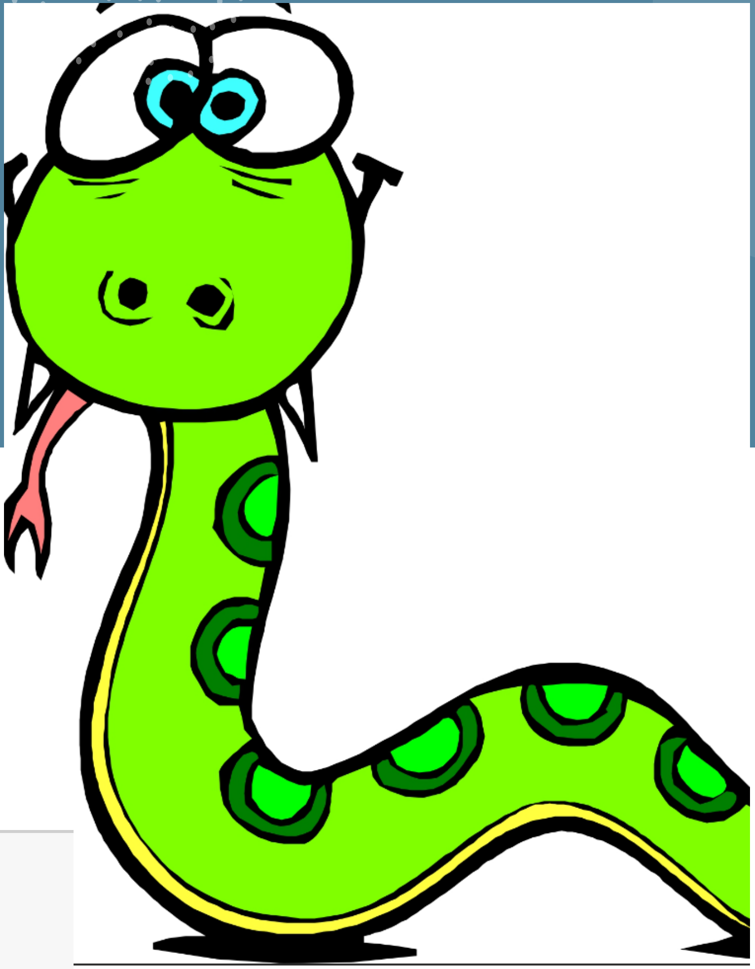
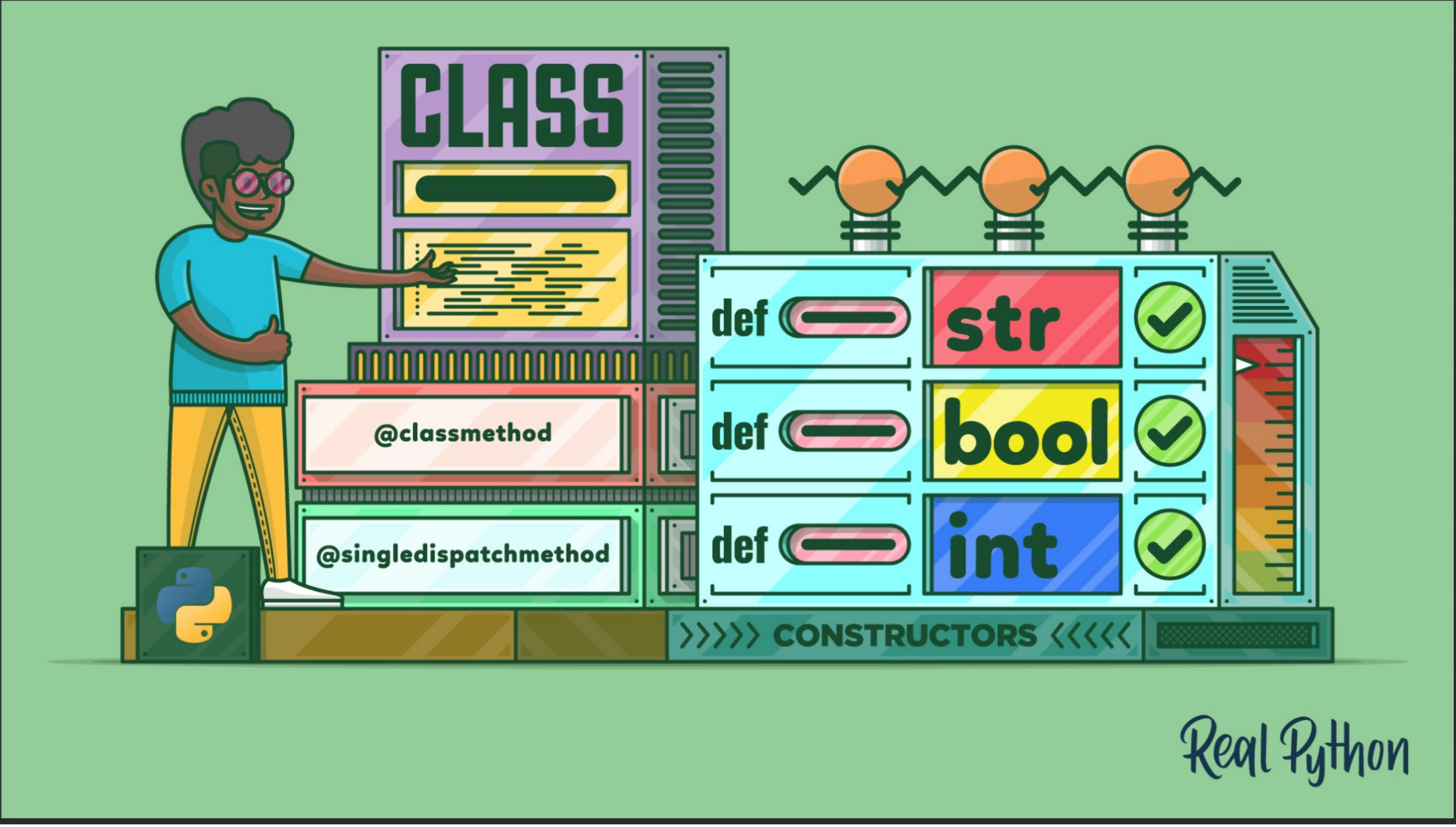
Welcome to my 5th Python Lecture

Lutz Plümer



```
WelcomeToMyLecture = "欢迎来到我的讲座"
print(WelcomeToMyLecture)
```

```
欢迎来到我的讲座
```



Classes and Objects

Classes and Objects

- Today we will talk about Objects and Classes
- They are a nice way to structure Programs
- **Class**: A class is a user-defined data structure that binds the data members and methods into a single unit. Class is a blueprint or code template for object creation. Using a class, you can create as many objects as you want.
- **Object**: An object is an instance of a class. It is a collection of attributes (variables) and methods. We use the object of a class to perform actions.
- Objects have two characteristics: They have states and behaviors An object has attributes and methods attached to it.
- **Attributes** represent its **state**.
- **Methods** represent its **behavior**.

Example: Bank Account

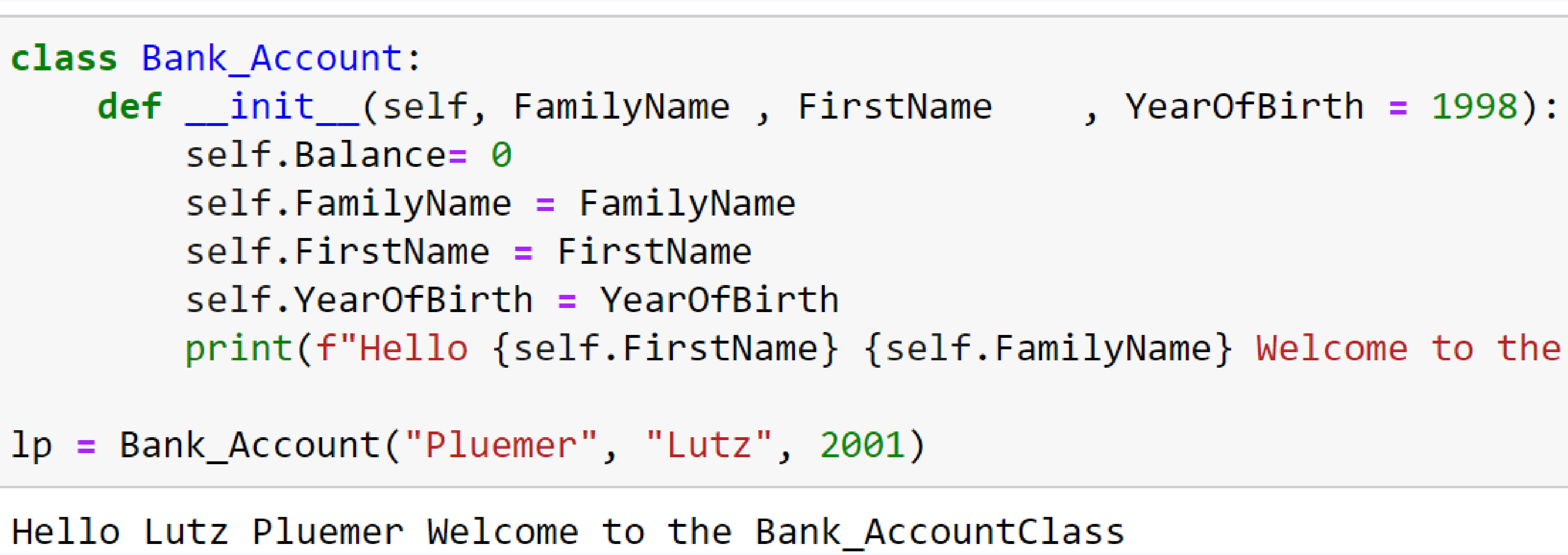
- Let’s start with an example on a class managing Bank Accounts
- We need the following attributes:
- FamilyName
- FirstName
- YearOfBirth
- Balance
- Let’s start with defining the class BankAccount

The Class Bank_Account

```
class Bank_Account:
    def __init__(self, FamilyName , FirstName    , YearOfBirth = 1998):
        self.Balance= 0
        self.FamilyName = FamilyName
        self.FirstName = FirstName
        self.YearOfBirth = YearOfBirth
        print(f"Hello {self.FirstName} {self.FamilyName} \
            Welcome to the Bank_AccountClass")
```

- **class** is the keyword initiating the class definition
- **__init__** if the (obligatory) name for the constructor, the function which generates an instance of that class
- **self** is the keyword referring to the instance of this class
- These are the **conventions** you need to know, understand and apply
- They are typical for object oriented programming

The first short Example



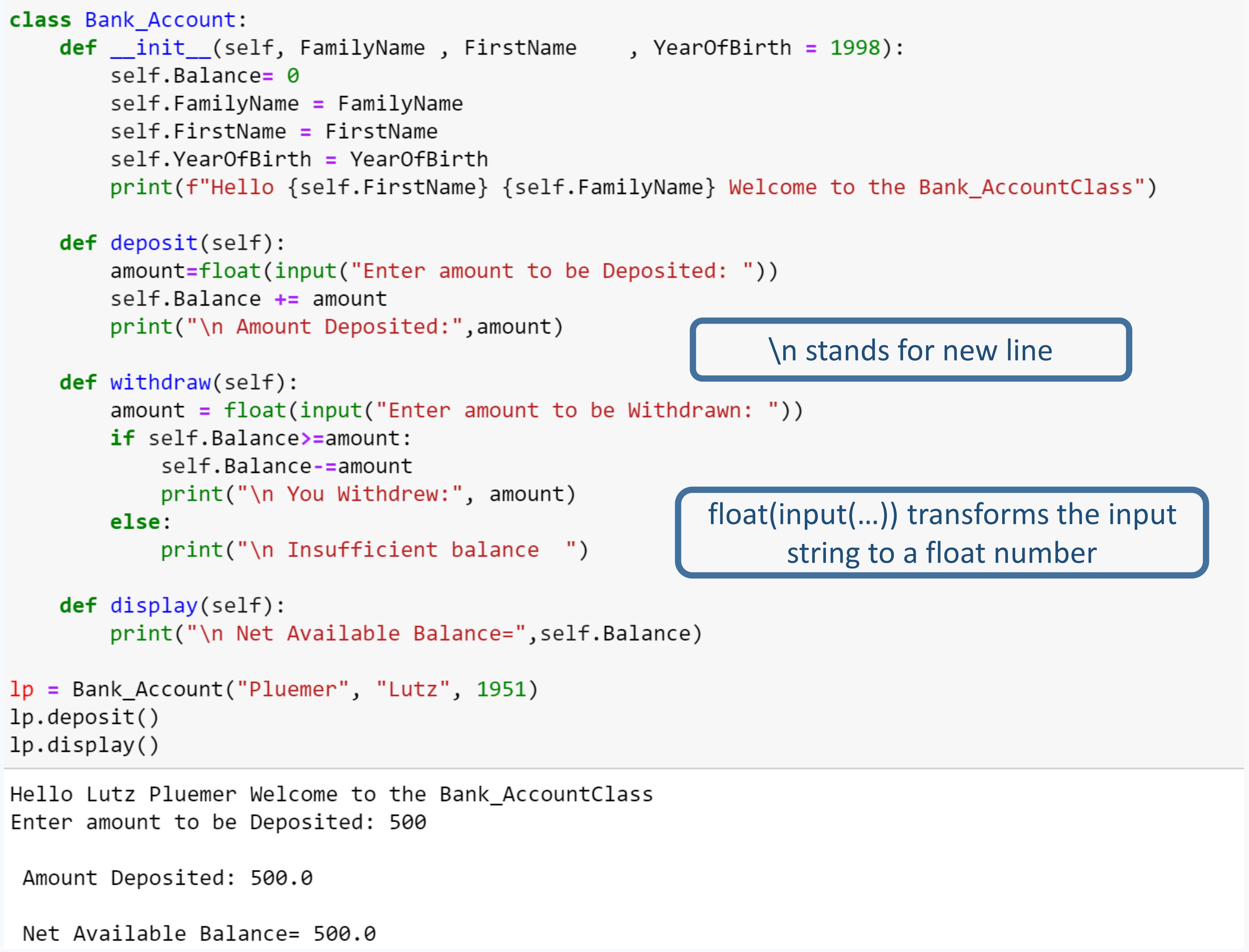
```
class Bank_Account:
    def __init__(self, FamilyName , FirstName     , YearOfBirth = 1998):
        self.Balance= 0
        self.FamilyName = FamilyName
        self.FirstName = FirstName
        self.YearOfBirth = YearOfBirth
        print(f"Hello {self.FirstName} {self.FamilyName} Welcome to the

lp = Bank_Account("Pluemer", "Lutz", 2001)
```

```
Hello Lutz Pluemer Welcome to the Bank_AccountClass
```

Note that the features of the class are implicitly defined in
the constructor function named **__init__**
note that __init__ starts and ends with **two underscores**

Now lets extend this class by some more Functions

```
def deposit(self, amount = 0):
        if amount == 0:
            amount=float(input("Enter amount to be Deposited: "))
        self.Balance += amount
        print("\n Amount Deposited:",amount)

    def withdraw(self, amount = 0):
        if amount == 0:
            amount = float(input("Enter amount to be Withdrawn: "))
        if self.Balance >= amount:
            self.Balance -= amount
            print("\n You Withdrew:", amount)
        else:
            print("\n Insufficient balance  ")
```



```python
class Bank_Account:
    def __init__(self, FamilyName , FirstName    , YearOfBirth = 1998):
        self.Balance= 0
        self.FamilyName = FamilyName
        self.FirstName = FirstName
        self.YearOfBirth = YearOfBirth
        print(f"Hello {self.FirstName} {self.FamilyName} Welcome to the Bank_AccountClass")

    def deposit(self):
        amount=float(input("Enter amount to be Deposited: "))
        self.Balance += amount
        print("\n Amount Deposited:",amount)

    def withdraw(self):
        amount = float(input("Enter amount to be Withdrawn: "))
        if self.Balance>=amount:
            self.Balance-=amount
            print("\n You Withdrew:", amount)
        else:
            print("\n Insufficient balance  ")

    def display(self):
        print("\n Net Available Balance=",self.Balance)

lp = Bank_Account("Pluemer", "Lutz", 1951)
lp.deposit()
lp.display()
```

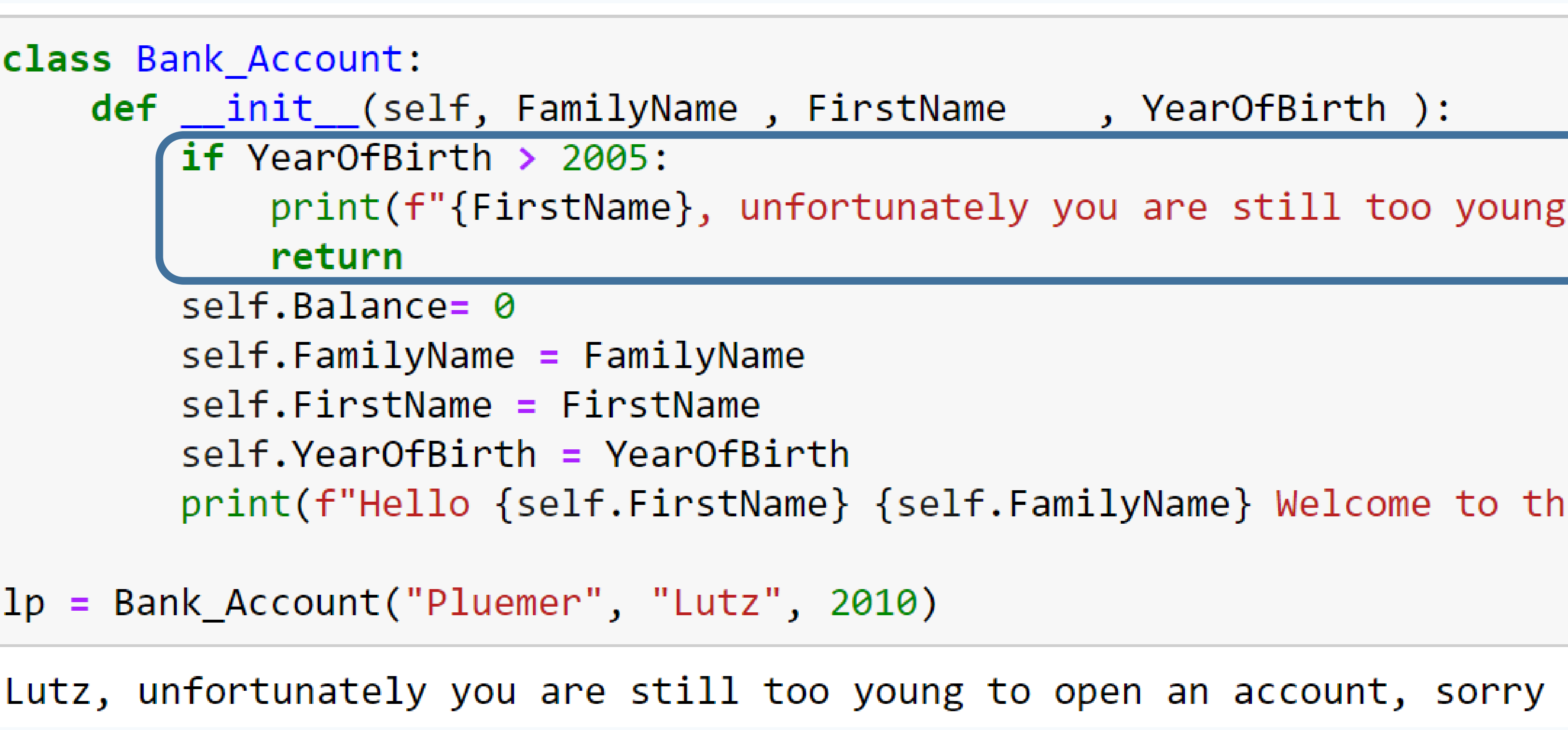
```
Hello Lutz Pluemer Welcome to the Bank_AccountClass
Enter amount to be Deposited: 500

 Amount Deposited: 500.0

 Net Available Balance= 500.0
```

\n stands for new line

float(input(...)) transforms the input string to a float number

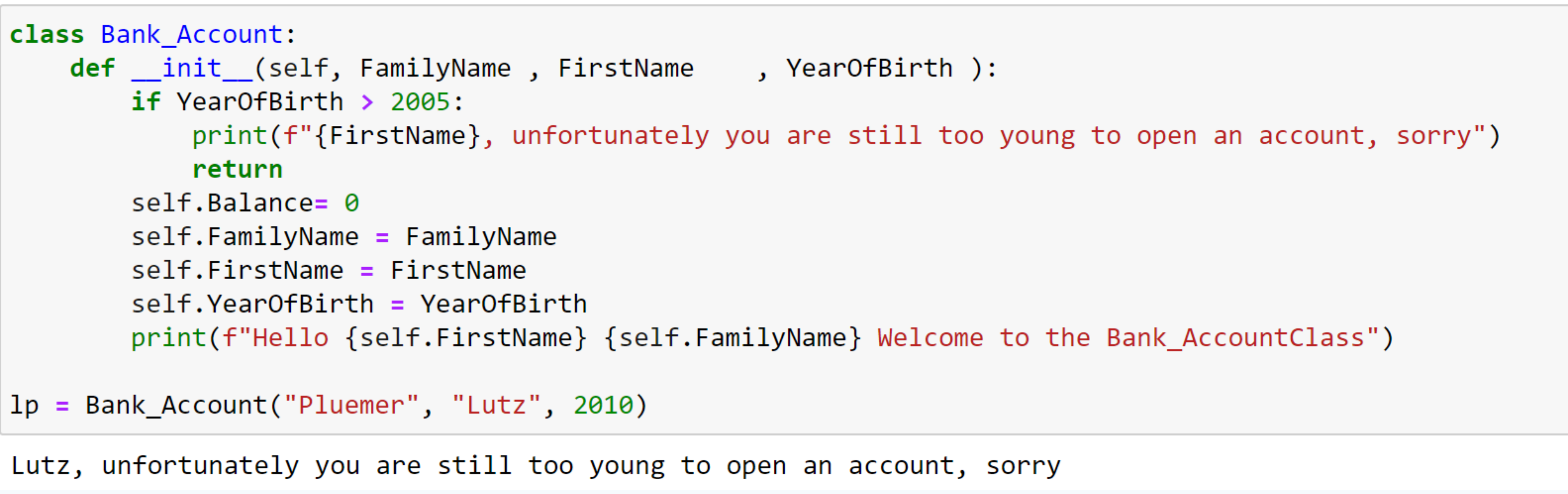
Improvements

- This class can still be improved
- For instance you should not allow negative deposits
- You should not allow children aged under 18 to open an account
- If you program it, the **return** command may be used to enforce immediate end
- after return the function is terminated immediately
- return can be used **without** an argument



```
class Bank_Account:
    def __init__(self, FamilyName , FirstName     , YearOfBirth ):
        if YearOfBirth > 2005:
            print(f"{FirstName}, unfortunately you are still too young
            return
        self.Balance= 0
        self.FamilyName = FamilyName
        self.FirstName = FirstName
        self.YearOfBirth = YearOfBirth
        print(f"Hello {self.FirstName} {self.FamilyName} Welcome to th

lp = Bank_Account("Pluemer", "Lutz", 2010)
```

```
Lutz, unfortunately you are still too young to open an account, sorry
```



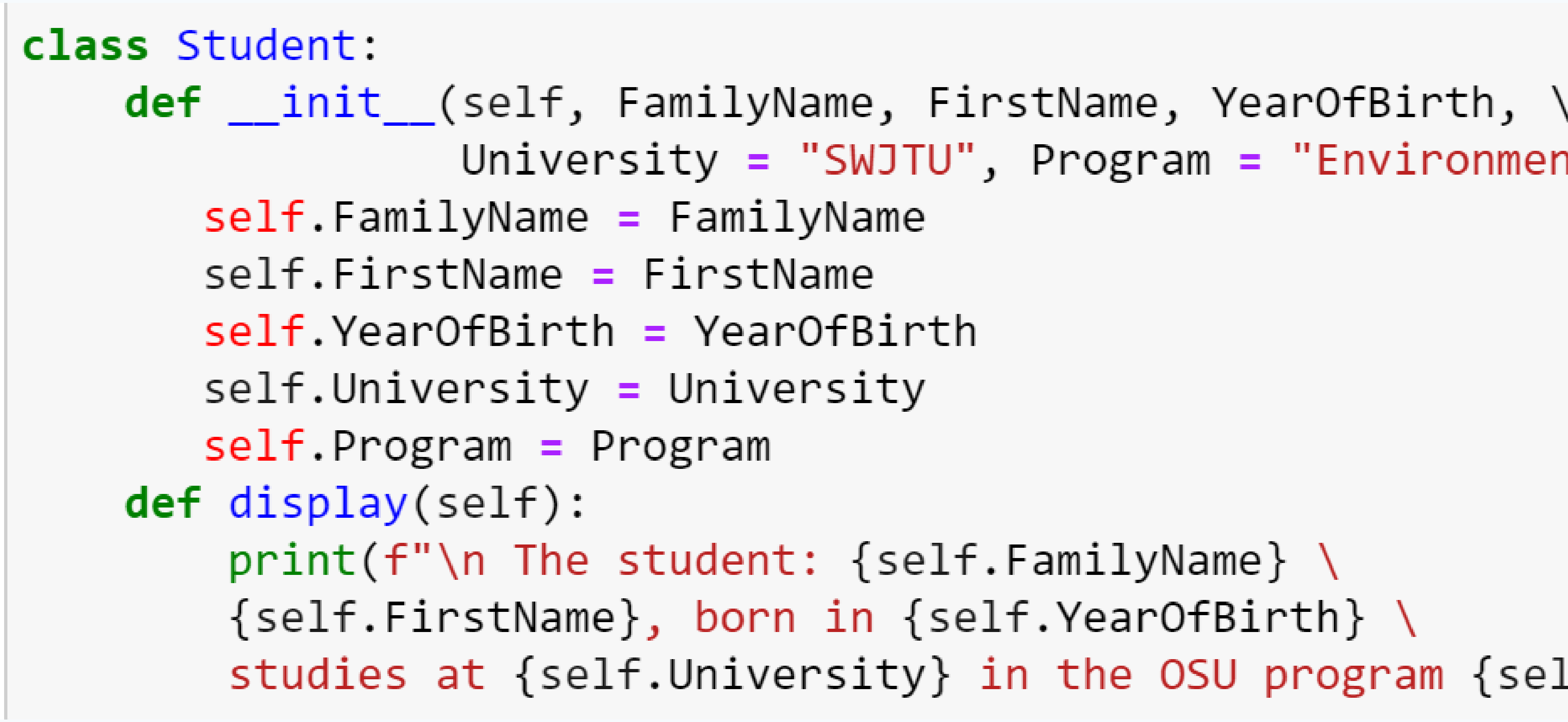
```python
class Bank_Account:
    def __init__(self, FamilyName , FirstName    , YearOfBirth ):
        if YearOfBirth > 2005:
            print(f"{FirstName}, unfortunately you are still too young to open an account, sorry")
            return
        self.Balance= 0
        self.FamilyName = FamilyName
        self.FirstName = FirstName
        self.YearOfBirth = YearOfBirth
        print(f"Hello {self.FirstName} {self.FamilyName} Welcome to the Bank_AccountClass")

lp = Bank_Account("Pluemer", "Lutz", 2010)
```

```
Lutz, unfortunately you are still too young to open an account, sorry
```

Another Class: Student

```
class Student:
    def __init__(self, FamilyName, FirstName, YearOfBirth, \
          University = "SWJTU", Program = "Environmental Engineering") :
      self.FamilyName = FamilyName
      self.FirstName = FirstName
      self.YearOfBirth = YearOfBirth
      self.University = University
      self.Program = Program
    def display(self):
      print(f"\n The student: {self.FamilyName}, {self.FirstName}, born in\
          {self.YearOfBirth} studies at {self.University} in the OSU program\
         {self.Program}")
```

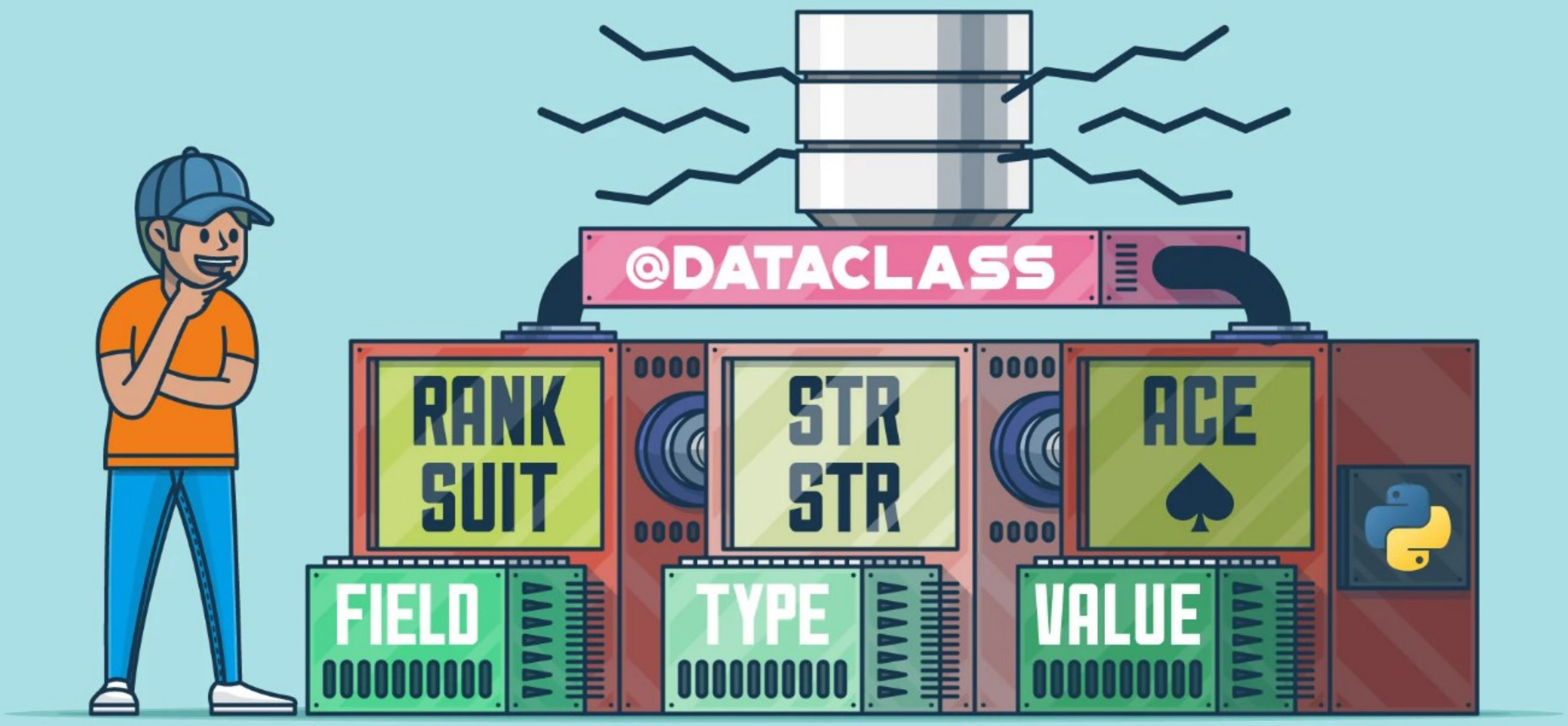
Cropped version



```
class Student:
    def __init__(self, FamilyName, FirstName, YearOfBirth, \
                 University = "SWJTU", Program = "Environmen
       self.FamilyName = FamilyName
       self.FirstName = FirstName
       self.YearOfBirth = YearOfBirth
       self.University = University
       self.Program = Program
    def display(self):
        print(f"\n The student: {self.FamilyName} \
        {self.FirstName}, born in {self.YearOfBirth} \
        studies at {self.University} in the OSU program {sel
```

Full version

```python
class Student:
    def __init__(self, FamilyName, FirstName, YearOfBirth, \
                 University = "SWJTU", Program = "Environmental Engineering") :
       self.FamilyName = FamilyName
       self.FirstName = FirstName
       self.YearOfBirth = YearOfBirth
       self.University = University
       self.Program = Program
    def display(self):
        print(f"\n The student: {self.FamilyName} \
        {self.FirstName}, born in {self.YearOfBirth} \
        studies at {self.University} in the OSU program {self.Program}")
```



Data Class Makes life easier

Real Python

```python
from dataclasses import dataclass

@dataclass
class Student:
    FirstName: str
    FamilyName: str
    University: str
    StudentId: int
    Program: str

st = Student("Lutz","Pluemer","SWJTU",123456789, "Env Eng")
```

What it means

from dataclasses **import** dataclass

This means that we **import** from a predefined module dataclass, more on it next lecture

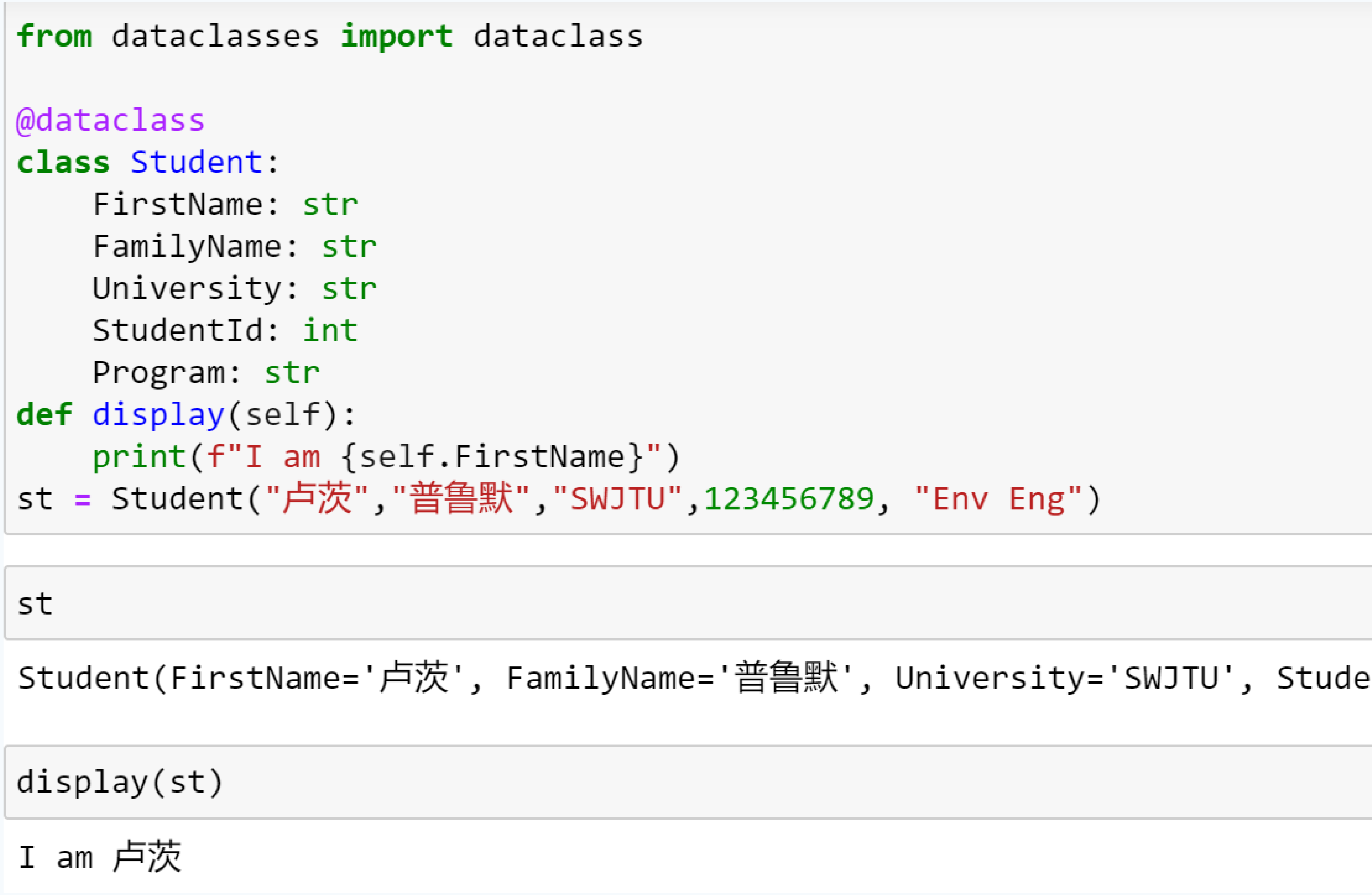
@dataclass

This is a **decorator**, which transforms the program

class Student:

Here we define attributes and their types, __init__ not needed

```
 FirstName: str
 FamilyName: str
 University: str
 StudentId: int
 Program: str
```



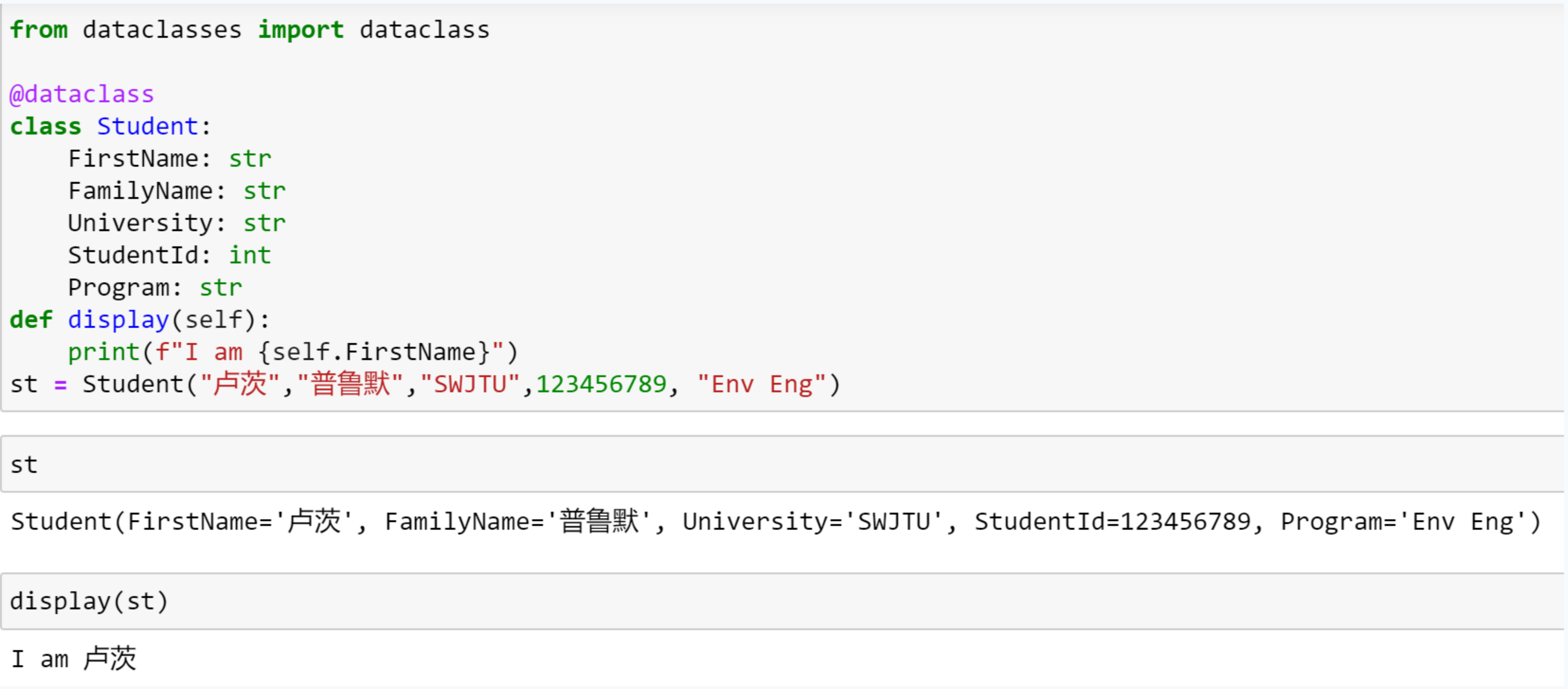
```
from dataclasses import dataclass

@dataclass
class Student:
    FirstName: str
    FamilyName: str
    University: str
    StudentId: int
    Program: str
def display(self):
    print(f"I am {self.FirstName}")
st = Student("卢茨","普鲁默","SWJTU",123456789, "Env Eng")
```

```
st
```

```
Student(FirstName='卢茨', FamilyName='普鲁默', University='SWJTU', Stude
```

```
display(st)
```

```
I am 卢茨
```



```python
from dataclasses import dataclass

@dataclass
class Student:
    FirstName: str
    FamilyName: str
    University: str
    StudentId: int
    Program: str
def display(self):
    print(f"I am {self.FirstName}")
st = Student("卢茨","普鲁默","SWJTU",123456789, "Env Eng")
```

```python
st
```

```
Student(FirstName='卢茨', FamilyName='普鲁默', University='SWJTU', StudentId=123456789, Program='Env Eng')
```

```python
display(st)
```

```
I am 卢茨
```

Class Inheritance

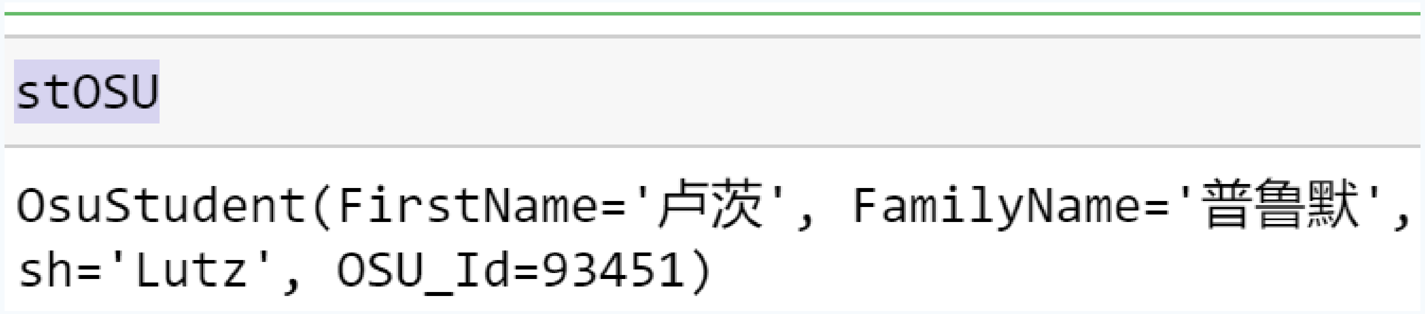
- Class Inheritance is an important concept of object oriented programming
- It means: a class can have subclasses, which inherit attributes and methods from the parent class
- The subclass is also called child class
- The child class can define new attributes and new methods
- But it can also override the methods of its parent

```
@dataclass
class OsuStudent(Student):
    FirstNameEnglish: str
    OSU_Id: int
def display(self):
    print(f"I am {self.FirstNameEnglish}")
stOSU = OsuStudent("卢茨","普鲁默","SWJTU",123456789, \
                   "Env Eng","Lutz",93451)
```

```
display(stOSU)
```

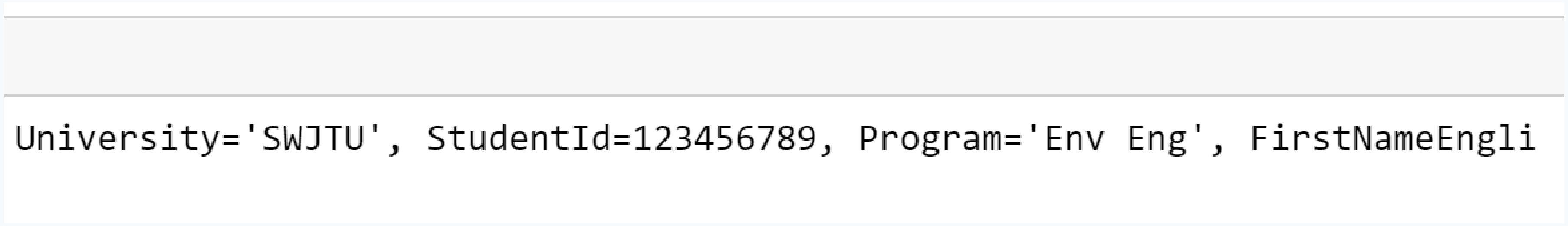
- OsuStudent(Student) means that OsuStudent is **subclass** of Student
- FirstNameEnglish and OSU_Id are new attributes, the others are **inherited**
- display(self) **overwrites** the parent method display
  the difference is that it prints the FirstNameEnglish instead of the FirstName

stOSU has its own and all attributes from its Parent Class



```
stOSU
```

```
OsuStudent(FirstName='卢茨', FamilyName='普鲁默',
sh='Lutz', OSU_Id=93451)
```



```
University='SWJTU', StudentId=123456789, Program='Env Eng', FirstNameEngli
```

Objects and Classes

- This was a very short introduction into classes, objects, sub classes and inheritance
- Just to give you a basic understanding

Classroom Exercise

- Program the Classes BankAccount, Student and OSU Student
- Modify BankAccount, make sure that age is >= 18
- And modify to make sure that deposit is positive
- Implement BankAccount as dataclass

Homework

- Write a sequence of 10 Chinese Surnames and another of 10 Chinese family names. Write a list comprehension which generates all combinations.
- Write a dictionary with student id as key and name as value. Take 10 of your classmates and enter the values stepwise.
- Write a Class Point with two arguments x and y
- Define a class method which gets another point as input and calculates the (Euklidean) distance between self and the other point
(find how to calculate distance if you do not know)
- Define a subclass rectangle with two new parameters length and width
- Write a Method to calculate the area
- Star Exercise: write a method which gets another rectangle as input and checks if the two rectangles overlap. Hint: make a sketch with paper and pencil and look at x and y values separately, use logical operators such as and, or, and not