is owner

BankAccount

```
account_1 = BankAccount("Geoff")
```

balance

0
int

Reais Remaining

name

"Geoff"
string

is the owner

BankAccount

`account_1.balance` = `100` int

`balance`

`0` int

Reais Remaining

`name`

`"Geoff"` string

is the owner

`BankAccount`

```
account_1.balance = 100
                    int
```

```
balance
100
int
```

100 Reais Remaining

```
name
"Geoff"
string
```

is the owner

BankAccount

Objects

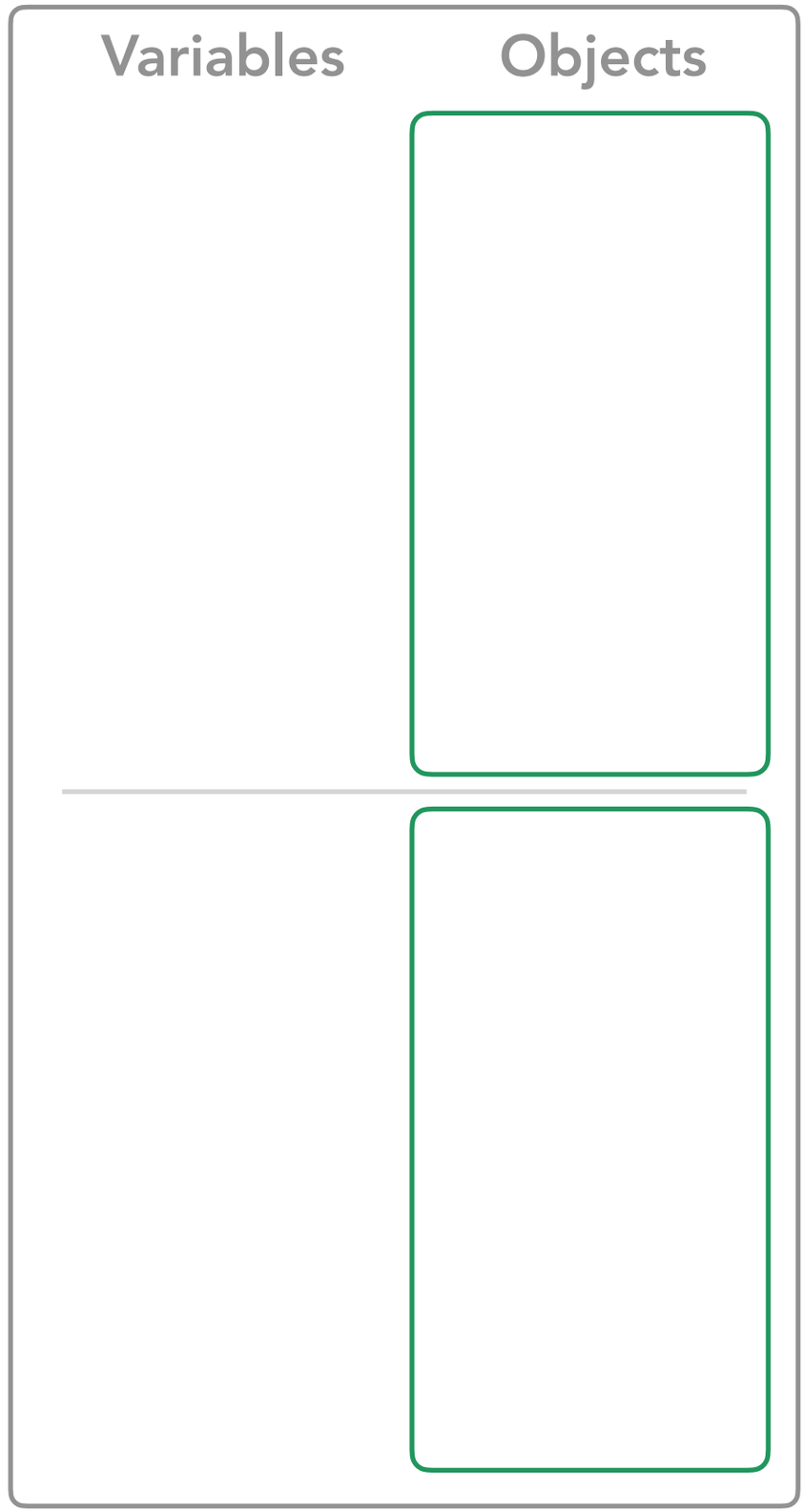
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

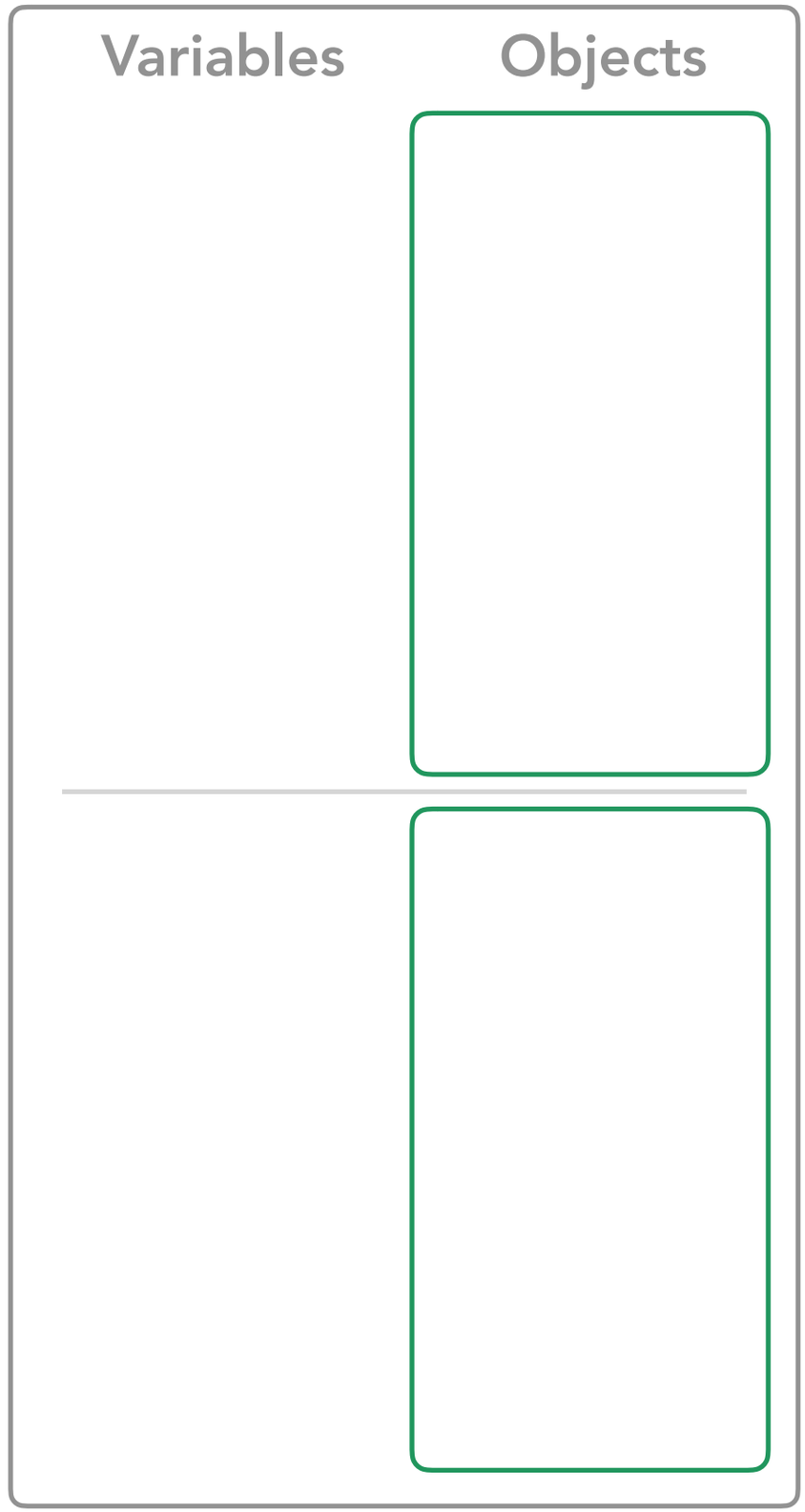
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory

Variables

Objects

Scope

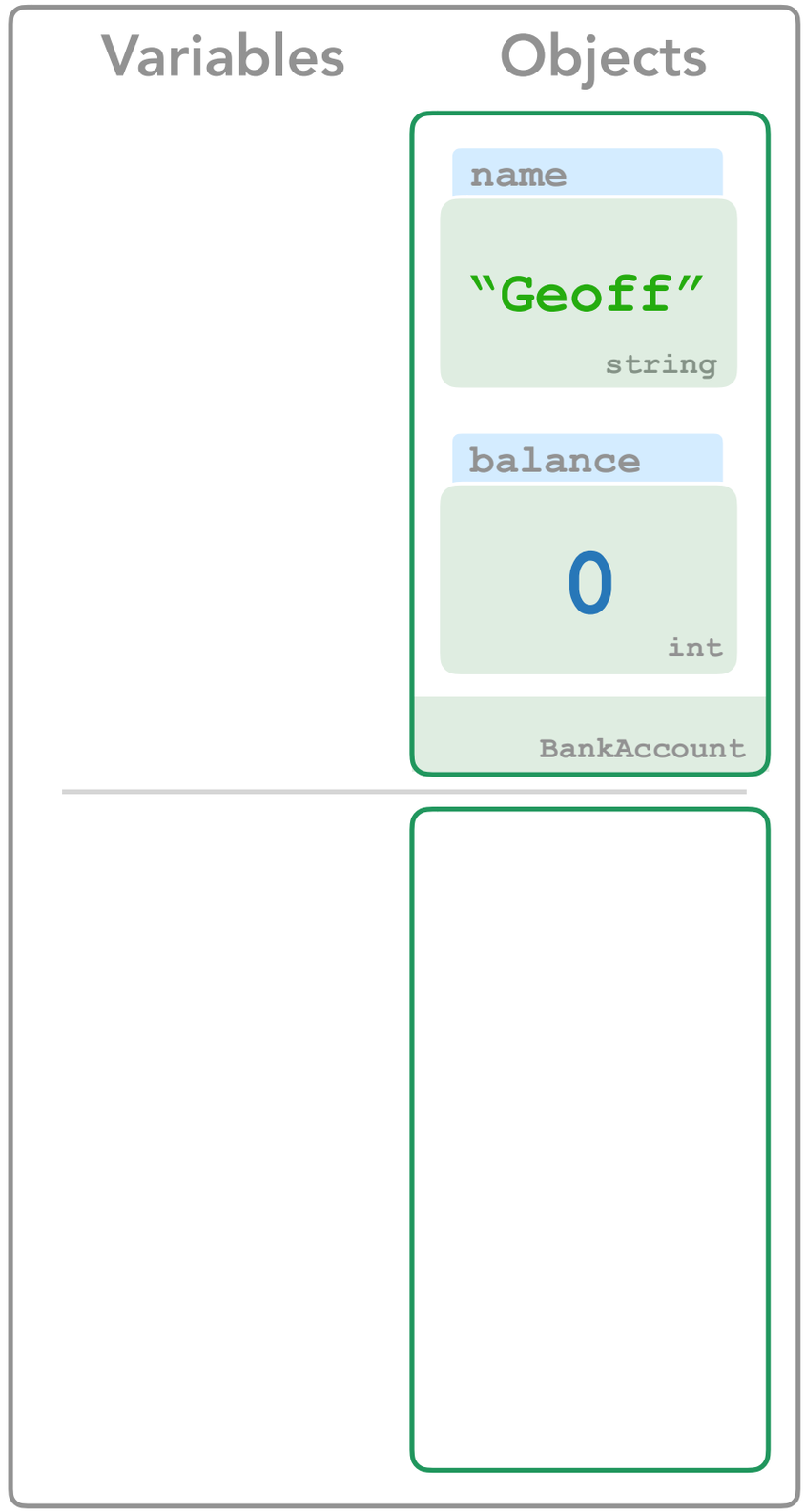
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

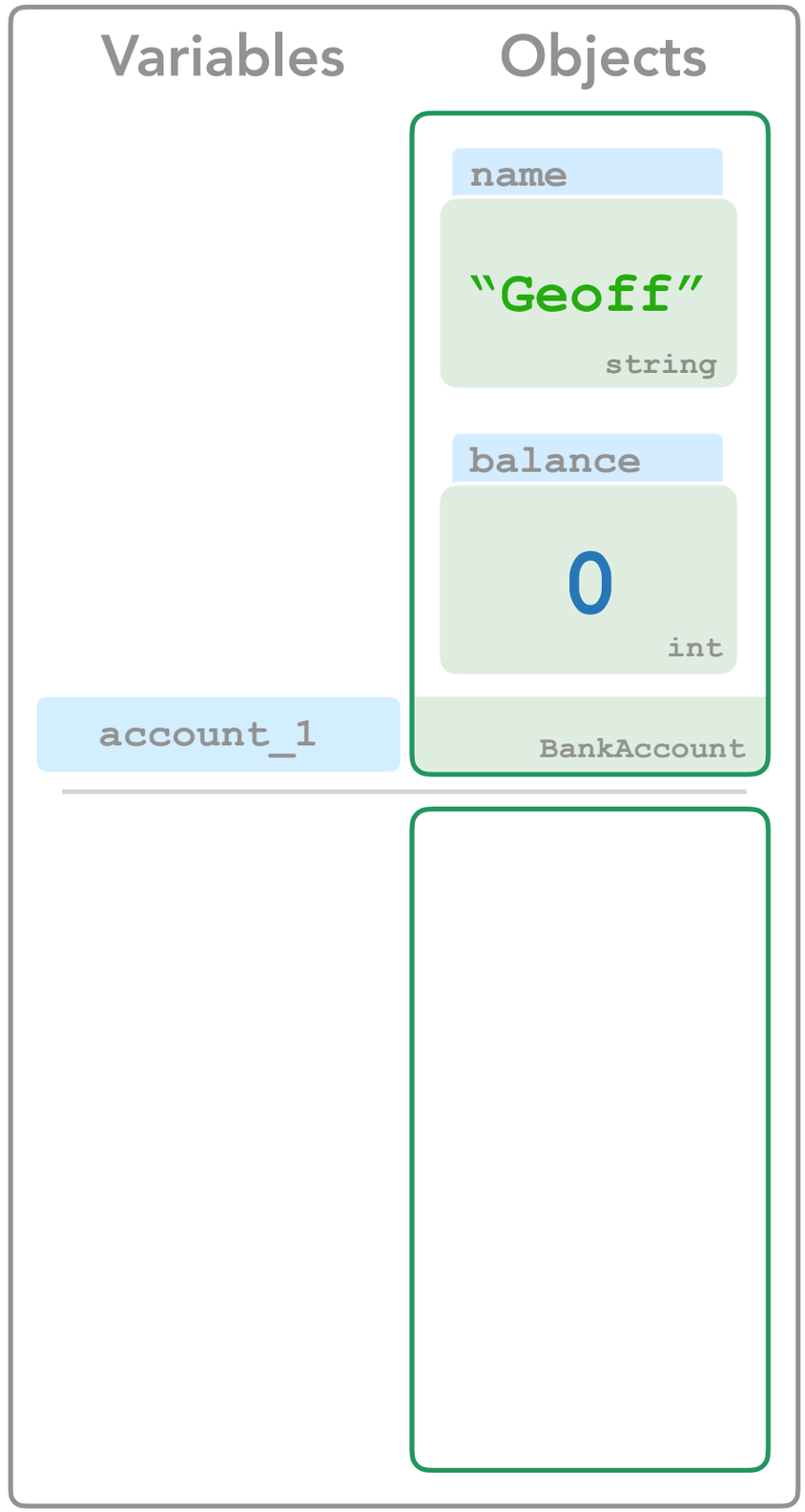
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

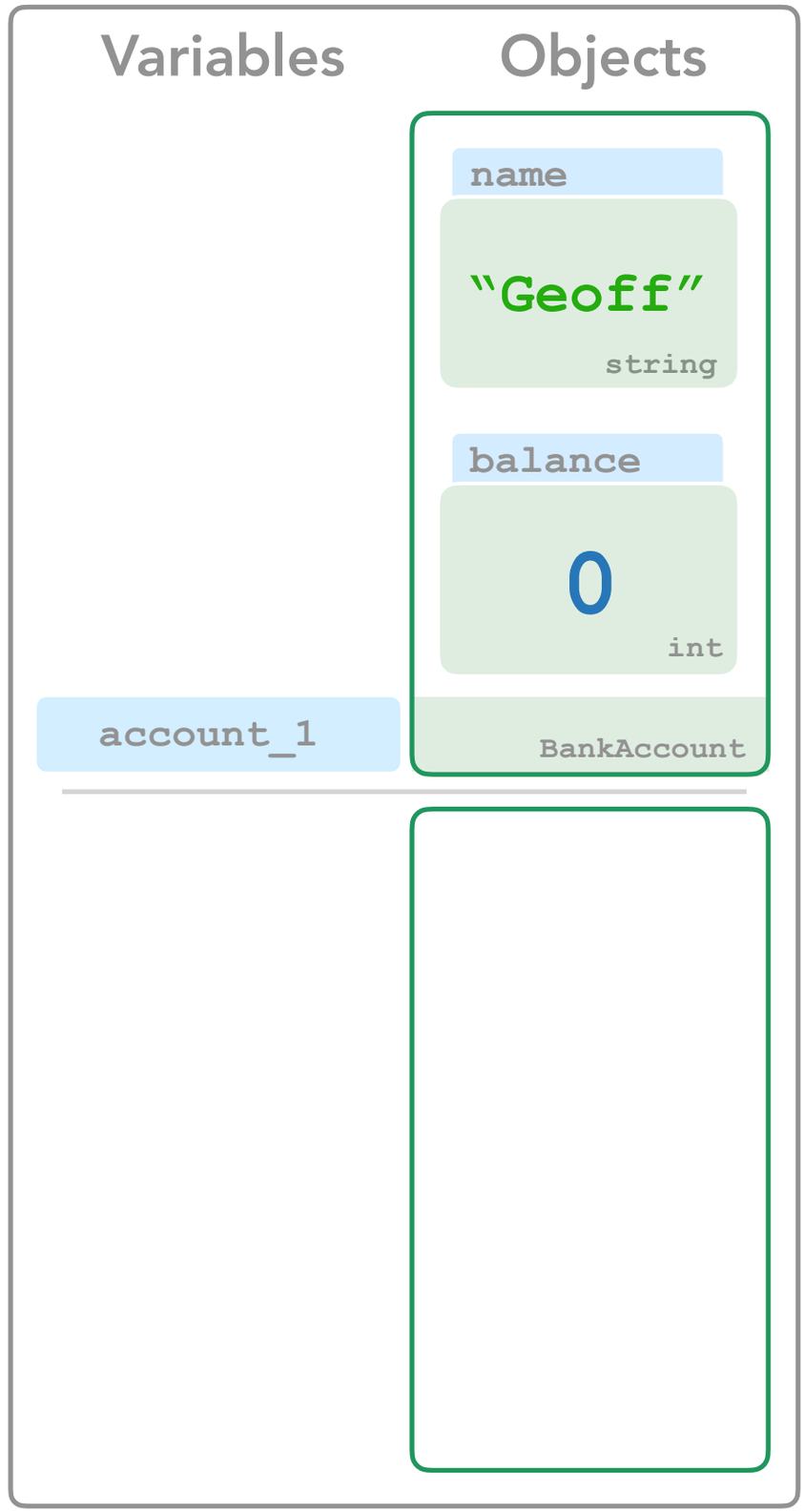
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

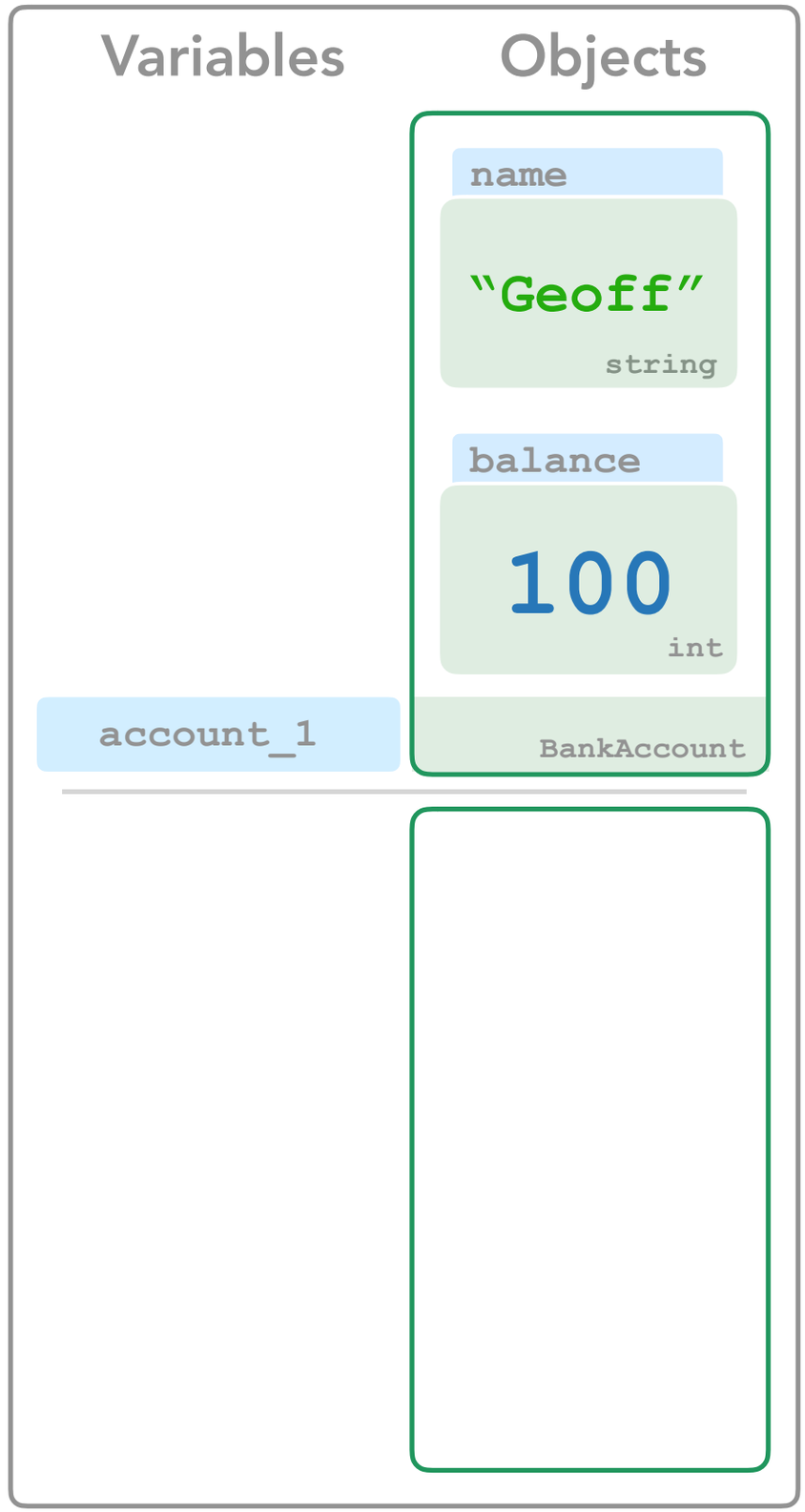
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

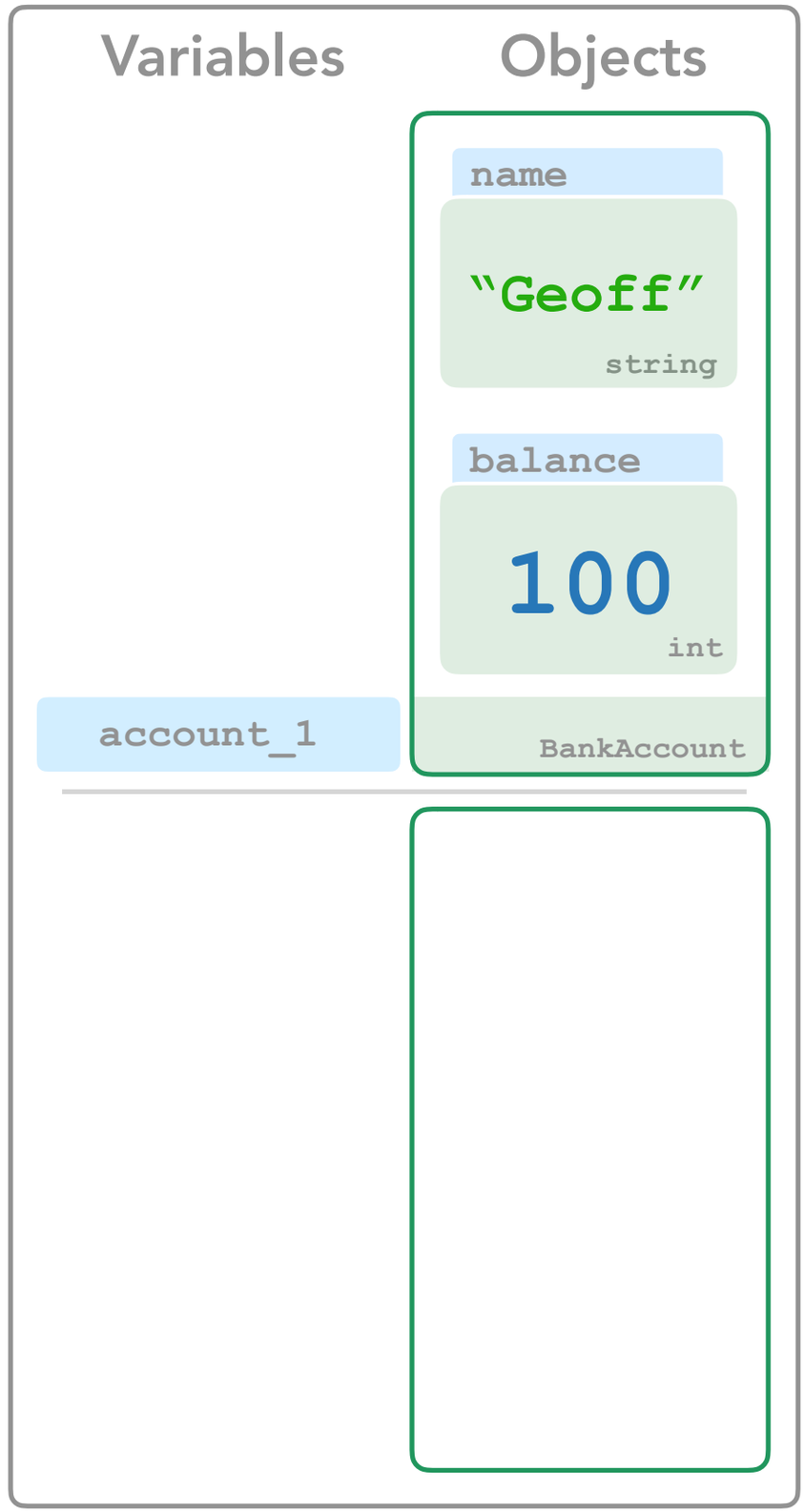
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

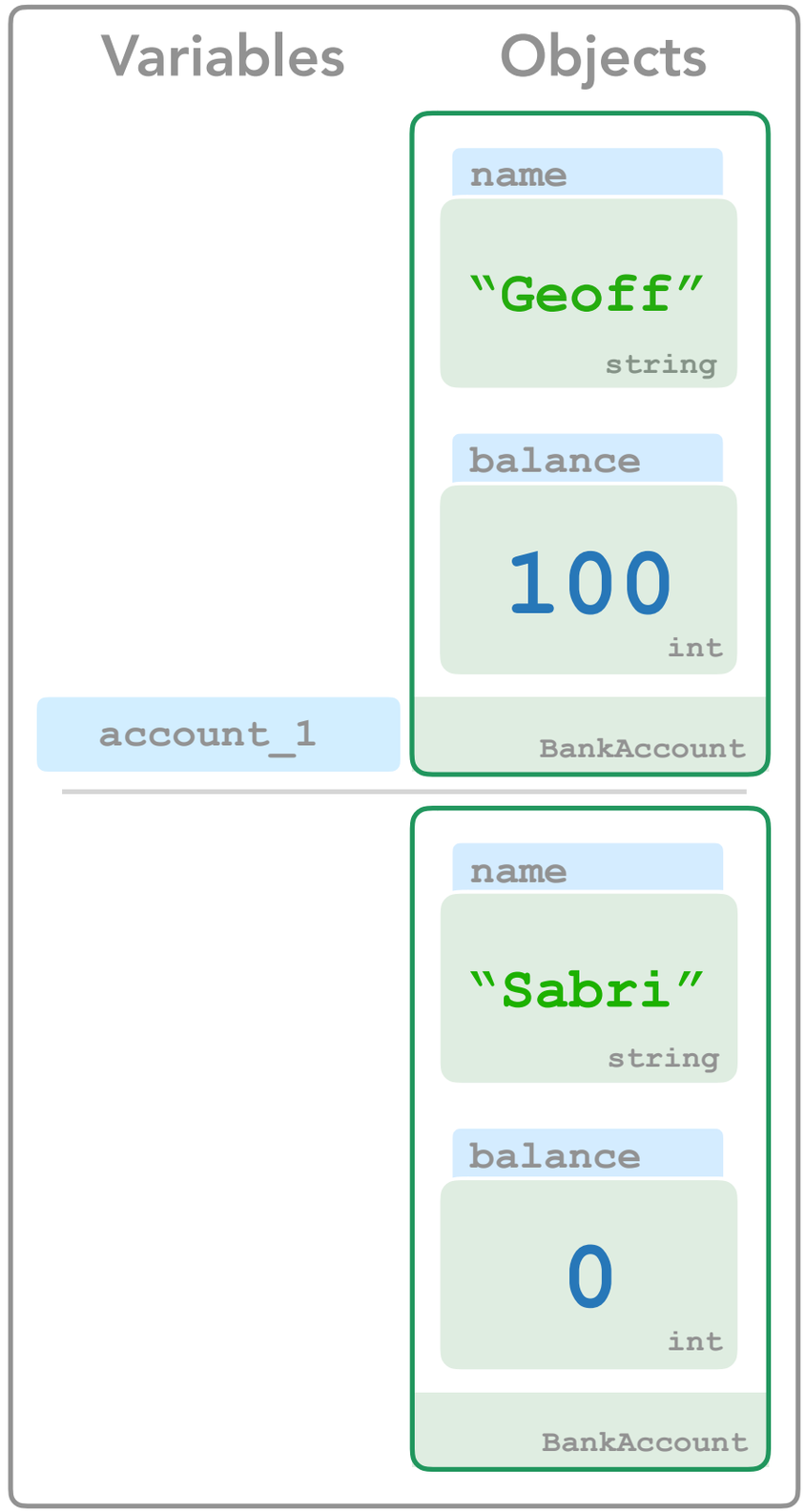
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

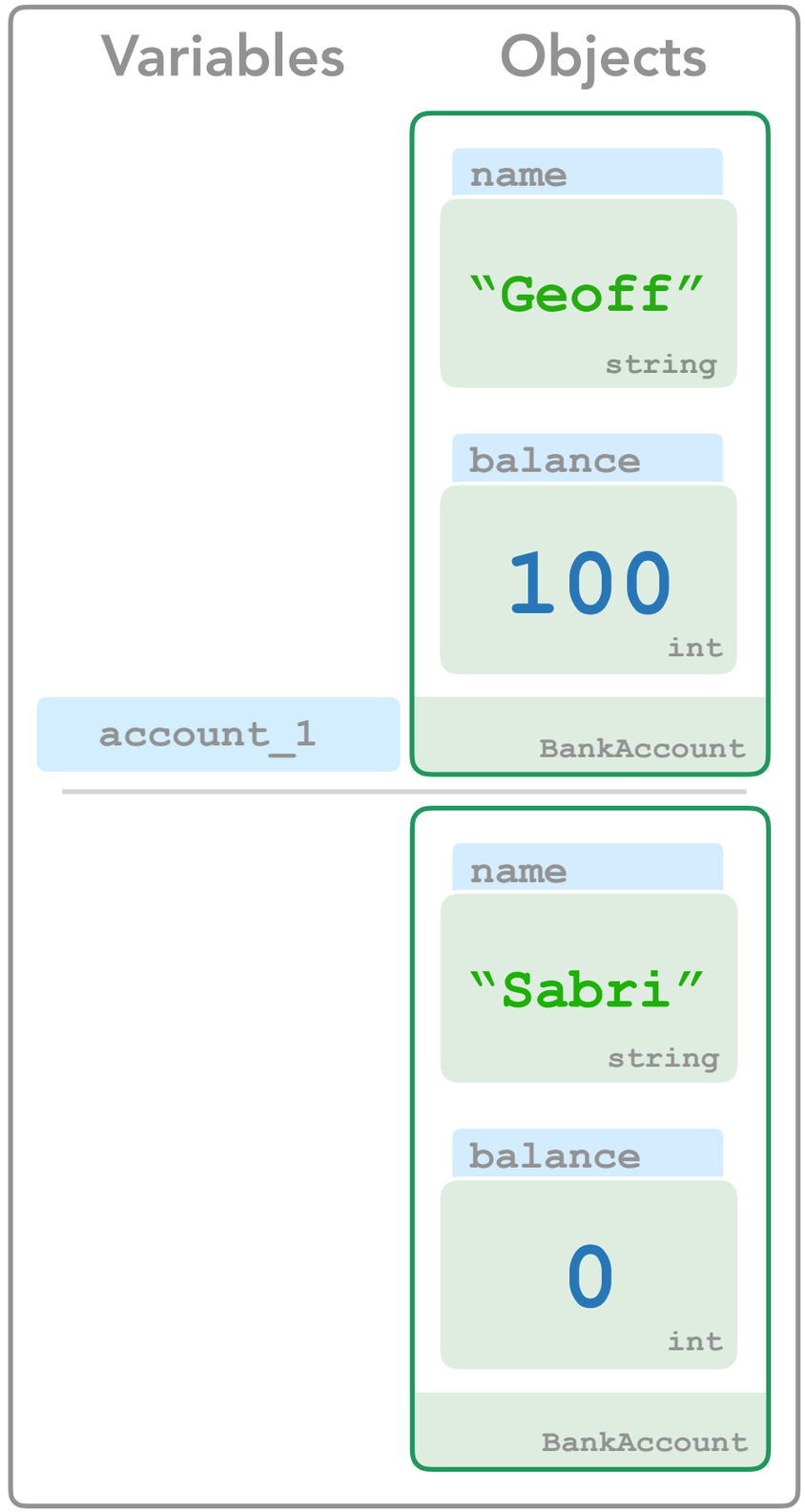
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

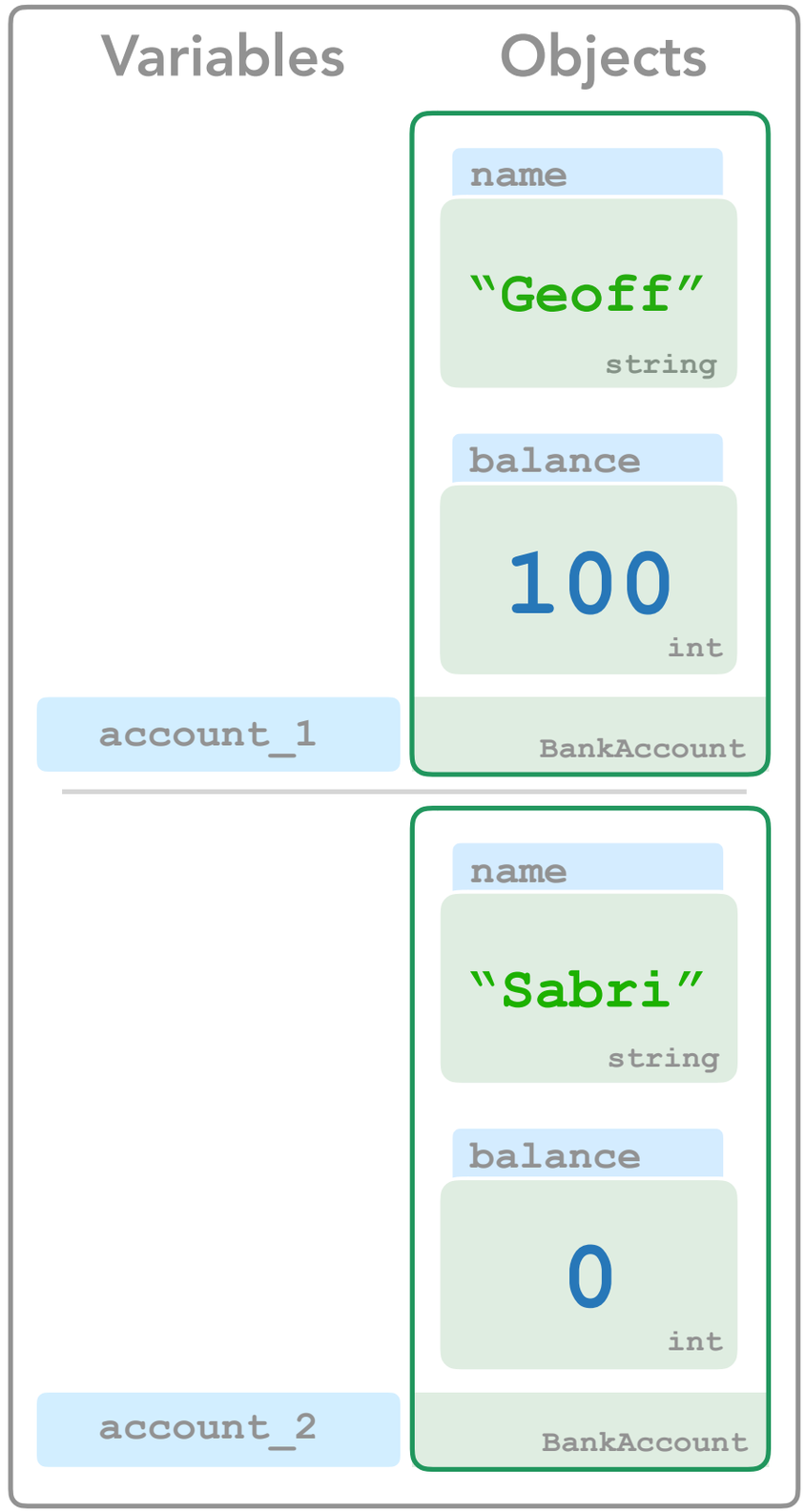
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

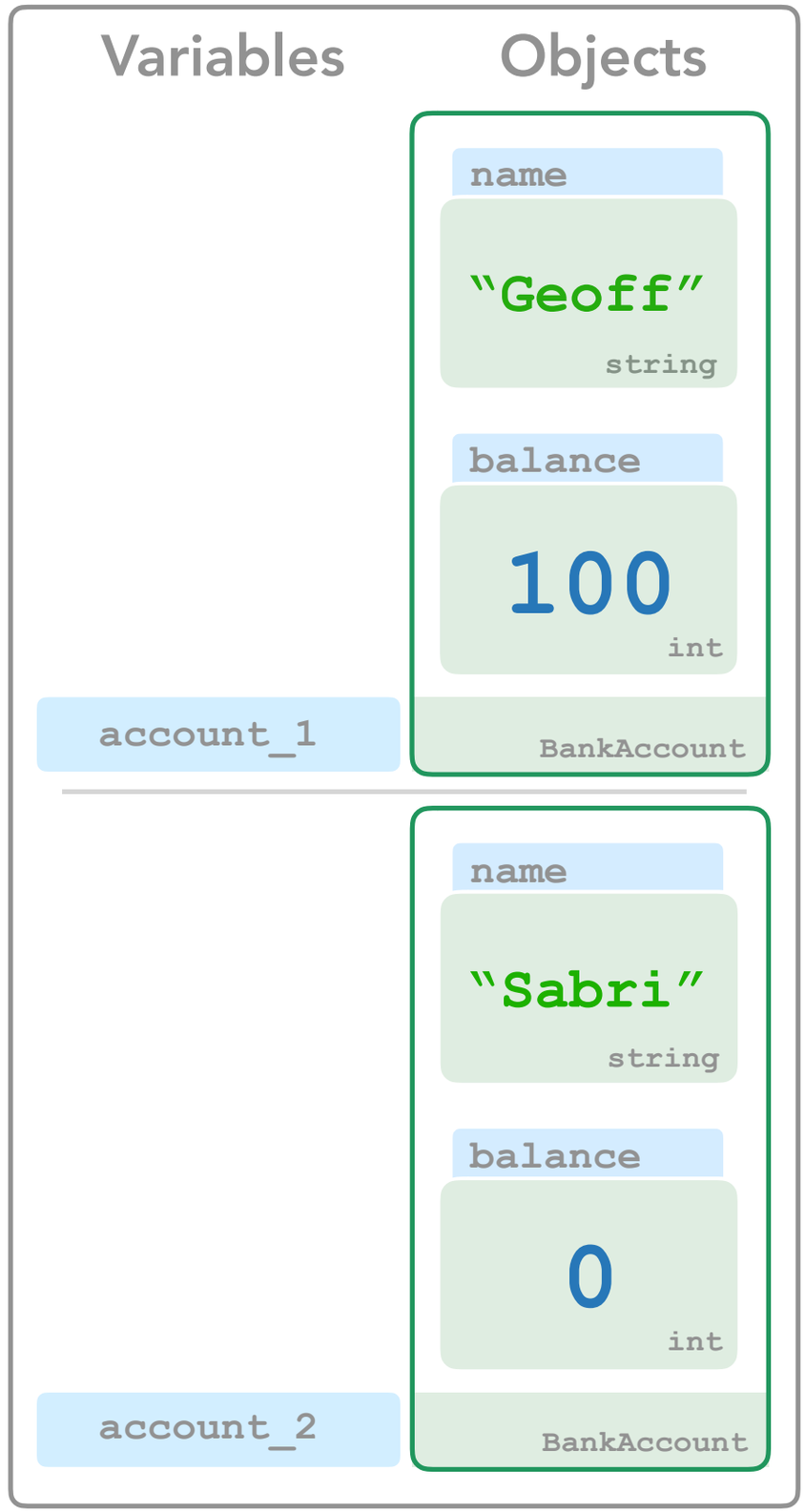
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

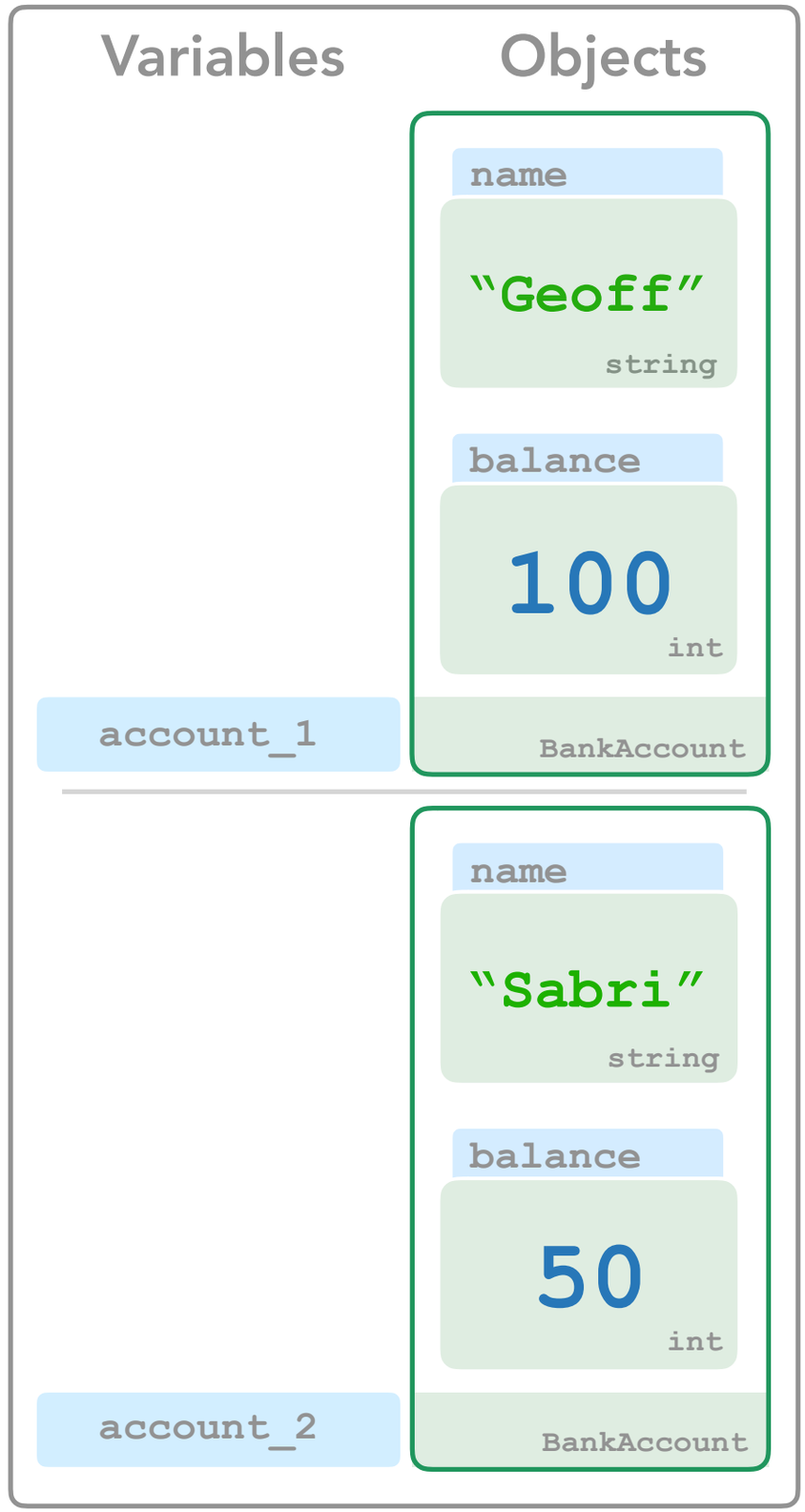
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

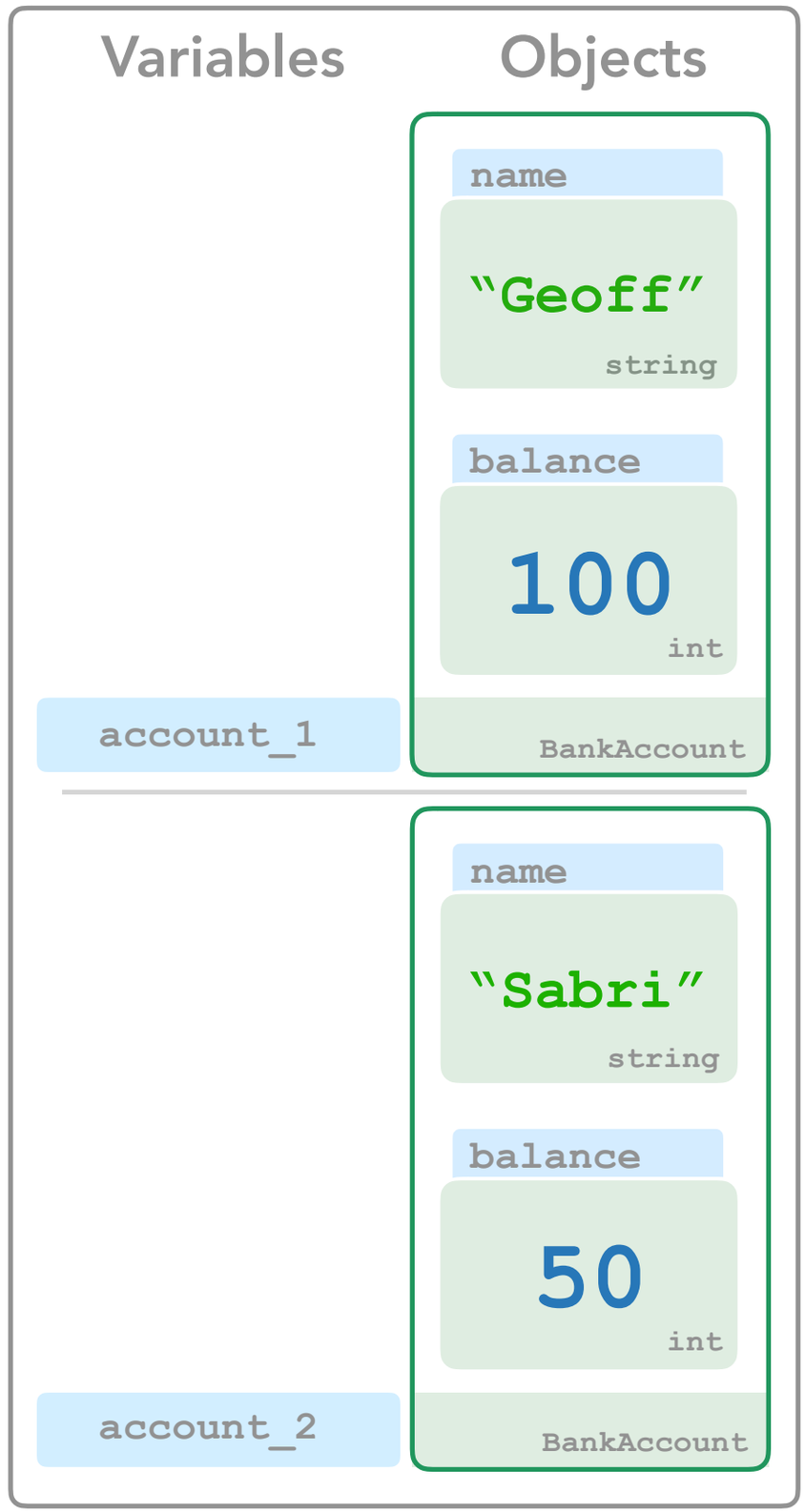
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Scope

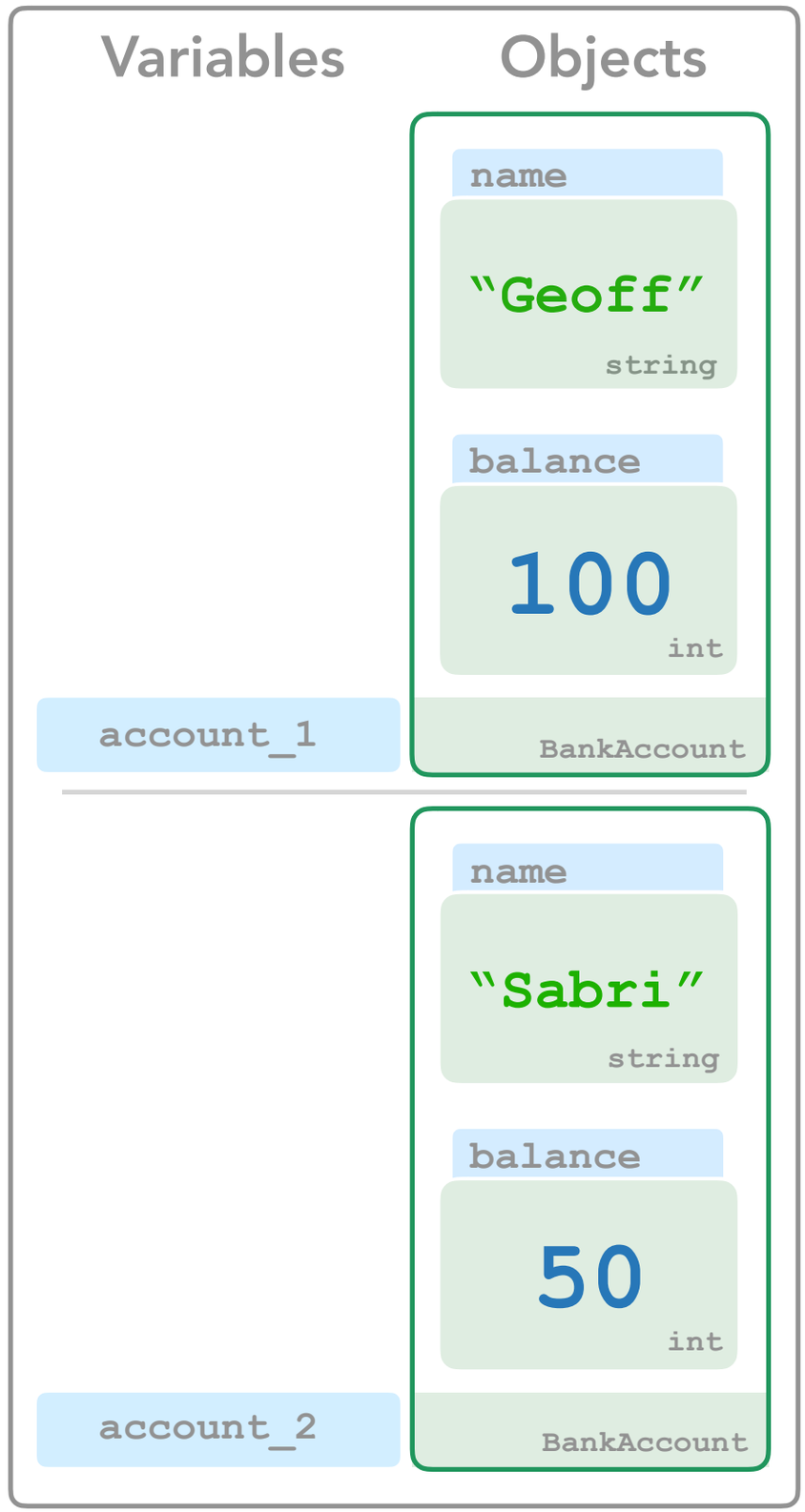
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
Geoff's account has R$100
```

Memory



Scope

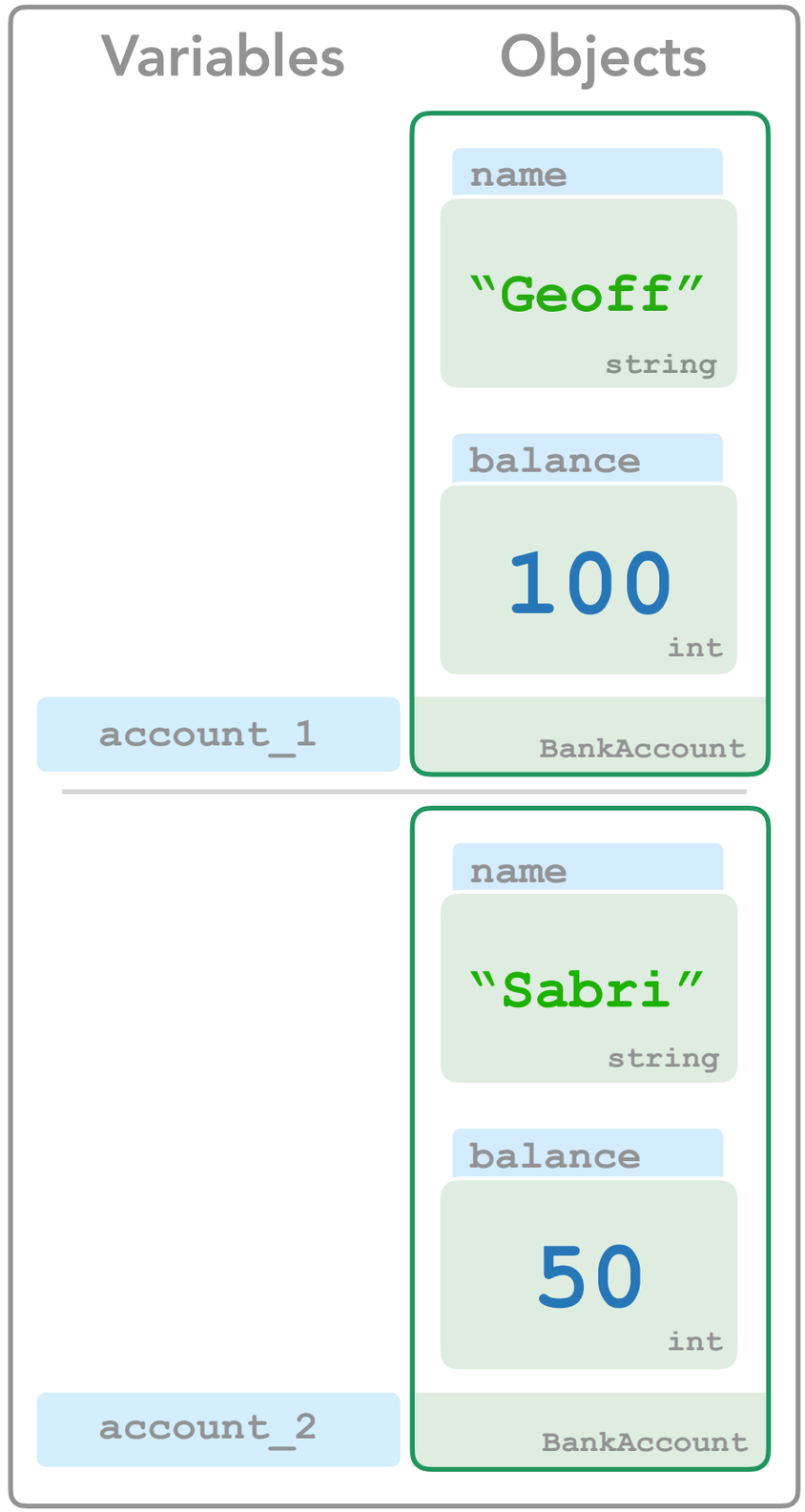
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
Geoff's account has R$100
```

Memory



Scope

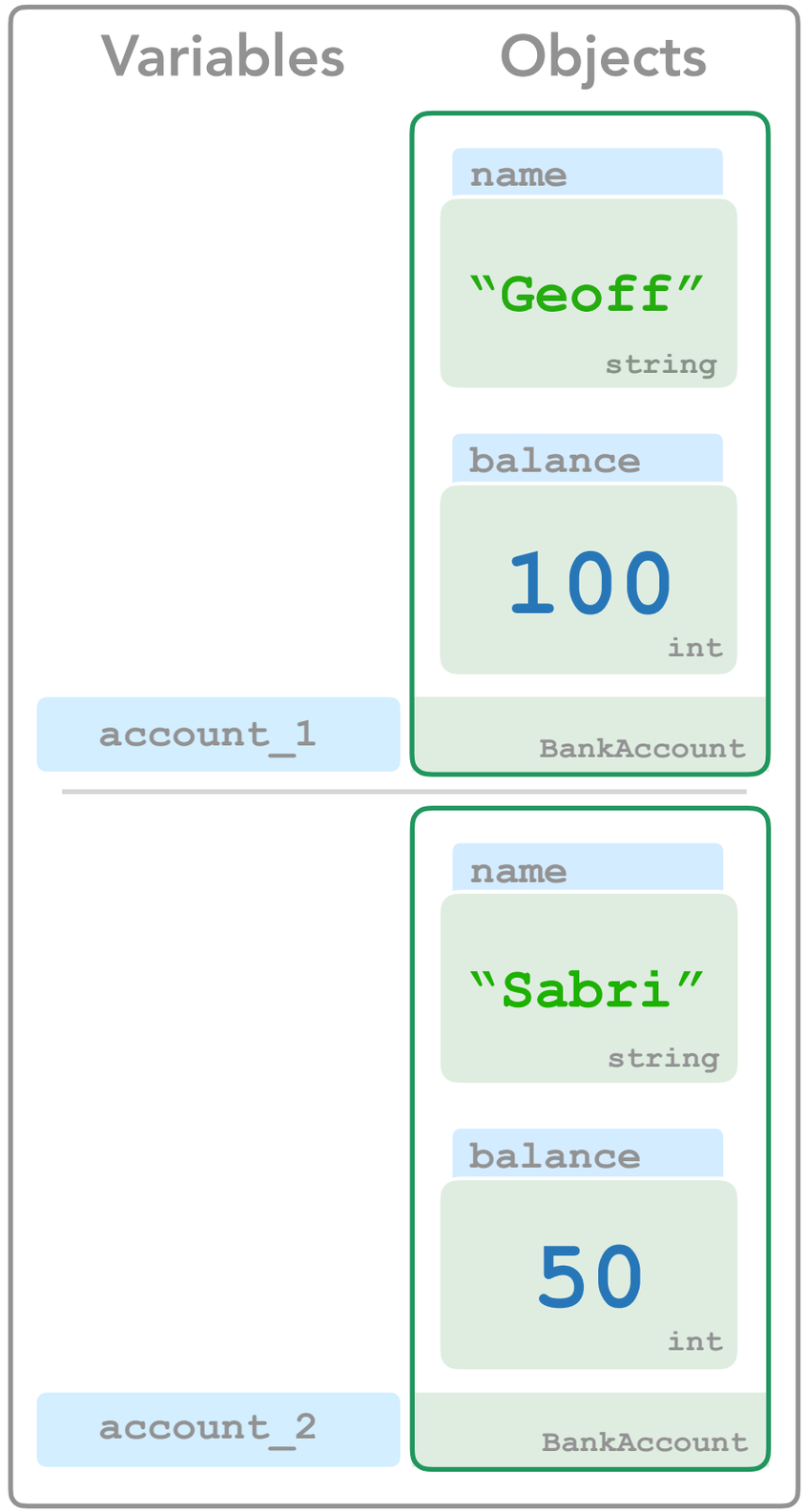
Code

```
from util import BankAccount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100

  account_2 = BankAccount("Sabri")
  account_2.balance = 50

  print("Geoff's account has R$" + str(account_1.balance))
  print("Sabri's account has R$" + str(account_2.balance))
```

Output

```
Geoff's account has R$100
Sabri's account has R$50
```

Memory



Complex Object **Syntax**

```
account_1 = BankAccount("Geoff")
```

Complex Object **Syntax**

"Constructor" Function

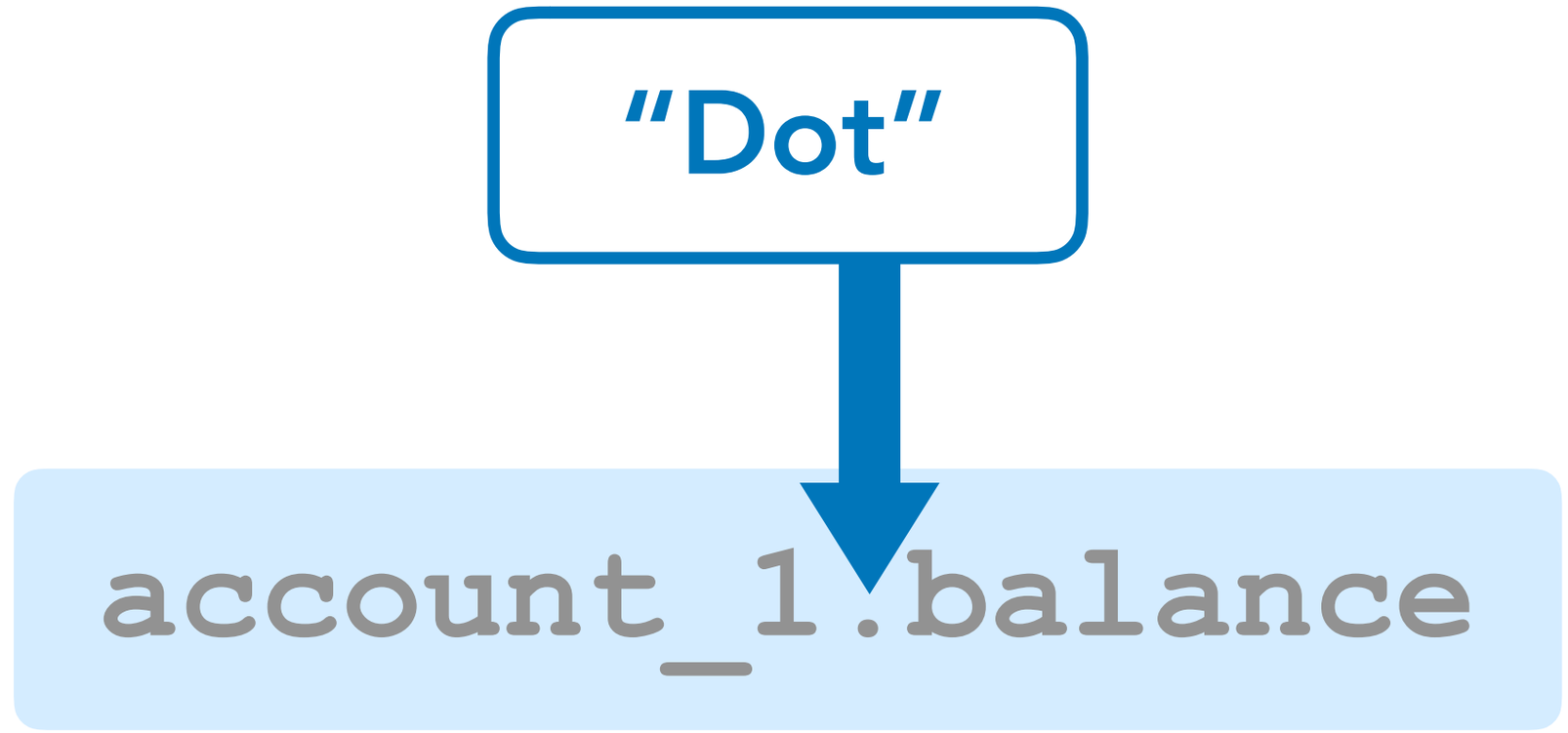
```
account_1 = BankAccount("Geoff")
```

Complex Object **Syntax**

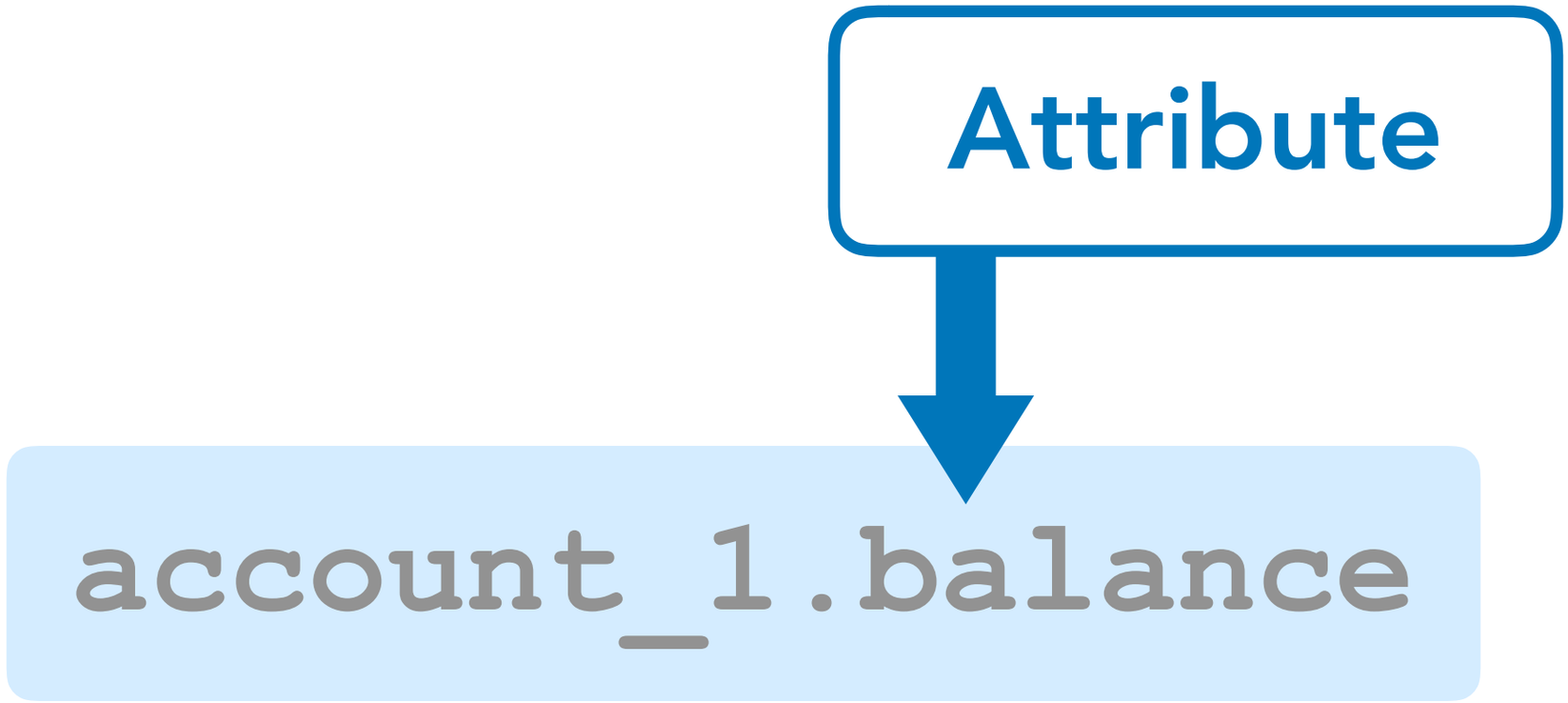
Parameters

```
account_1 = BankAccount("Geoff")
```

Complex Object **Syntax**



Complex Object **Syntax**



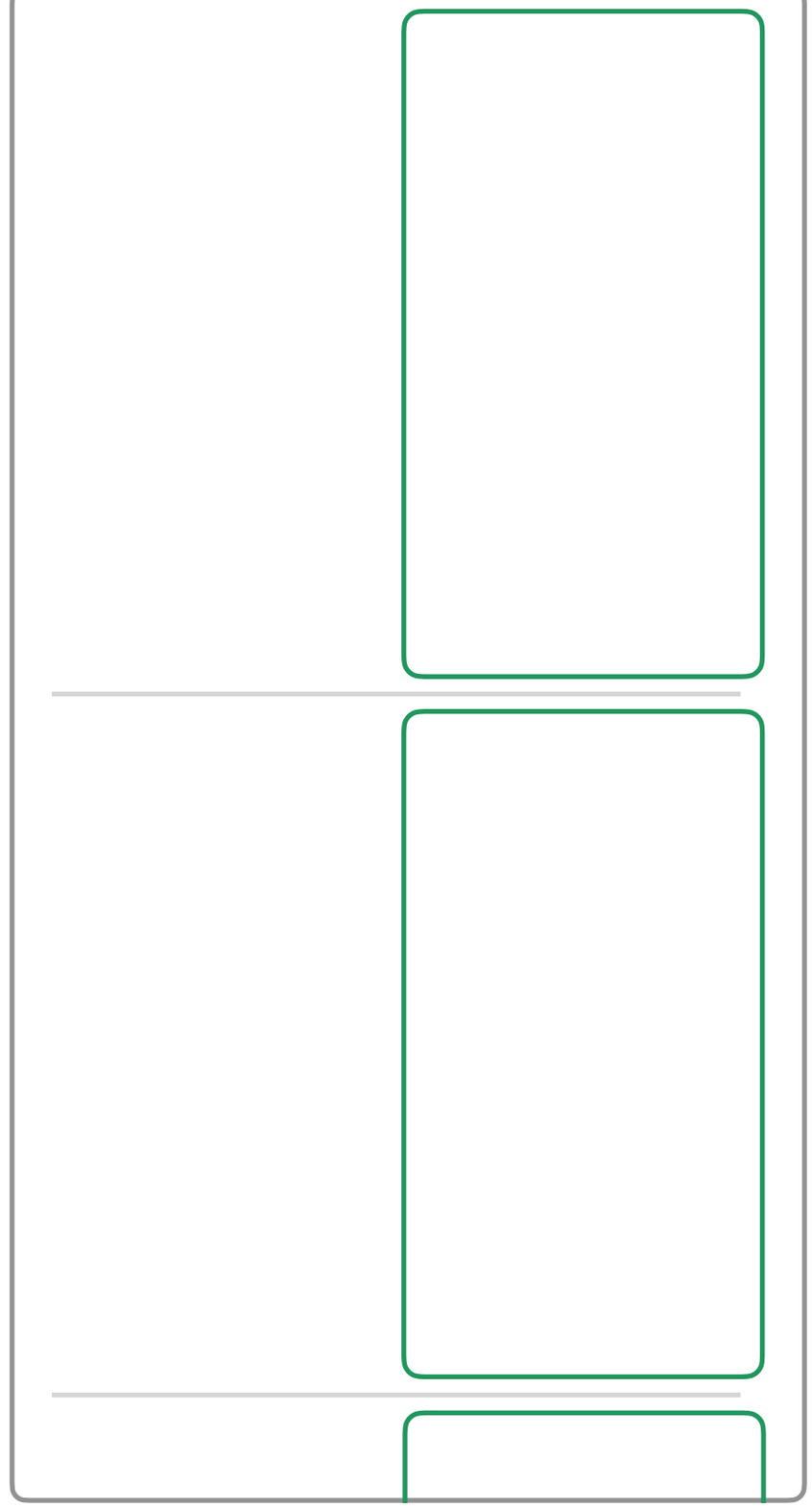
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



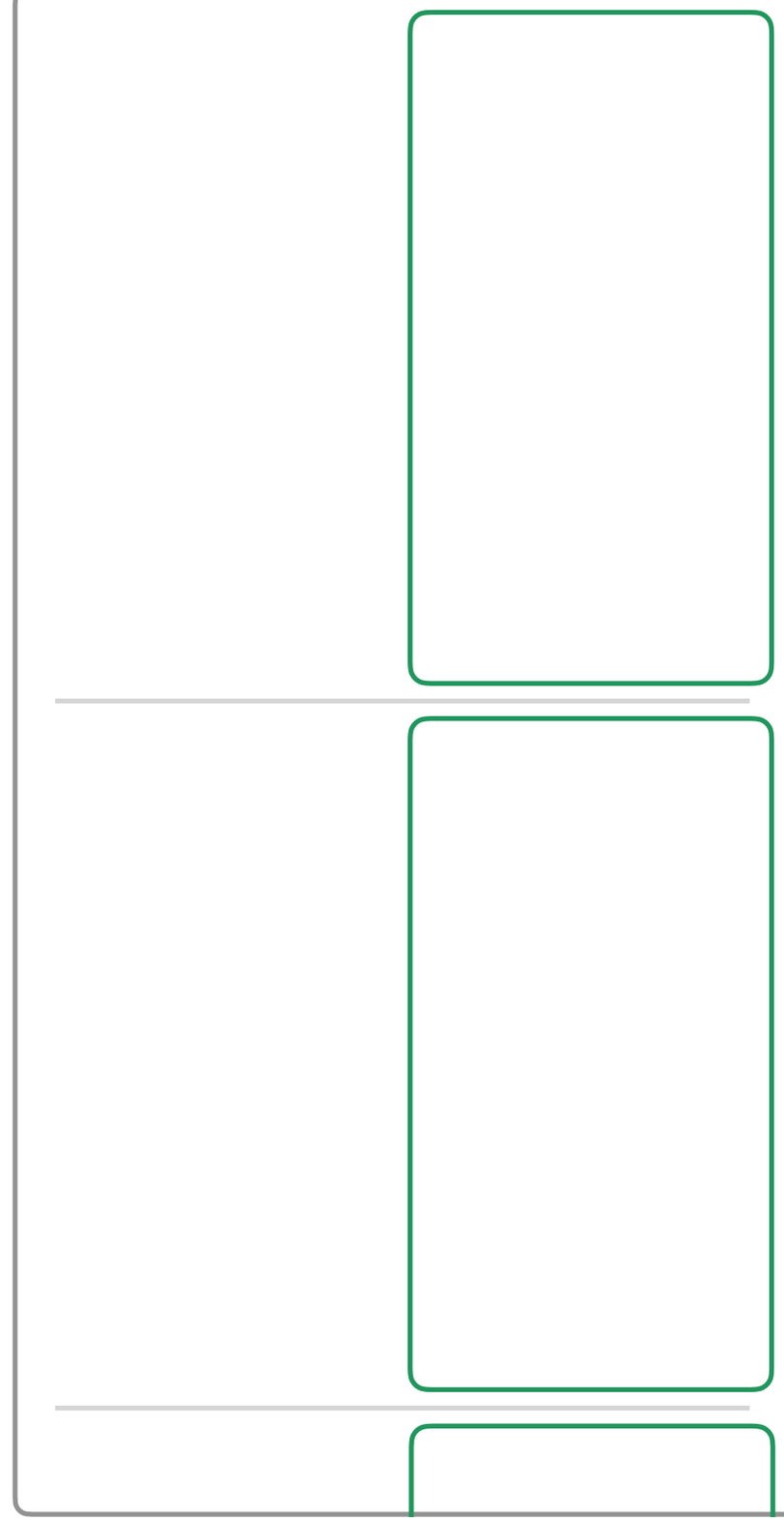
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



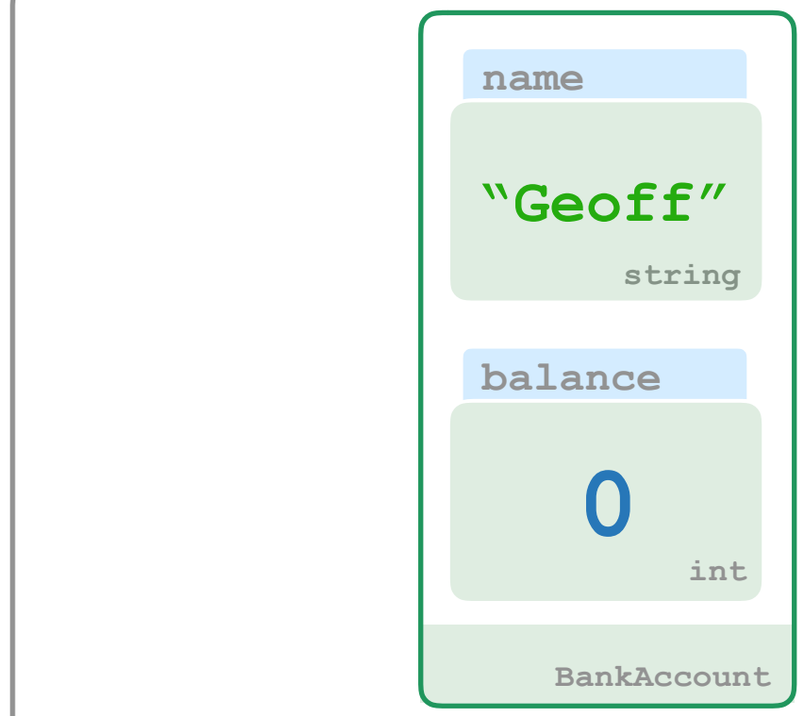
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
>
```

Memory

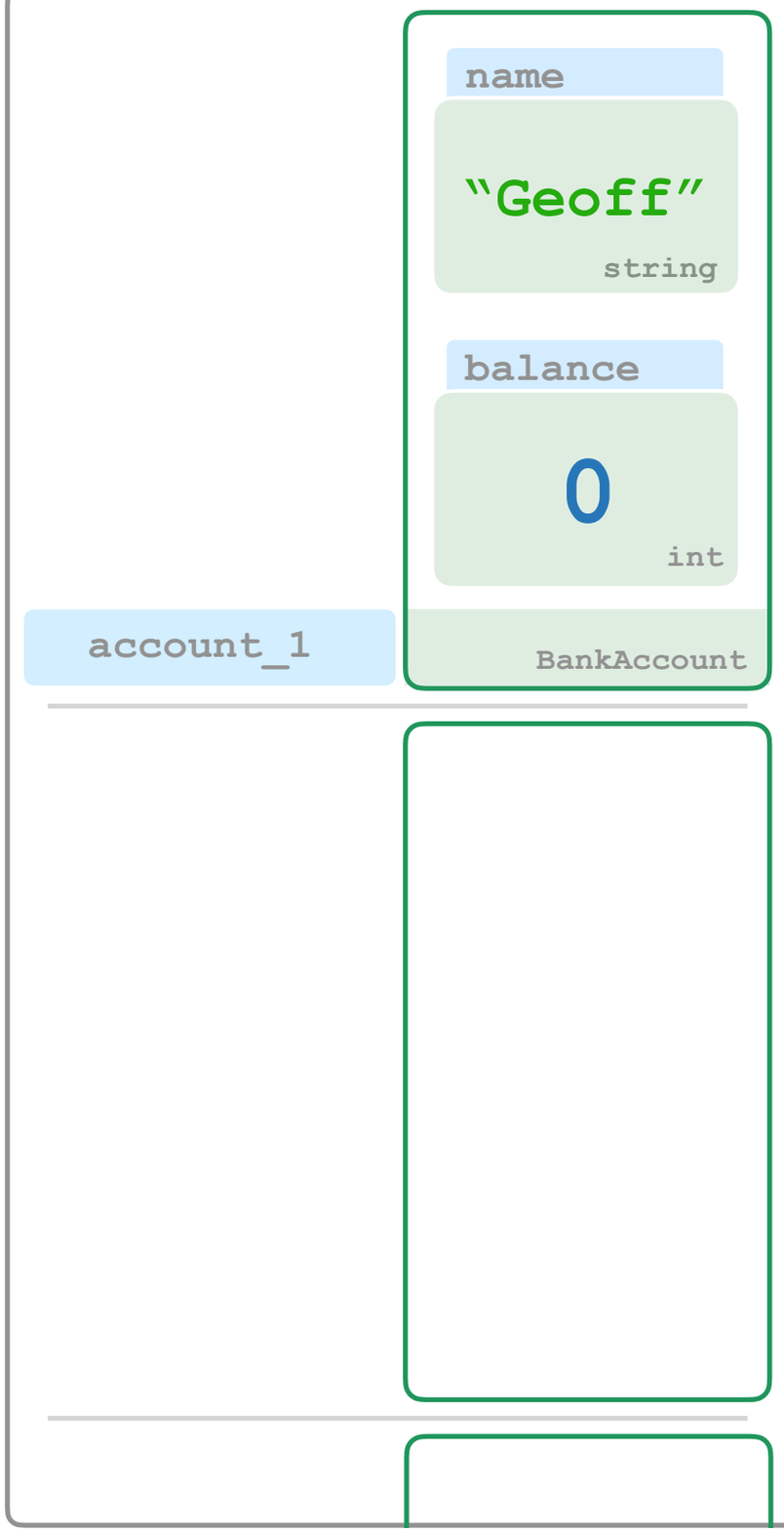
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



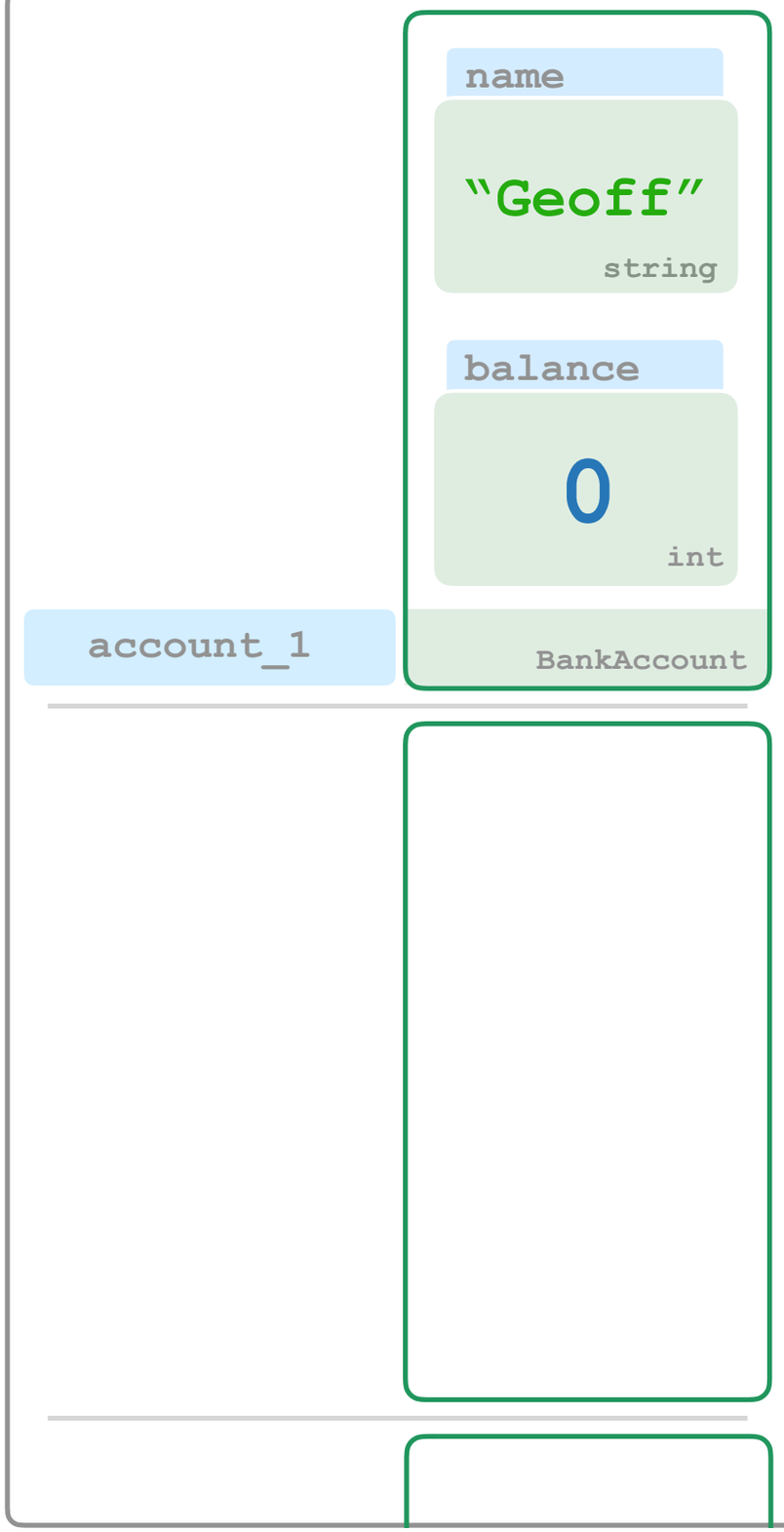
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



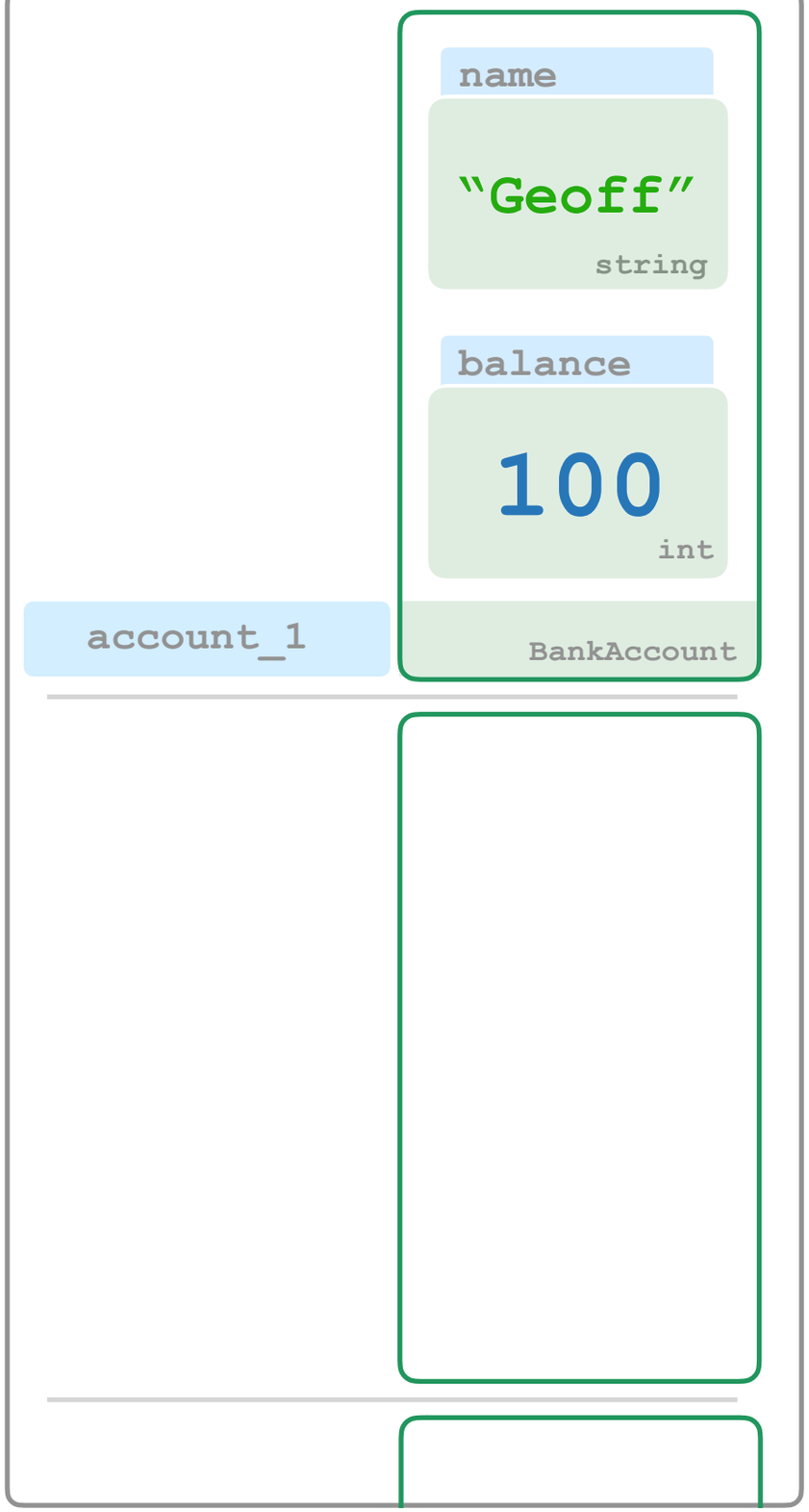
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



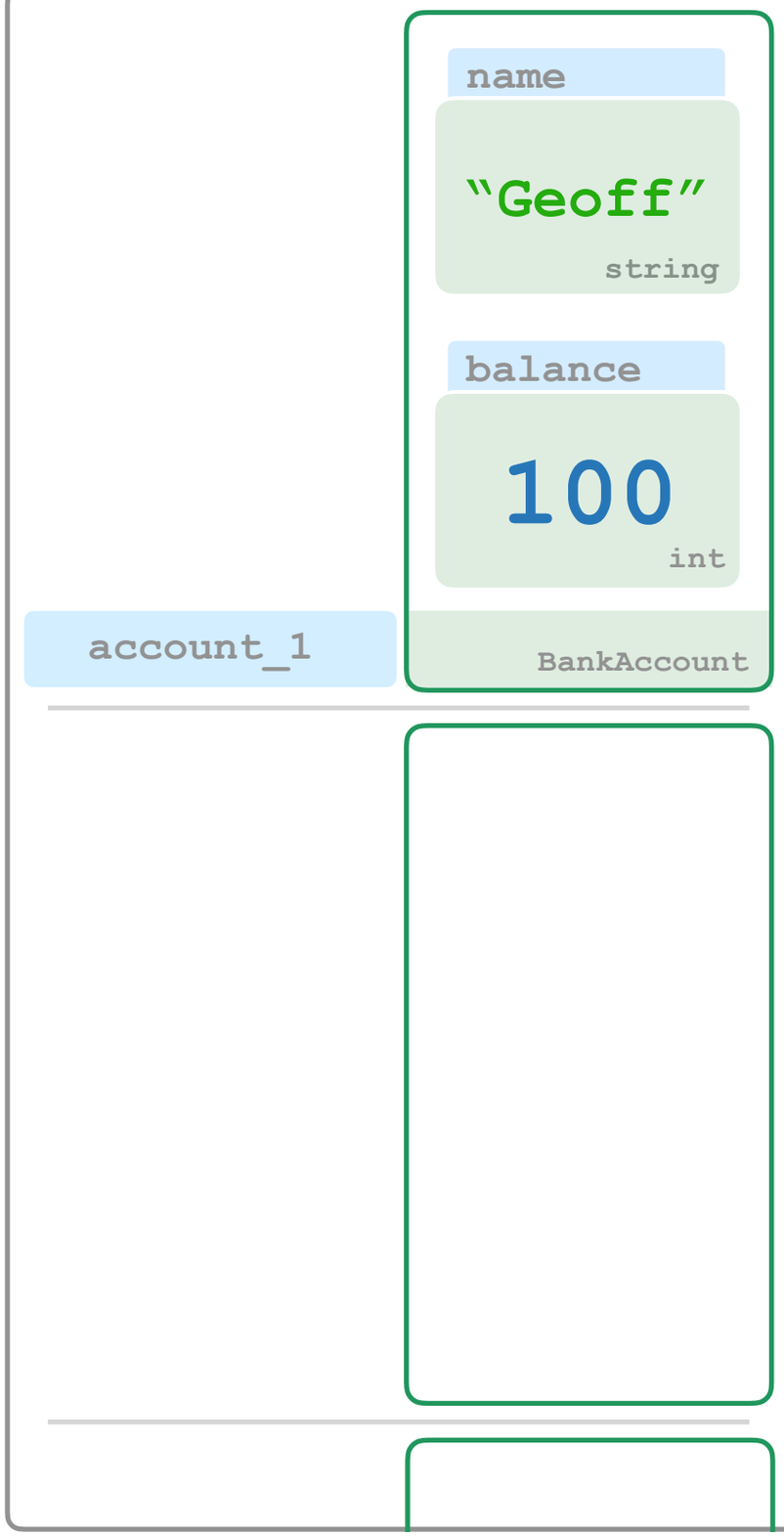
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



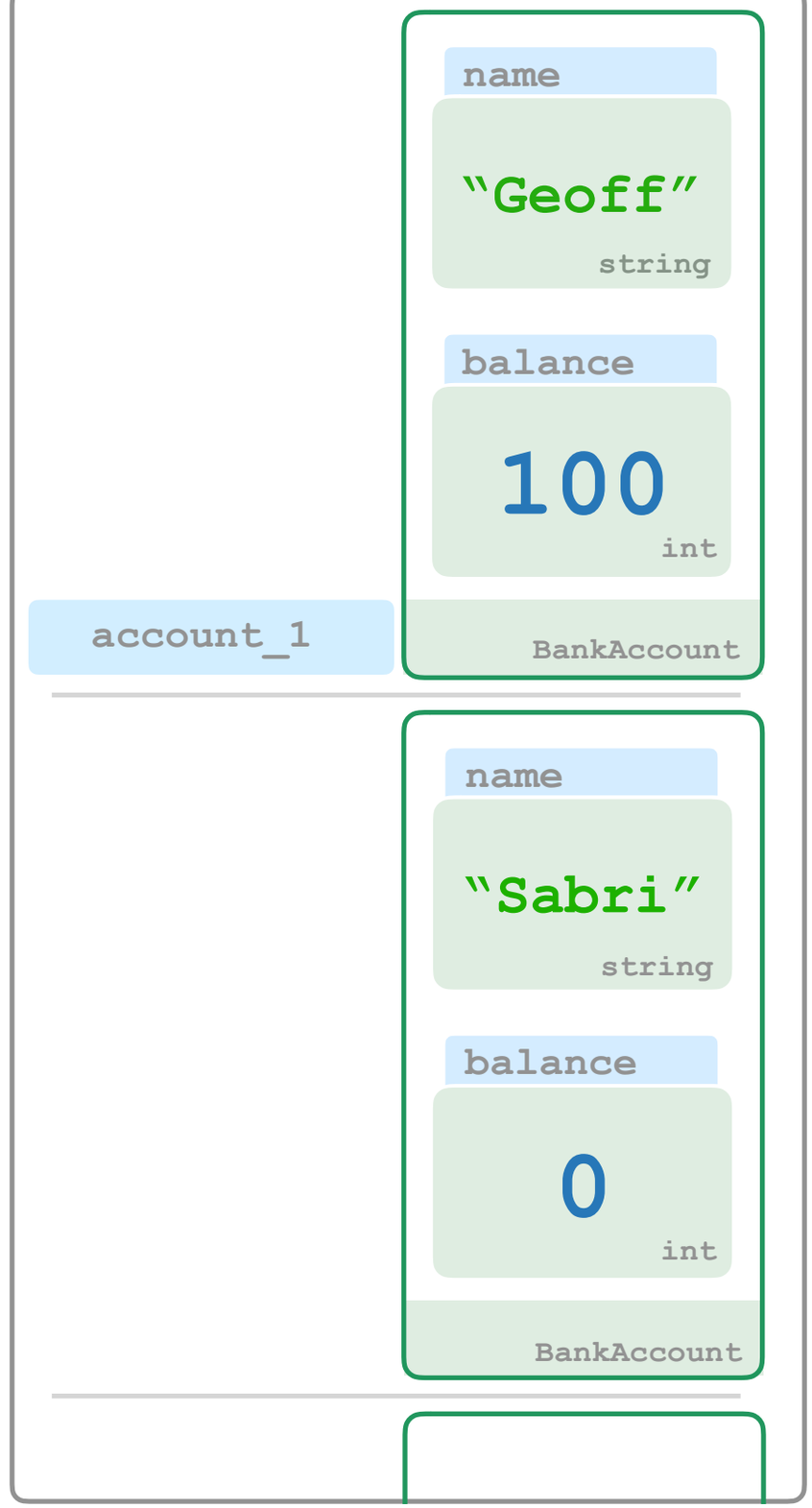
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
```

Memory



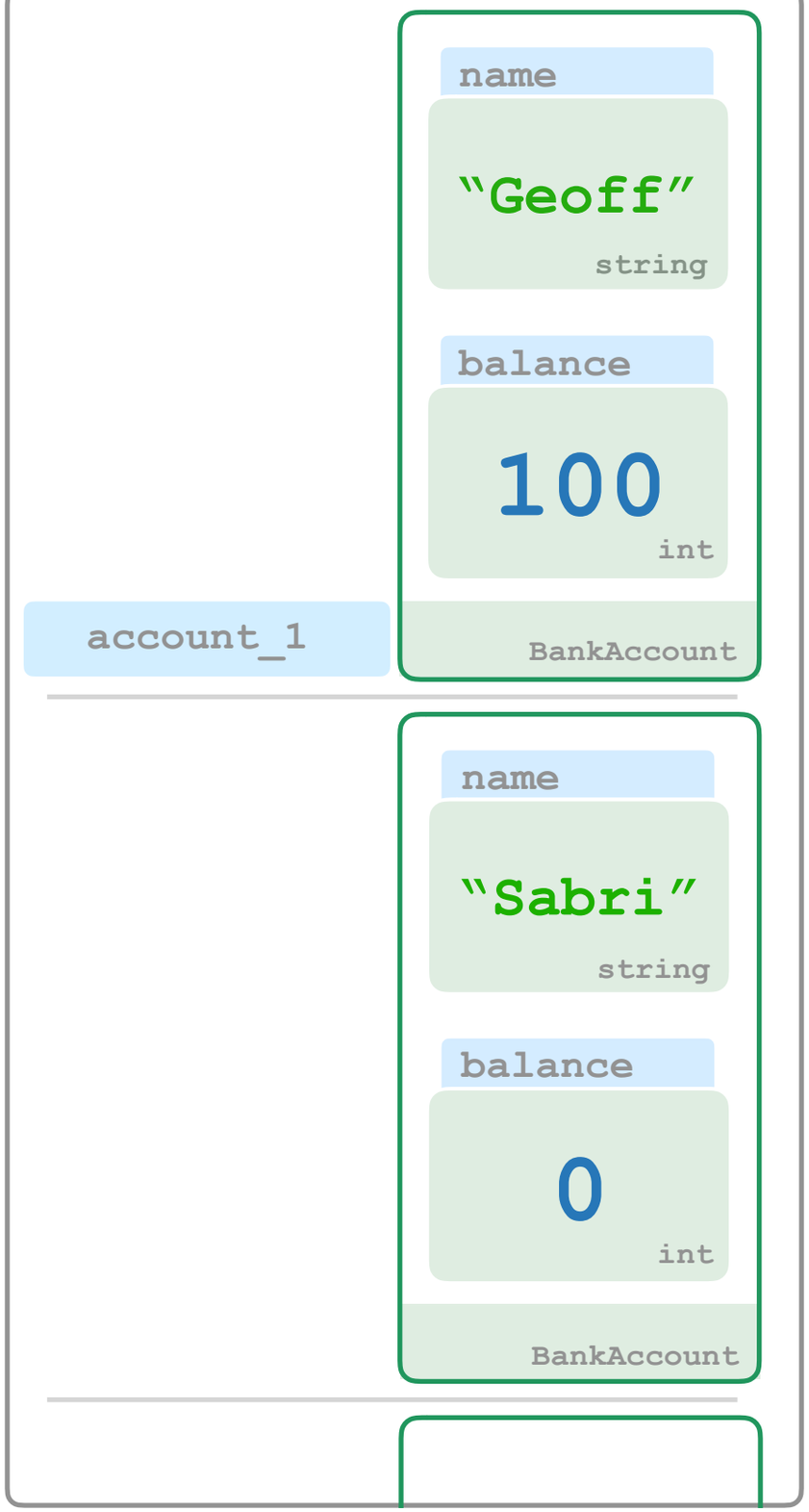
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



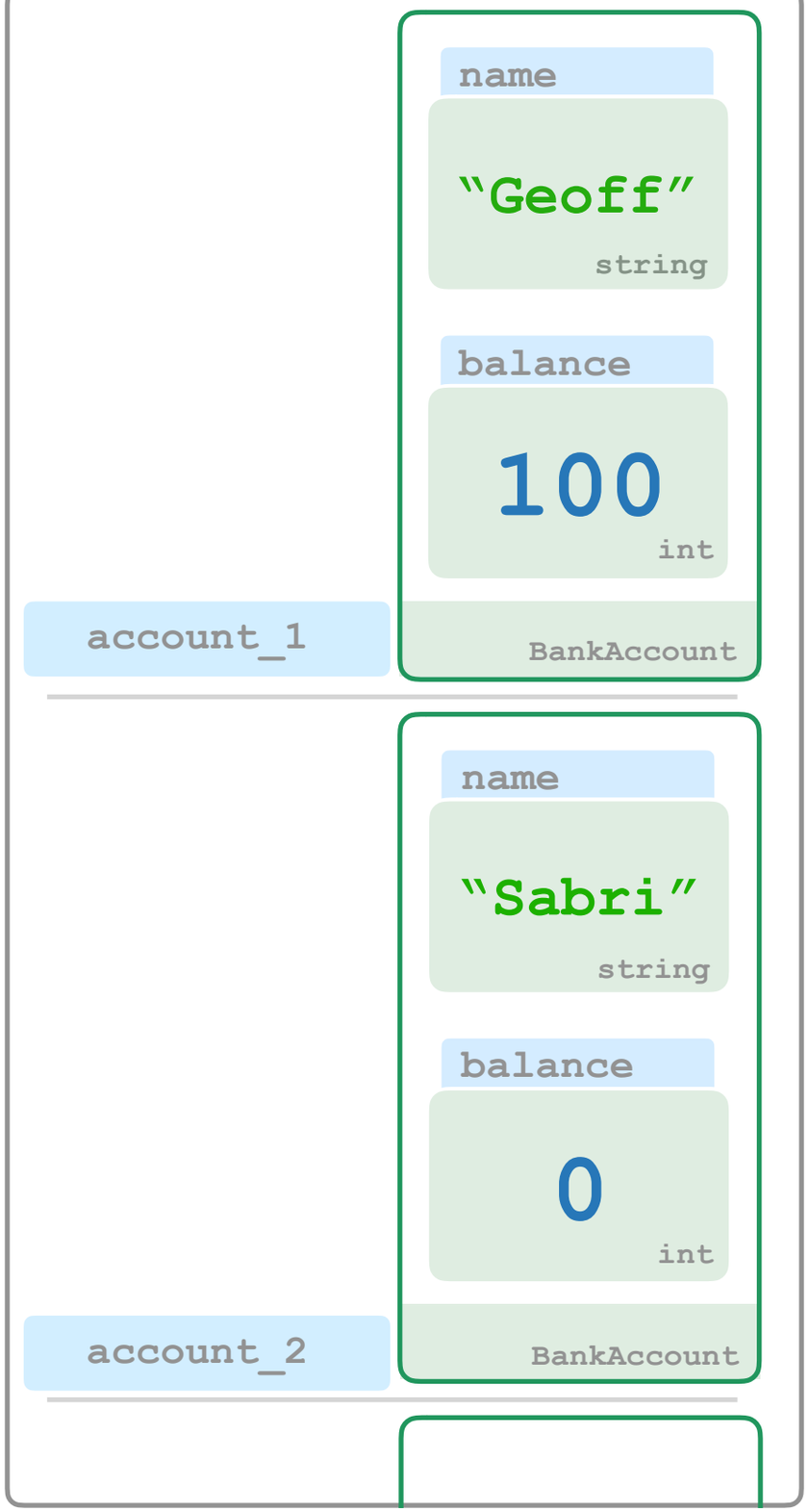
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



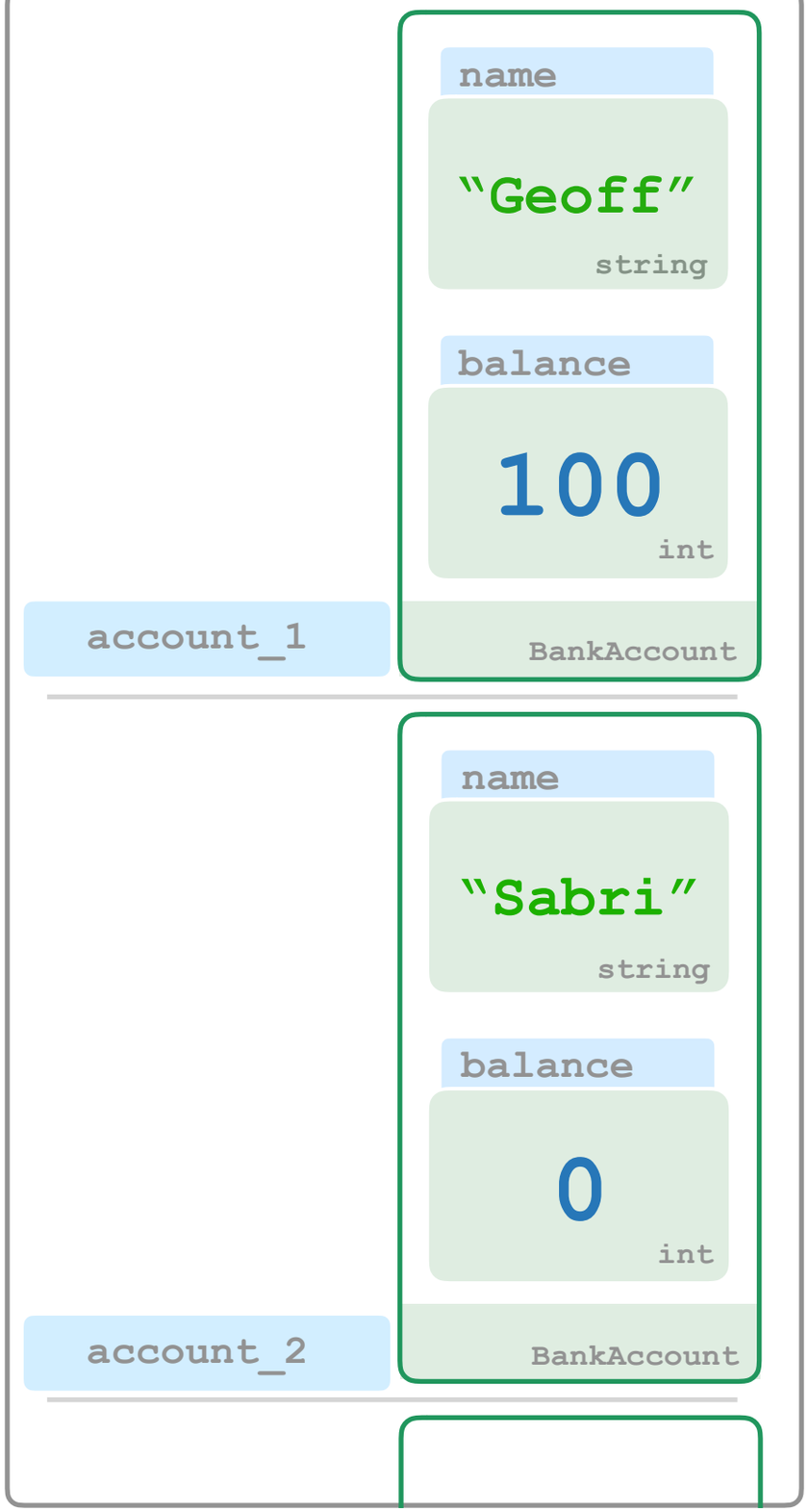
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
```

Memory



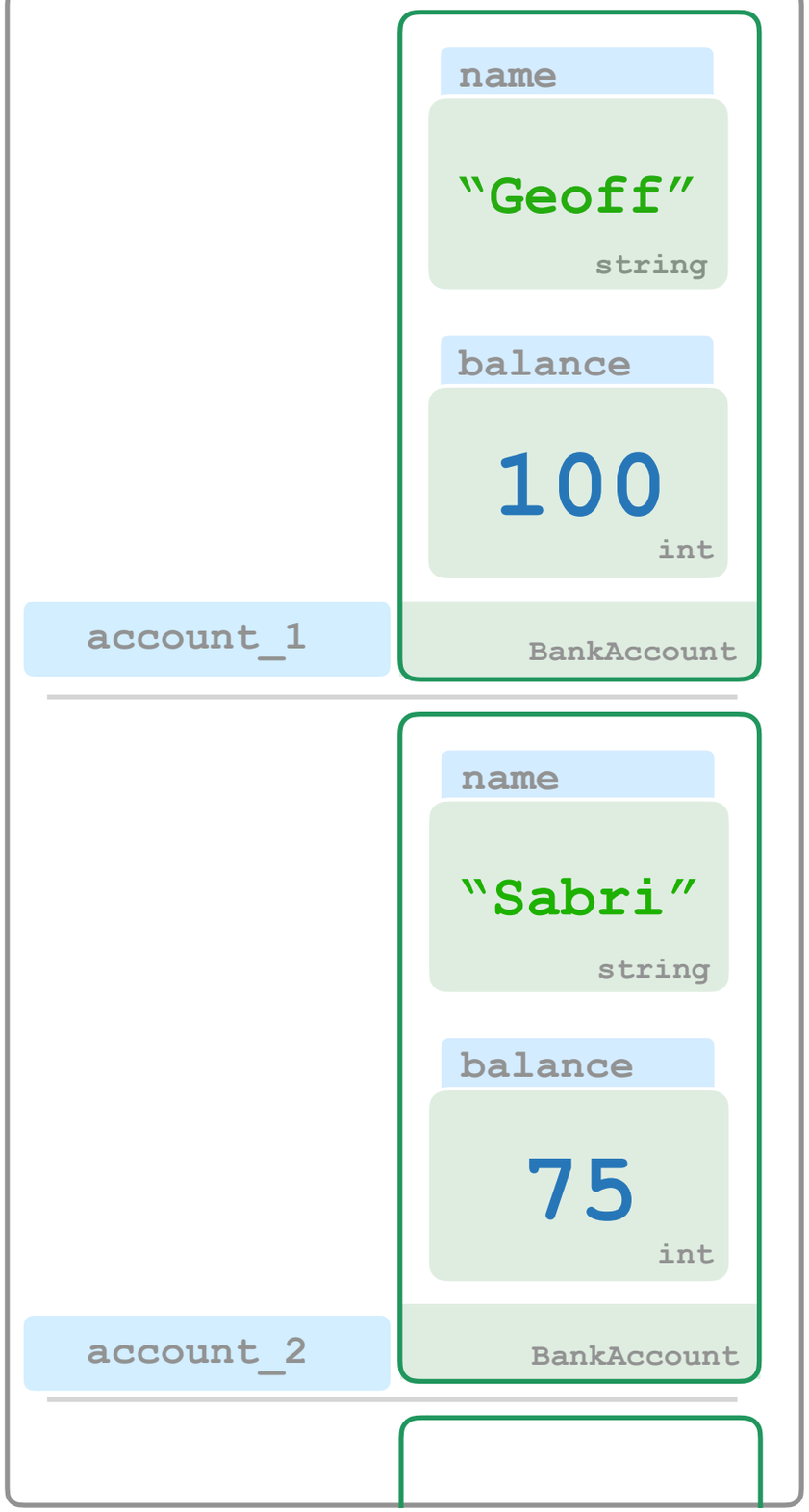
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
>
```

Memory



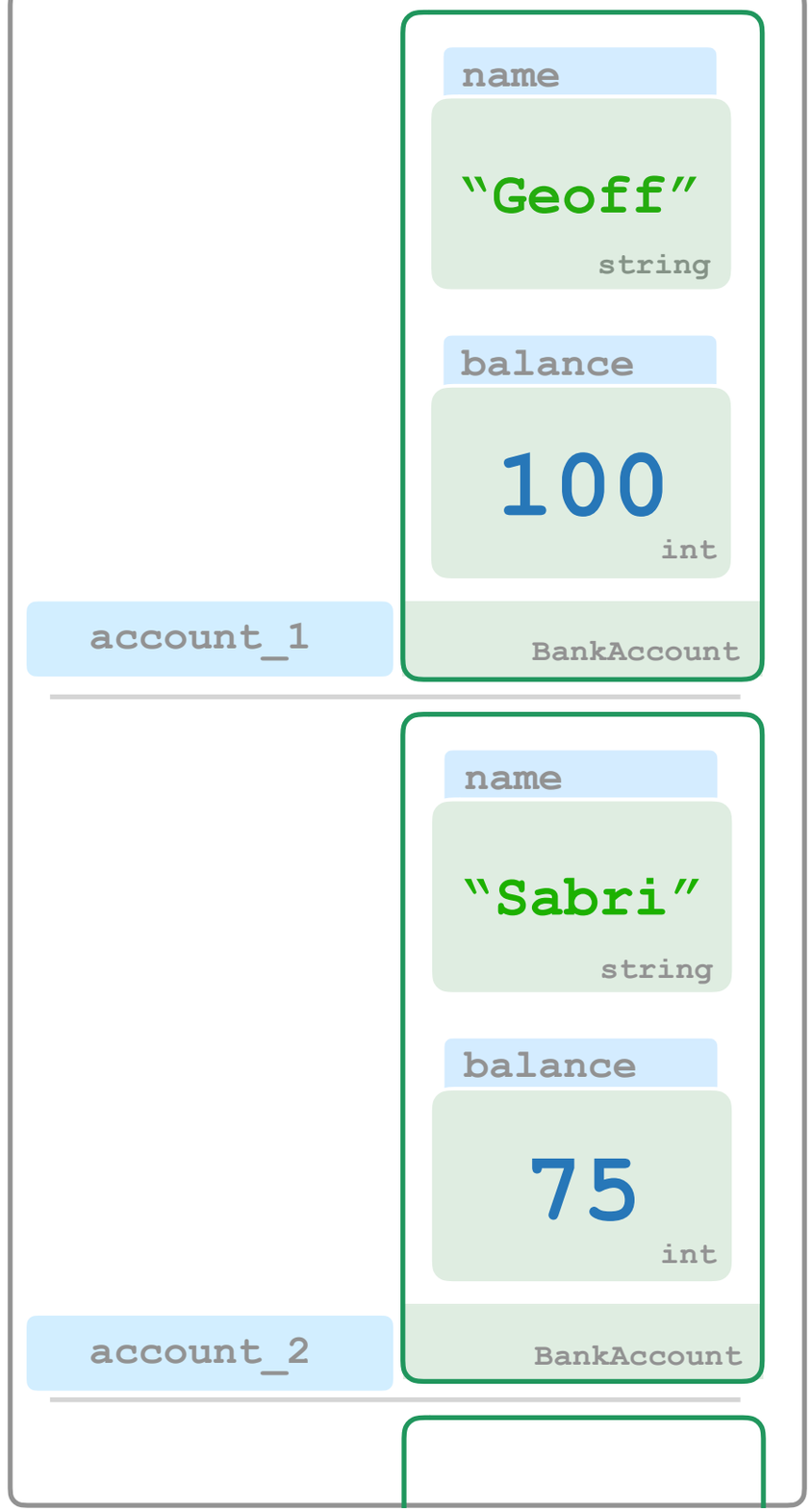
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
```

Memory



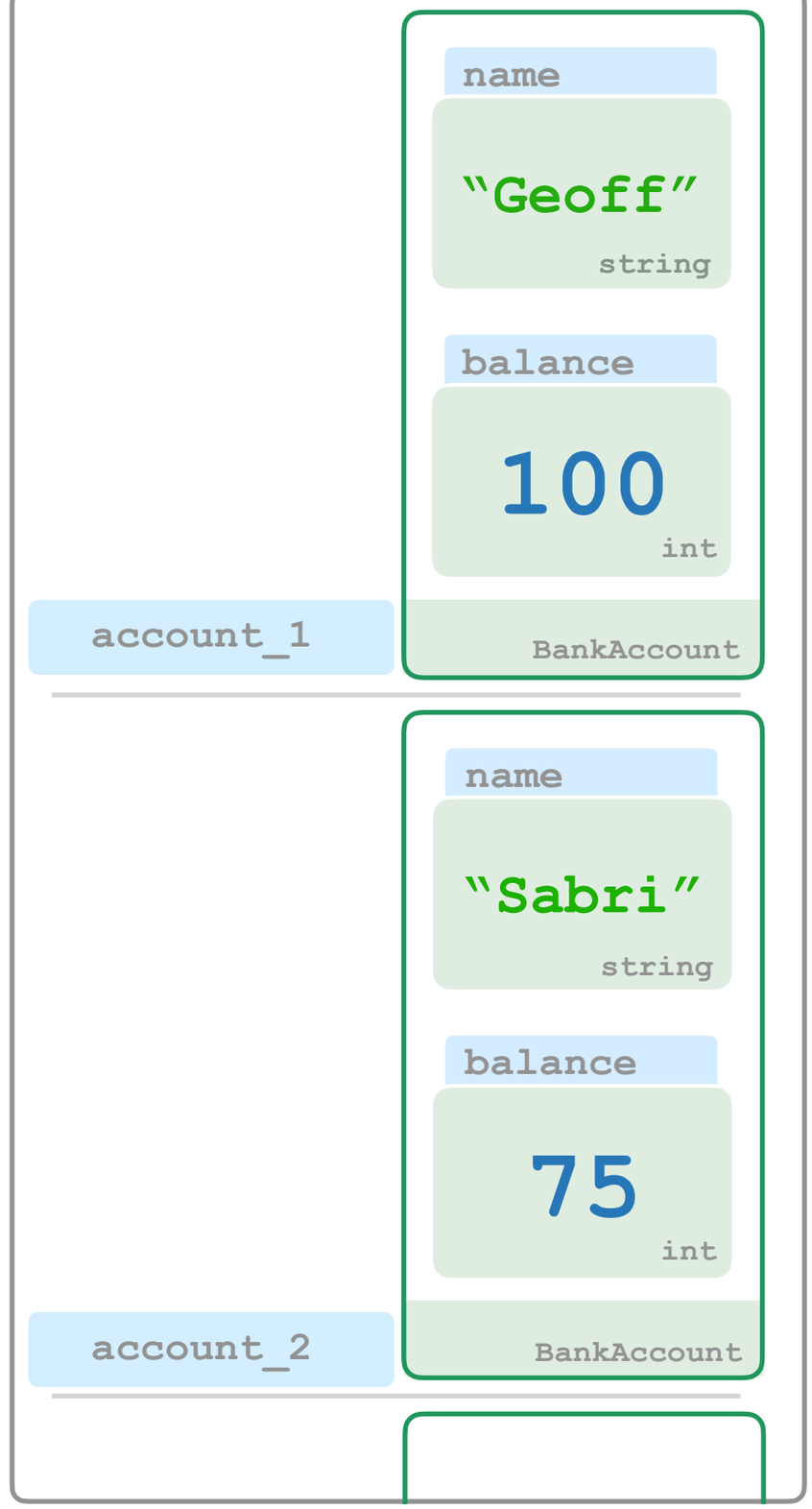
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
```

Memory



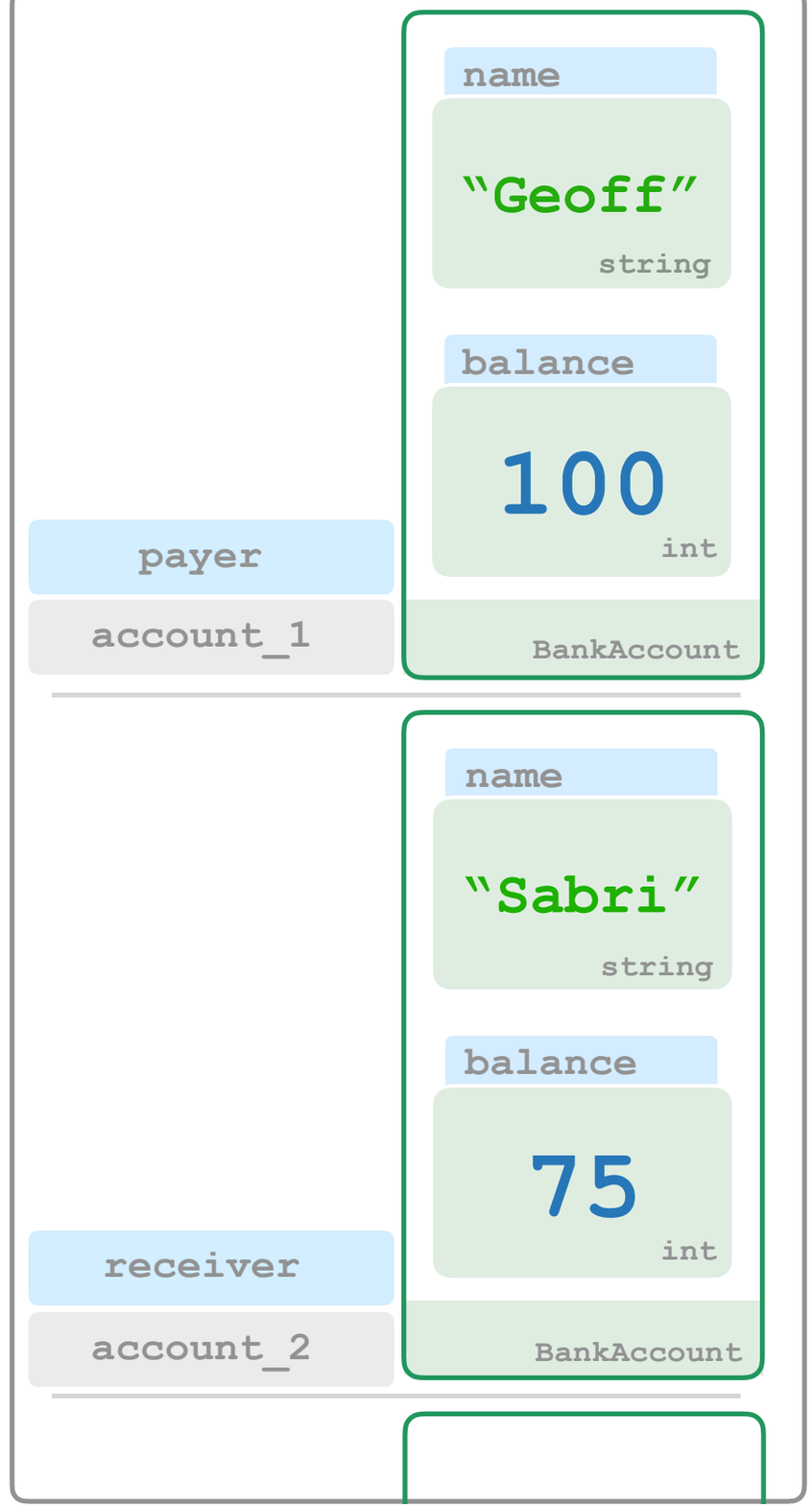
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
```

Memory



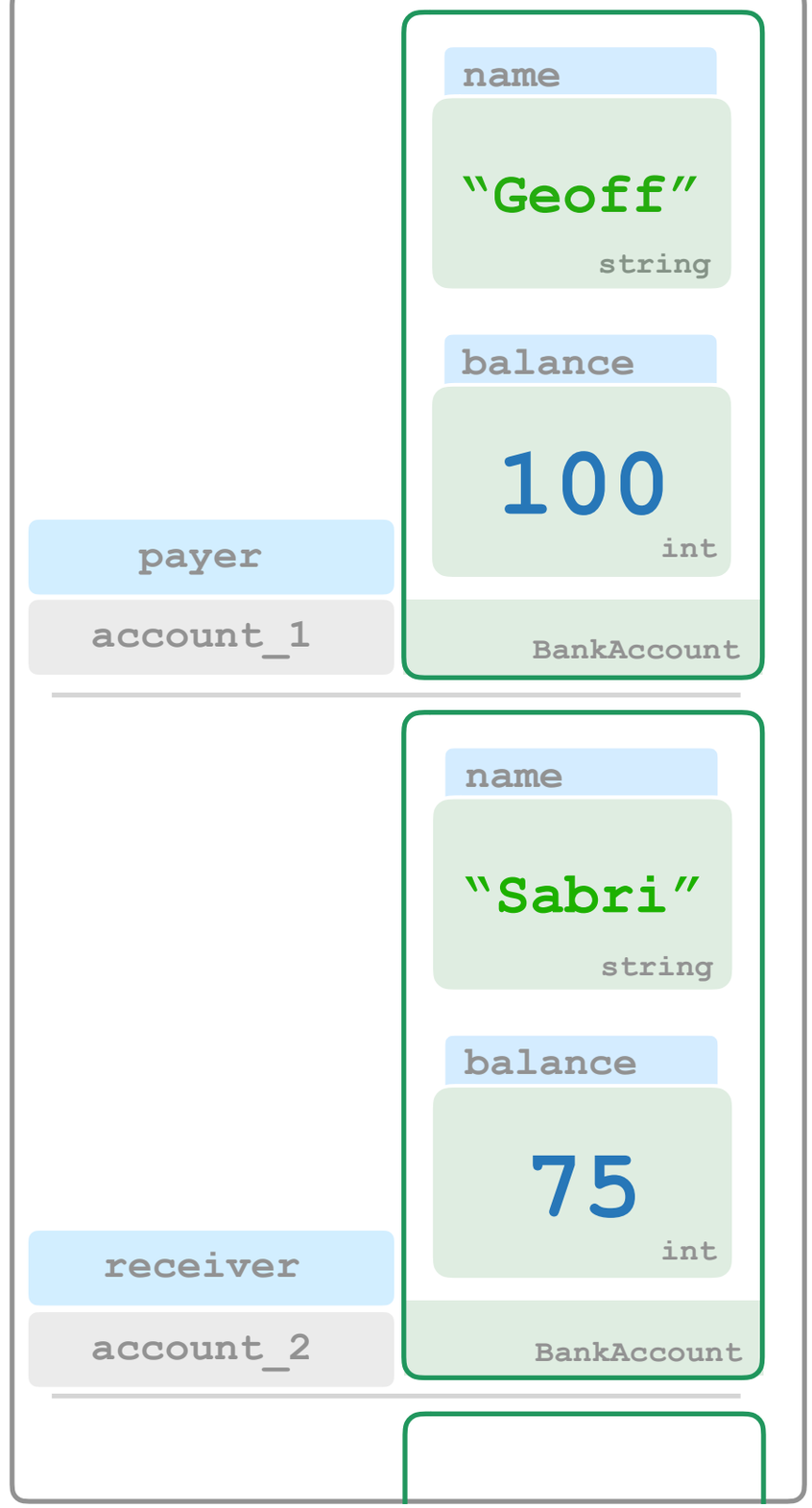
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
```

Memory



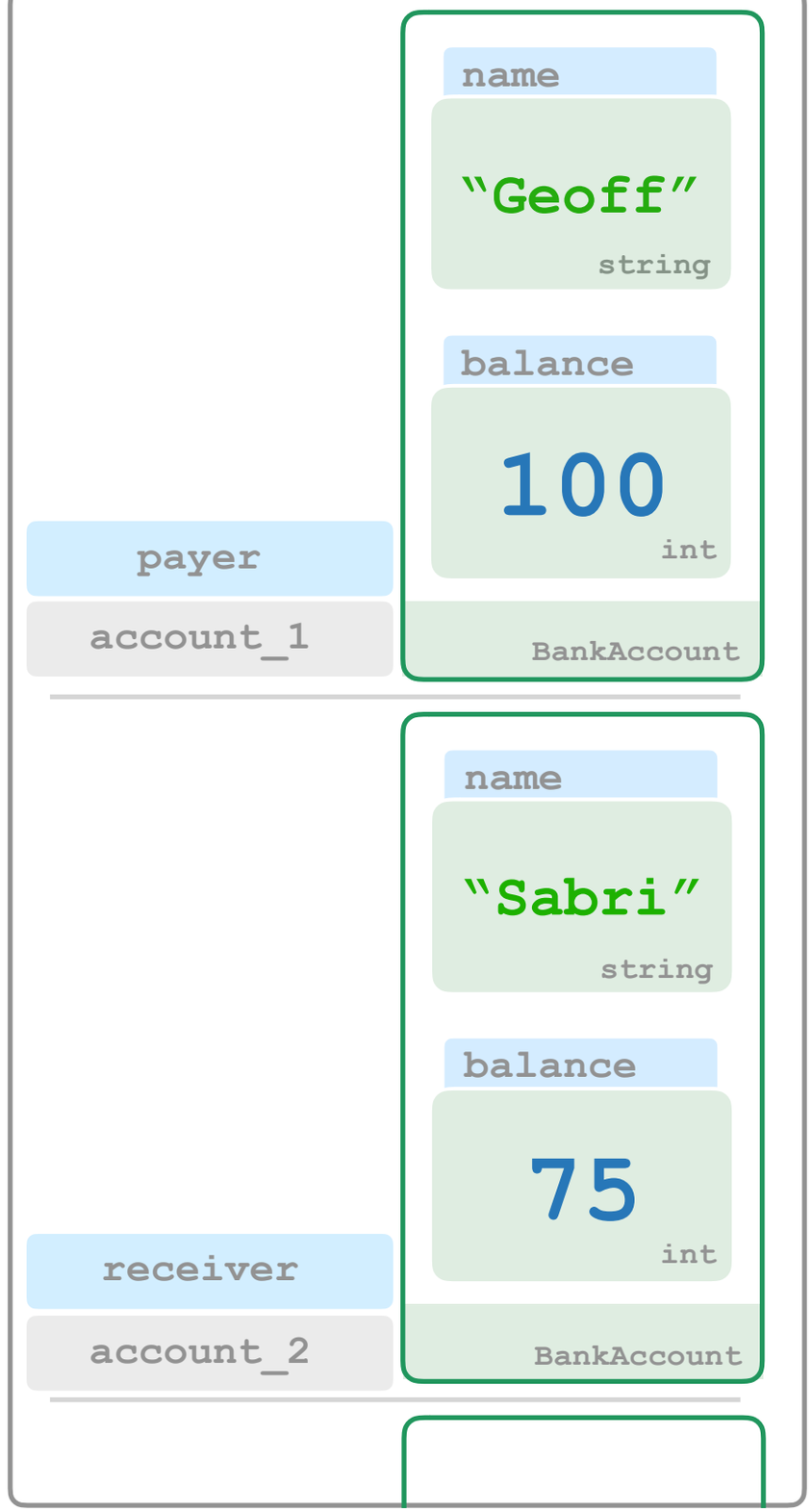
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer?
```

Memory



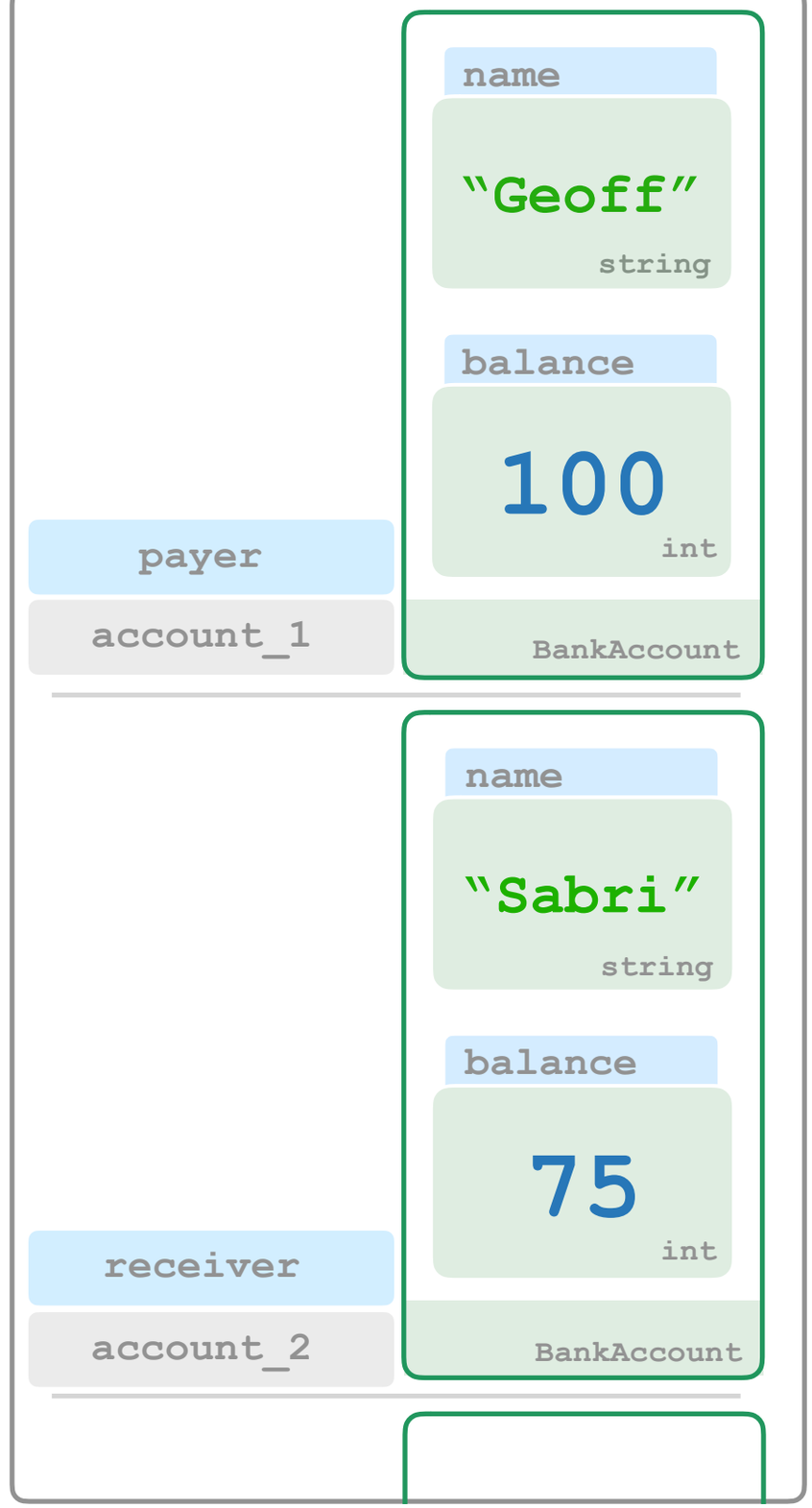
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer? 10
```

Memory



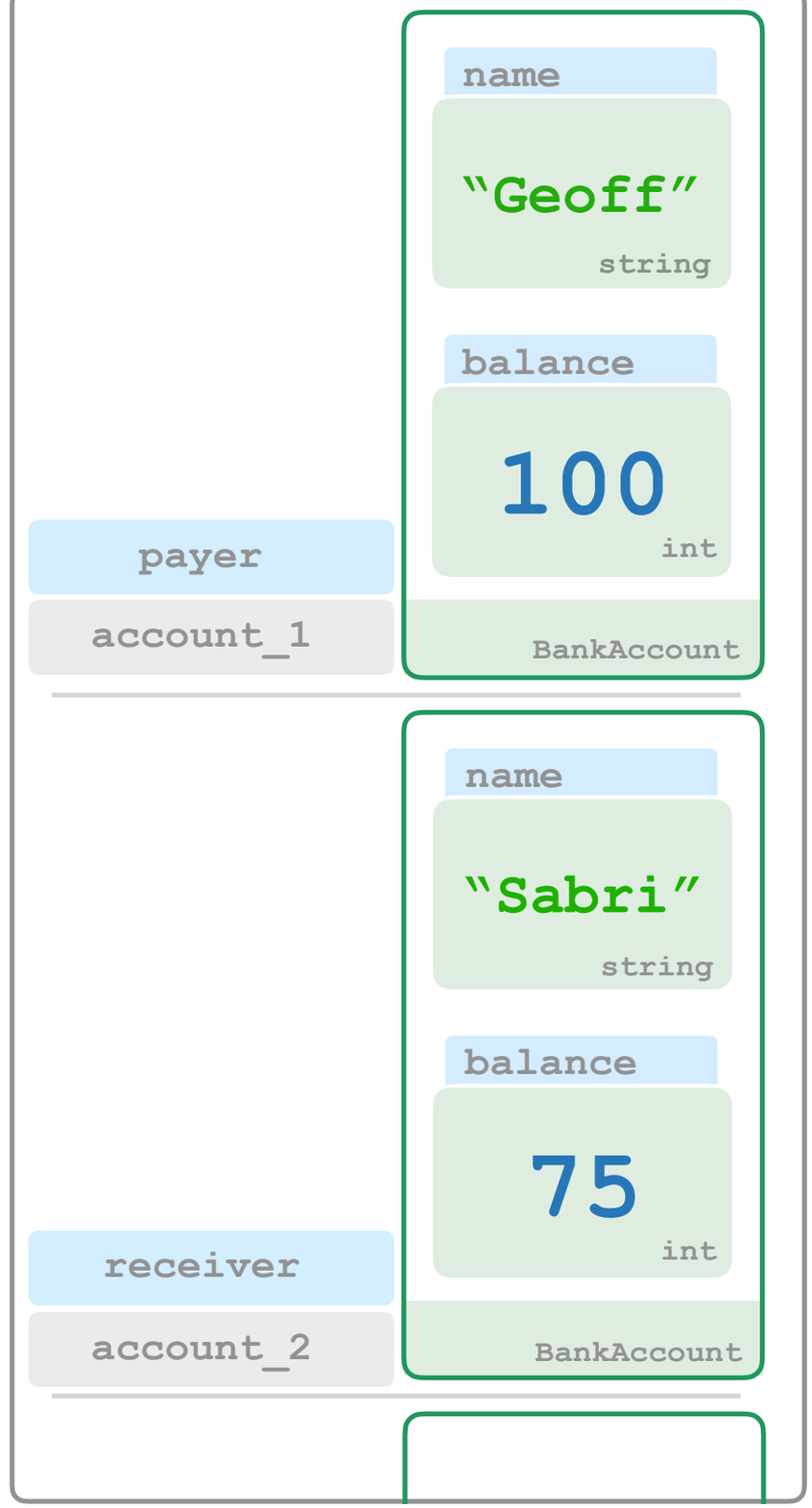
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer? 10
```

Memory



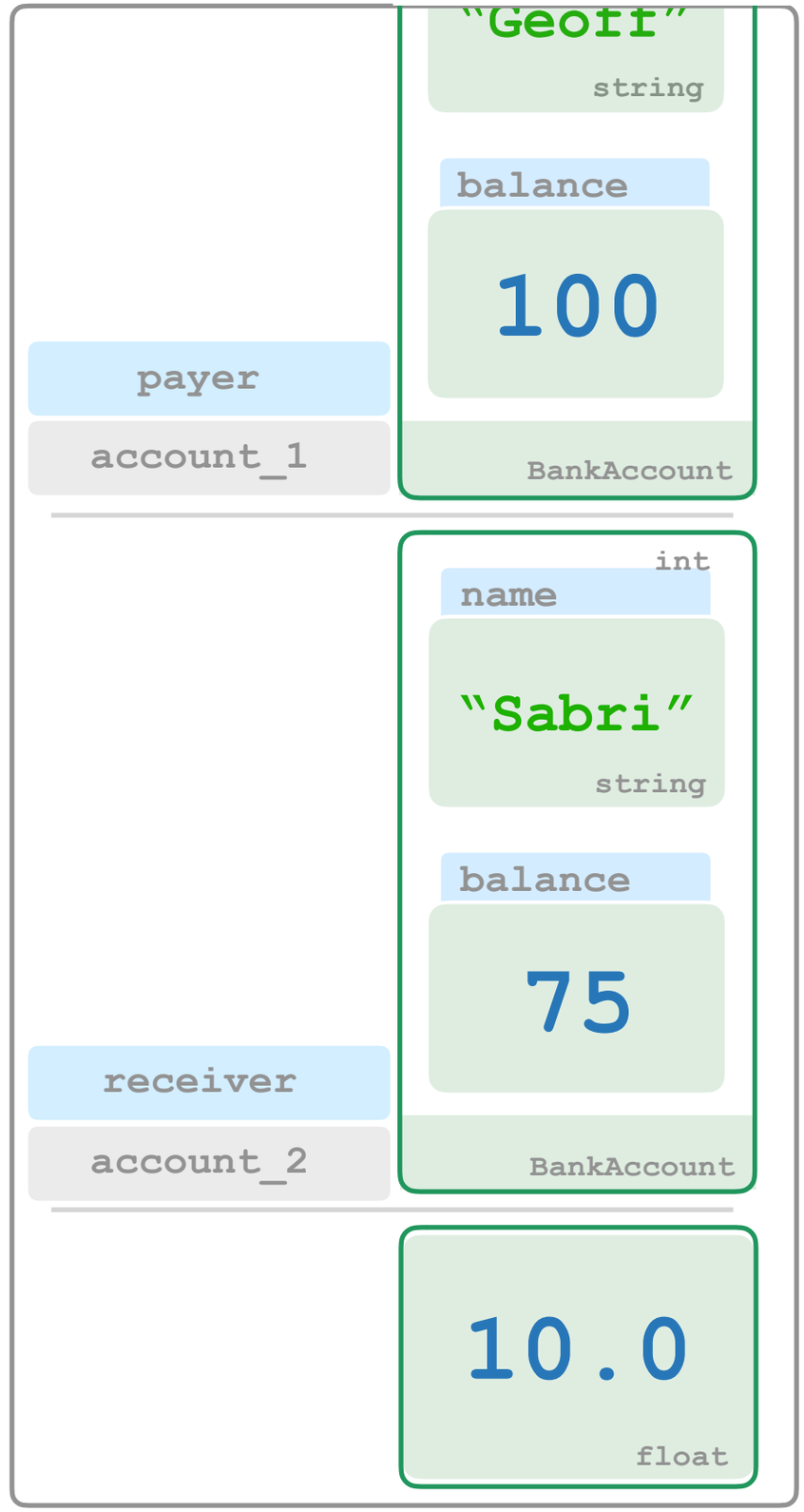
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer? 10
```

Memory



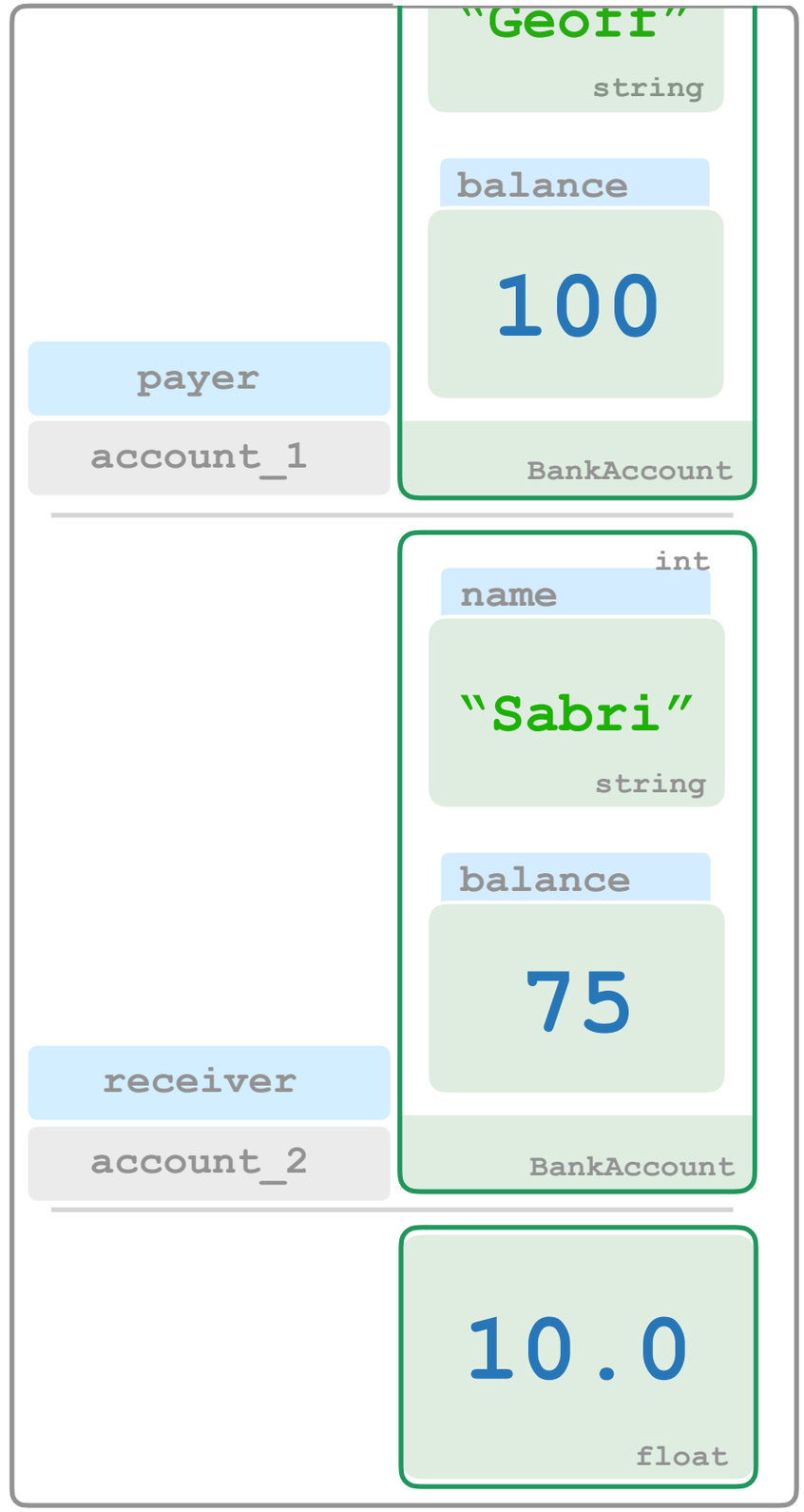
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer? 10
```

Memory



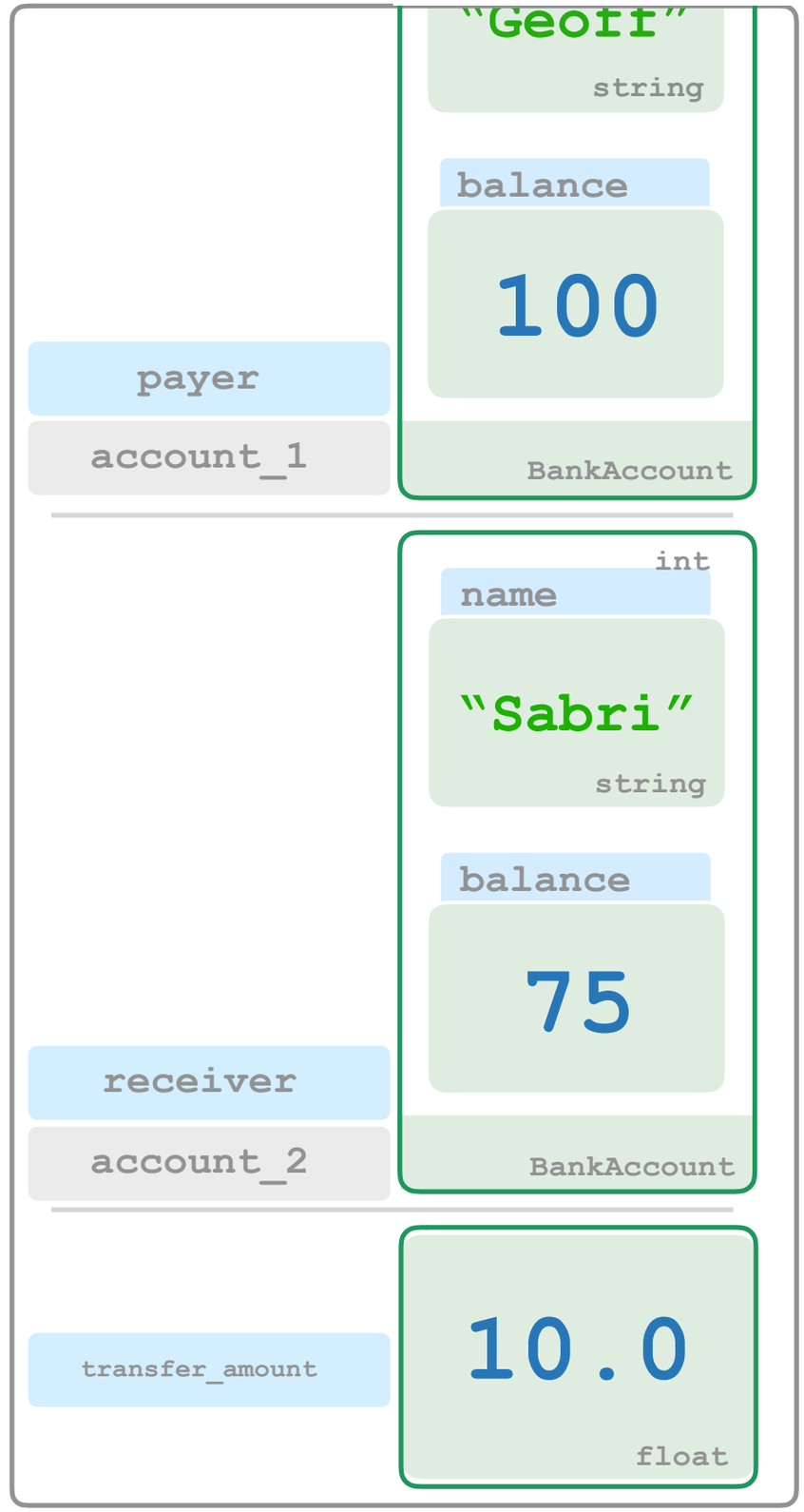
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer? 10
```

Memory



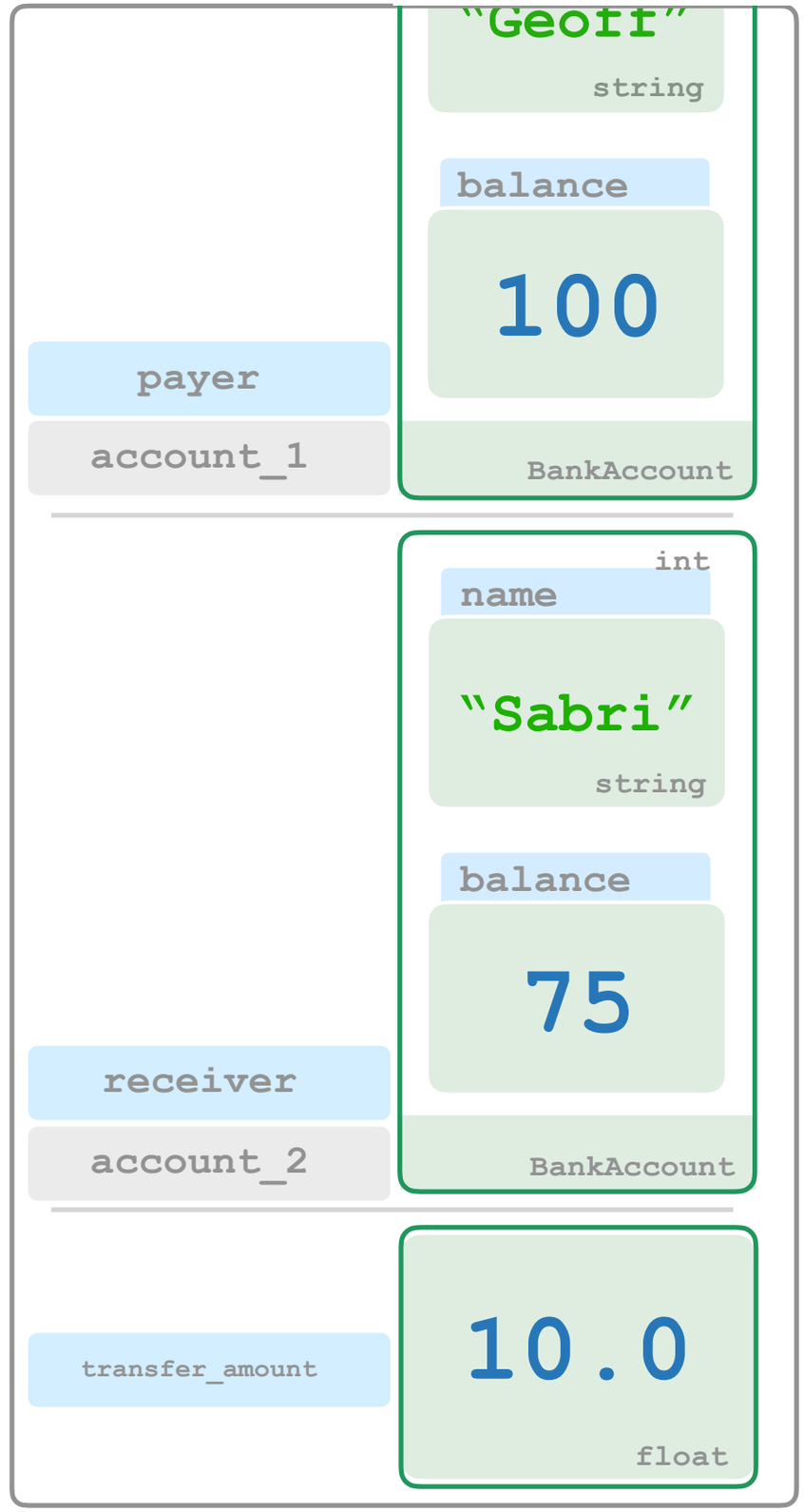
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer? 10
```

Memory



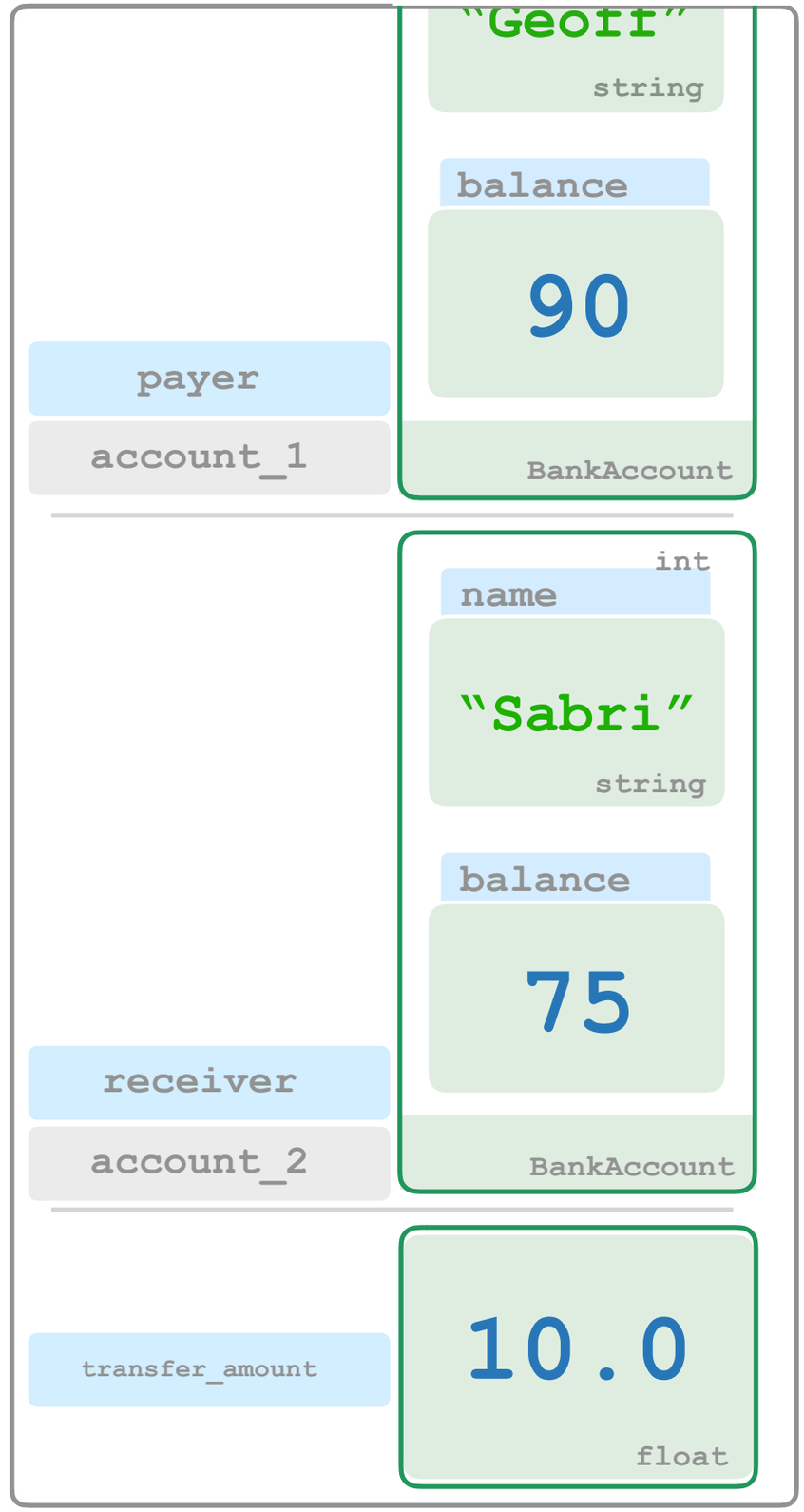
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer? 10
```

Memory



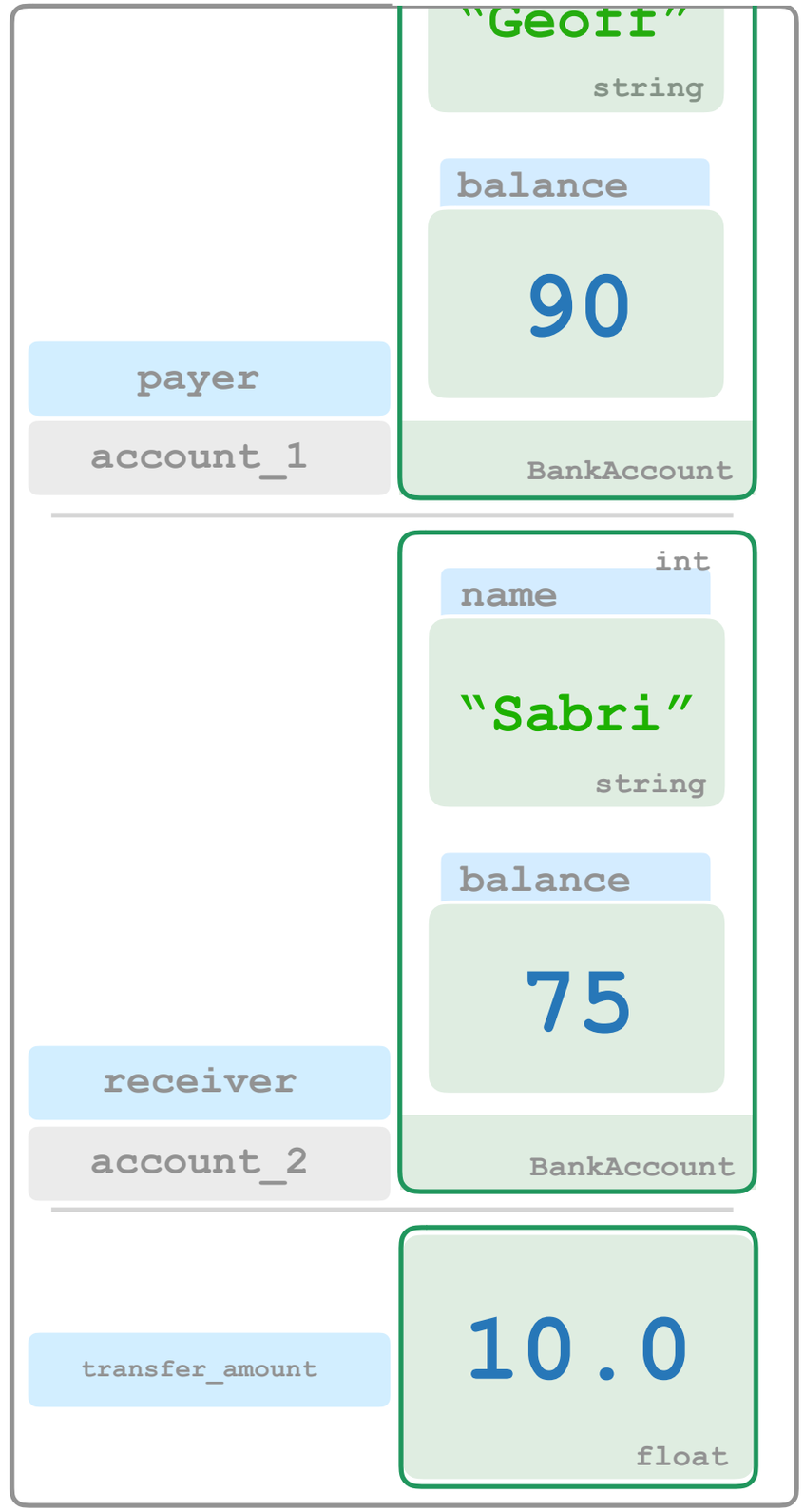
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer? 10
```

Memory



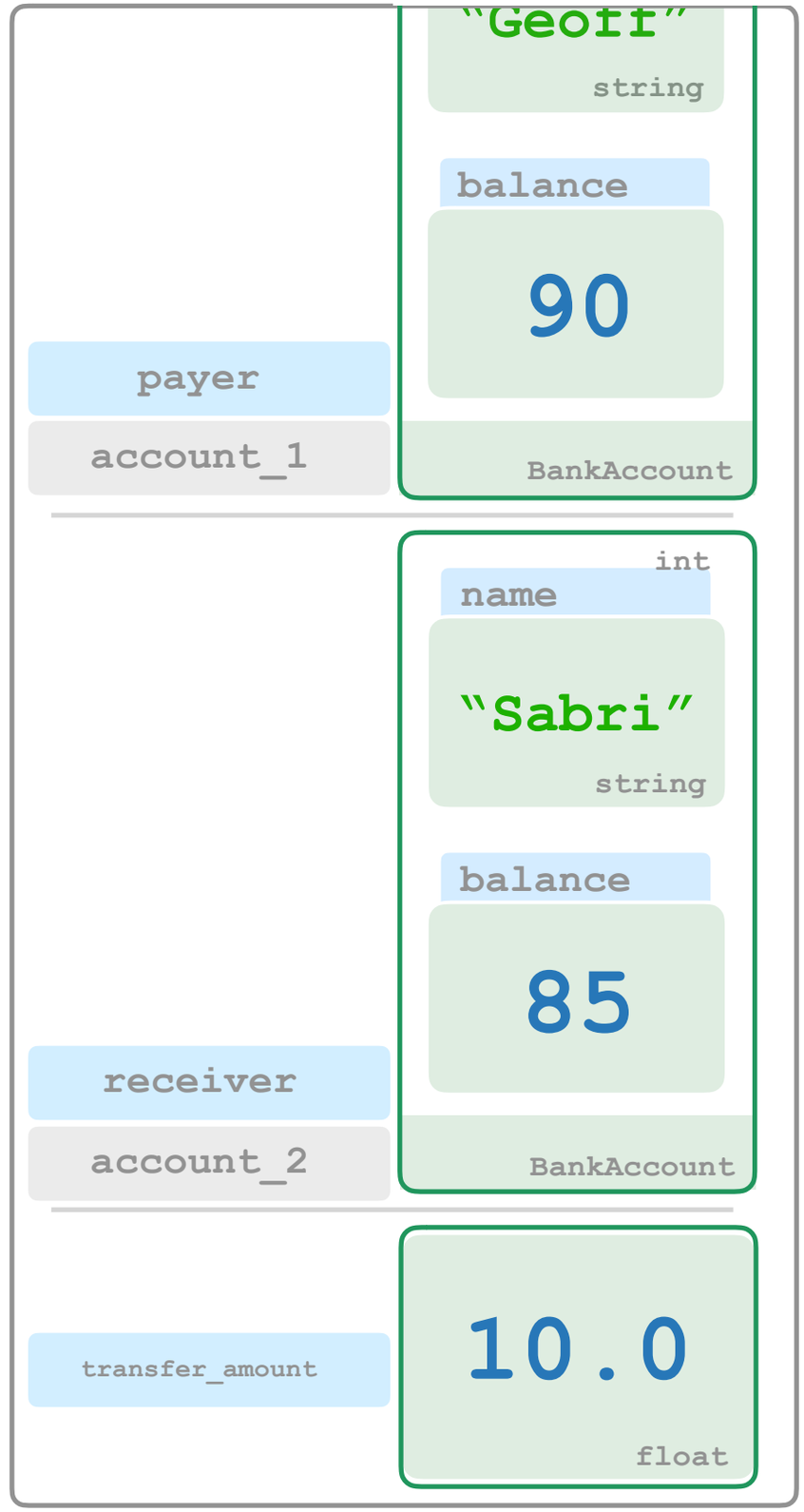
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance = payer.balance - transfer_amount
  receiver.balance = receiver.balance + transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer? 10
```

Memory



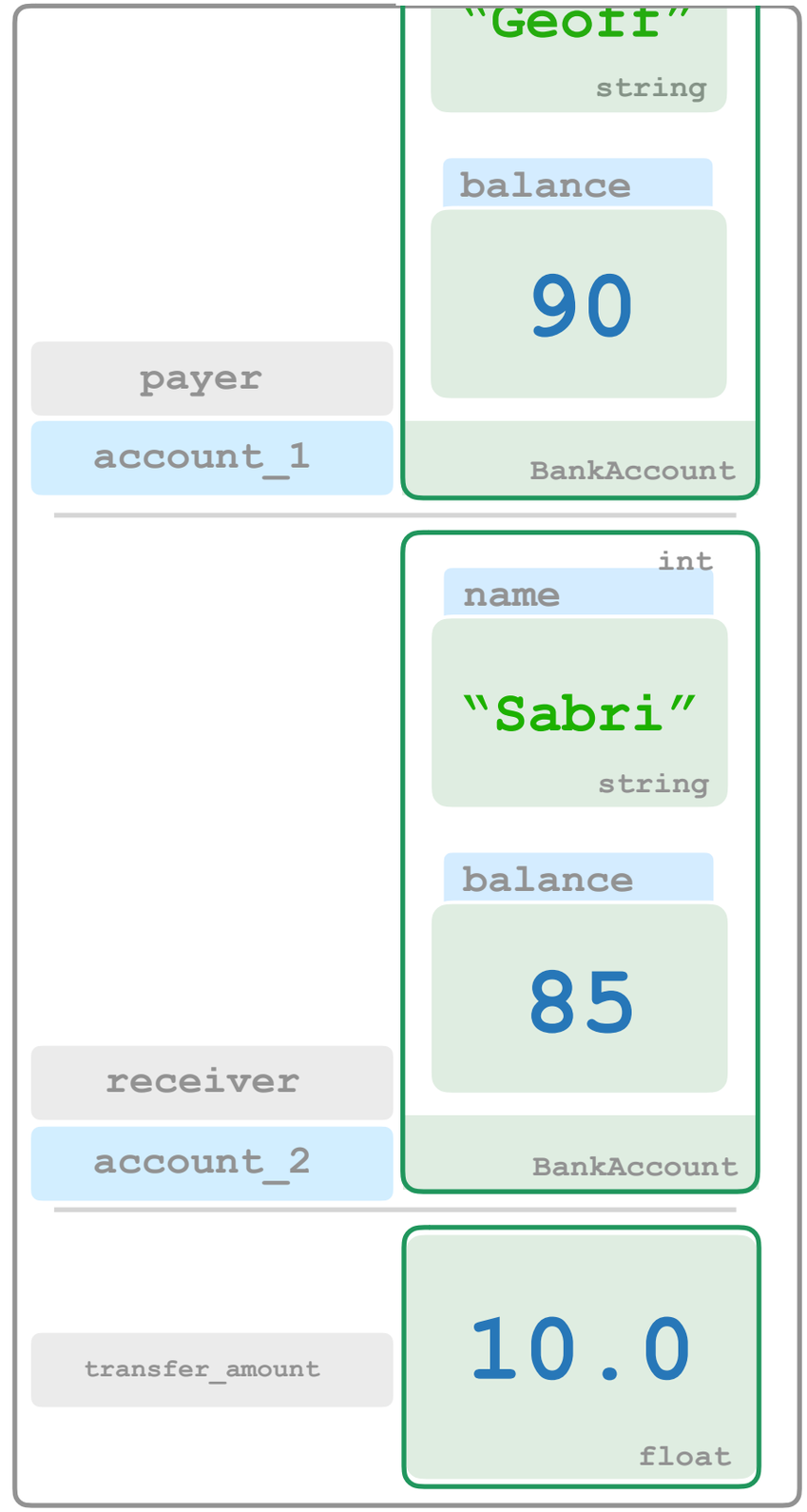
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance -= transfer_amount
  receiver.balance += transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer? 10
```

Memory



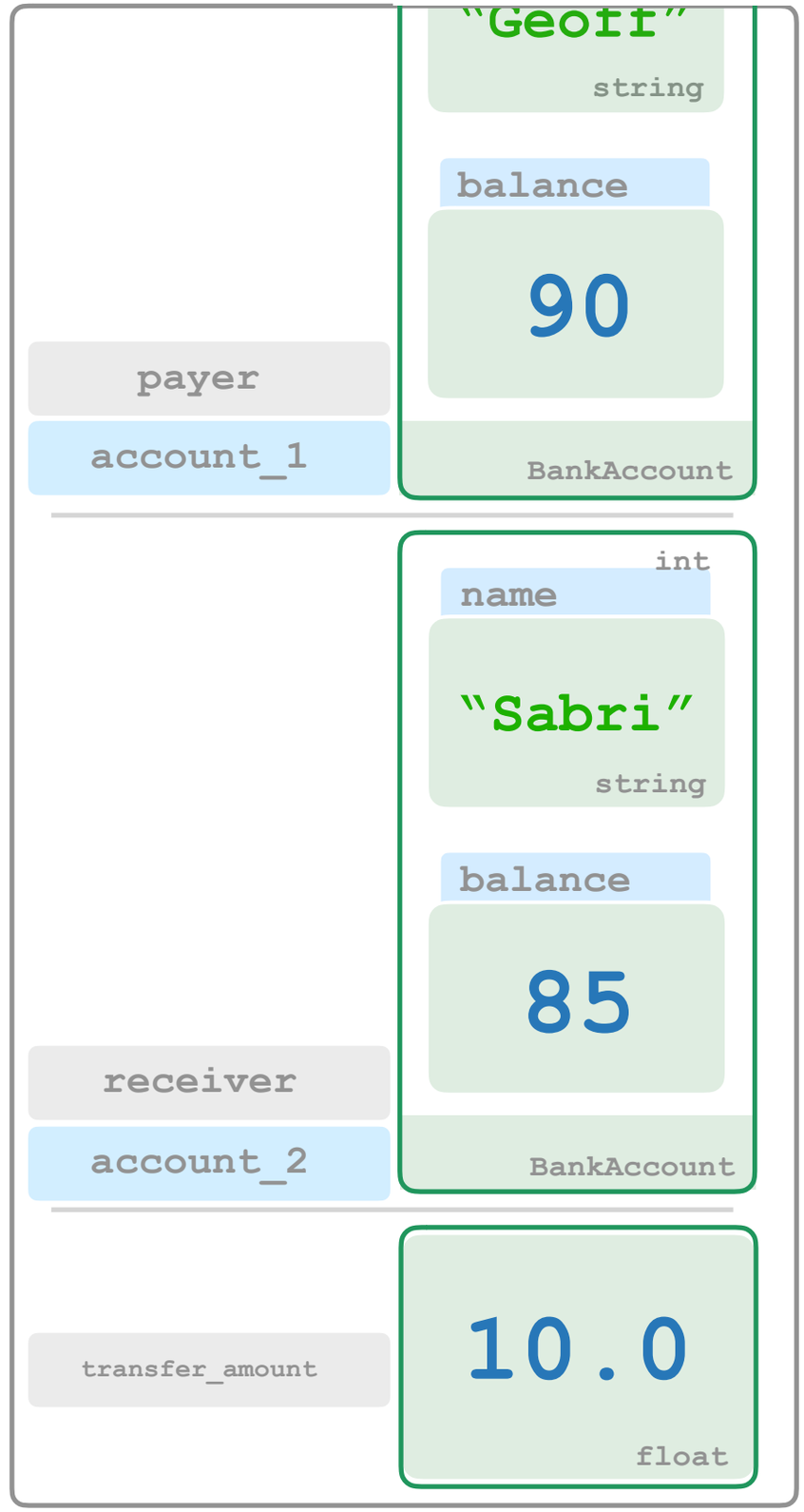
Code

```
from util import BankAccount, input_float

def transfer(payer, receiver):
  transfer_amount = input_float("How much to transfer?")
  payer.balance -= transfer_amount
  receiver.balance += transfer_amount

def main():
  account_1 = BankAccount("Geoff")
  account_1.balance = 100
  account_2 = BankAccount("Sabri")
  account_2.balance = 75
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
  transfer(account_1, account_2)
  print("Geoff has R$" + str(account_1.balance))
  print("Sabri has R$" + str(account_2.balance))
```

Output

```
Geoff has R$100
Sabri has R$75
How much to transfer? 10
Geoff has R$90.0
Sabri has R$85.0
```

Memory



Transfer (Link!)

How do we know what **functions** and **variables** are **available**?

Definition

Documentation - *Information about a class describing every usable function and object.*

Today's **Exercises**

Caixa Eletrônico