https://www.halvorsen.blog

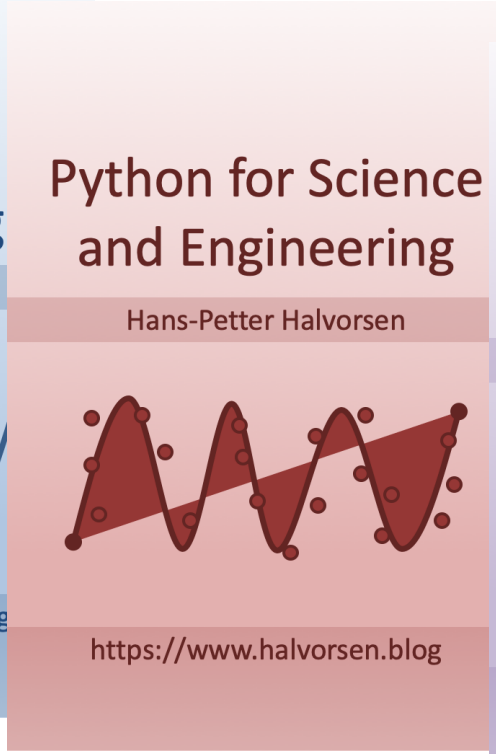
Create Classes in Python

Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

https://www.halvorsen.blog/documents/programming/python/

Additional Python Resources



https://www.halvorsen.blog/documents/programming/python/

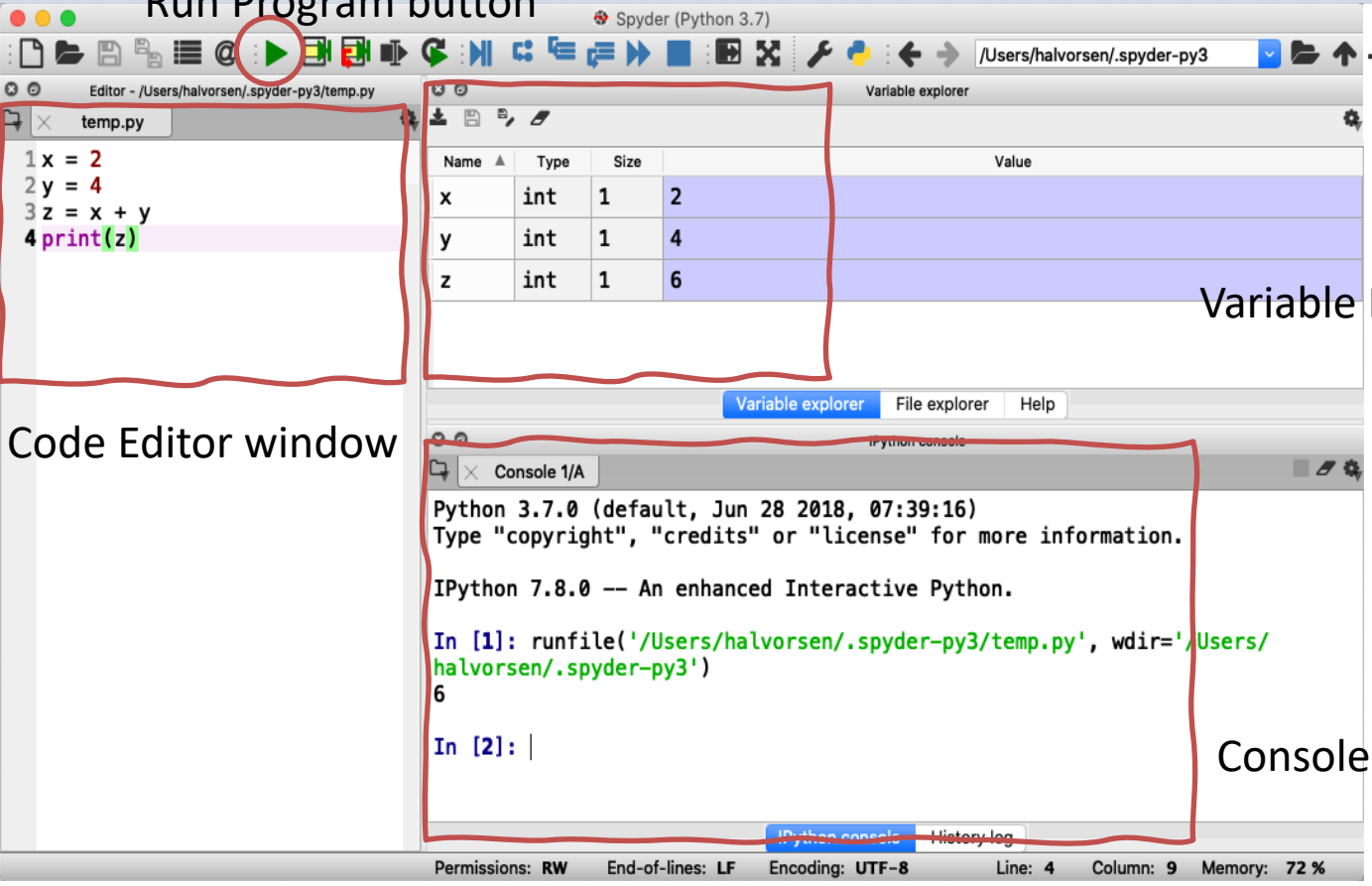
Contents

- Create Classes
- Use Classes
- Create a Module
- Use a Module
- Inheritance

Python Editors

- Python IDLE
- **Spyder** (Anaconda distribution)
- PyCharm
- **Visual Studio Code**
- Visual Studio
- Jupyter Notebook
- …

Spyder (Anaconda distribution)



SPYDER
The Scientific Python Development Environment

Variable Explorer window

Code Editor window

Console window

Object-oriented Programming

- Python is an object-oriented programming (OOP) language.
- Almost everything in Python is an object, with its properties and methods.
- The foundation for all Object-oriented Programming (OOP) languages are Classes.
- Almost all Programming Languages today use Object-oriented Programming techniques.

Create Class

To create a class, use the keyword **class**:

```
class ClassName:
    <statement-1>
    .
    .
    .
    <statement-N>
```

Let's create a simple Class in Python:

```
class Car:
    model = "Volvo"
    color = "Blue"

car = Car()

print(car.model)
print(car.color)
```

Define the Class

Use the Class

Create an **Object** of the Class

The results will be in this case:

```
Volvo
Blue
```

This example don't illustrate the good things with classes so we will create some more examples.

Classes -Example

Let's create the following Python Code:

We start using the Class by creating an Object of that Class

Set **Properties**

```
class Car:
    model = ""
    color = ""

car = Car()

car.model = "Volvo"
car.color = "Blue"

print(car.color + " " + car.model)

car.model = "Ford"
car.color = "Green"

print(car.color + " " + car.model)
```

Define the Class

Use the Class

Classes -Example

Let's create the following Python Code:

```
class Car:
    model = ""
    color = ""

car1 = Car()

car1.model = "Volvo"
car1.color = "Blue"
print(car1.color + " " + car1.model)


car2 = Car()

car2.model = "Ford"
car2.color = "Green"
print(car2.color + " " + car2.model)
```

Define the Class

Use the Class

We can create and use multiple Objects of the same Class

The __init__() Function

In Python all classes have a built-in function called __init__(), which is always executed when the class is being initiated.
In many other OOP languages we call this the **Constructor**.

We will create a simple example where we use the __init__() function to illustrate the principle:

The **self** parameter is a reference to the current instance of the class and is used to access variables that belongs to the class.

```
class Car:
  def __init__(self, model, color):
    self.model = model
    self.color = color

car1 = Car("Ford", "Green")
print(car1.color + " " + car1.model)

car2 = Car("Volvo", "Blue")
print(car2.color + " " + car2.model)
```

The __init__() Function

Let's extend the Class by defining a **Function** as well:

The **self** parameter is a reference to the current instance of the class and is used to access variables that belongs to the class.

Modifying a Property

```
# Defining the Class Car
class Car:
    def __init__(self, model, color):
        self.model = model
        self.color = color

    def displayCar(self):
        print("Car: " + self.model + "-" + self.color)

# Let's start using the Class
car1 = Car("Tesla", "Red")
car1.displayCar()

car2 = Car("Ford", "Green")
car2.displayCar()

car3 = Car("Volvo", "Blue")
car3.displayCar()
car3.color="Black"
car3.displayCar()
```

Classes - Summary

- As you see from the code, we have now defined a Class "Car" that has 2 Class variables called "model" and "color" (these variables are called **Properties**), and in addition we have defined a Function (or Method) called "displayCar()".
- Its normal to use the term "**Method**" for Functions that are defined within a Class.
- You declare class methods like normal functions with the exception that the first argument to each method is **self**.
- To **create instances of a class**, you call the class using class name and pass in whatever arguments its __init__() method accepts, e.g., car1 = Car("Tesla", "Red")

Create Class in separate File

We start by creating the Class and then we save the code in "**Car.py**":

Python Module

```
class Car:
    def __init__(self, model, color):
        self.model = model
        self.color = color

    def displayCar(self):
        print("Car: " + self.model + "-" + s
```

Then we create a Python Script (testCar.py) where we are using the Class:

```
from Car import Car

car1 = Car("Tesla", "Red")
car1.displayCar()

car2 = Car("Ford", "Green")
car2.displayCar()

car3 = Car("Volvo", "Blue")
car3.displayCar()
car3.color="Black"
car3.displayCar()
```

Inheritance

- Another important aspect of Object-oriented Programming is Inheritance
- Inheritance allows us to define a class that inherits all the methods and properties from another class.

Person.py

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def displayPerson(self):
        print("Person: " + self.name + ", " + str(self.age))

class Teacher(Person):
    pass

class Student(Person):
    pass
```

testPerson.py

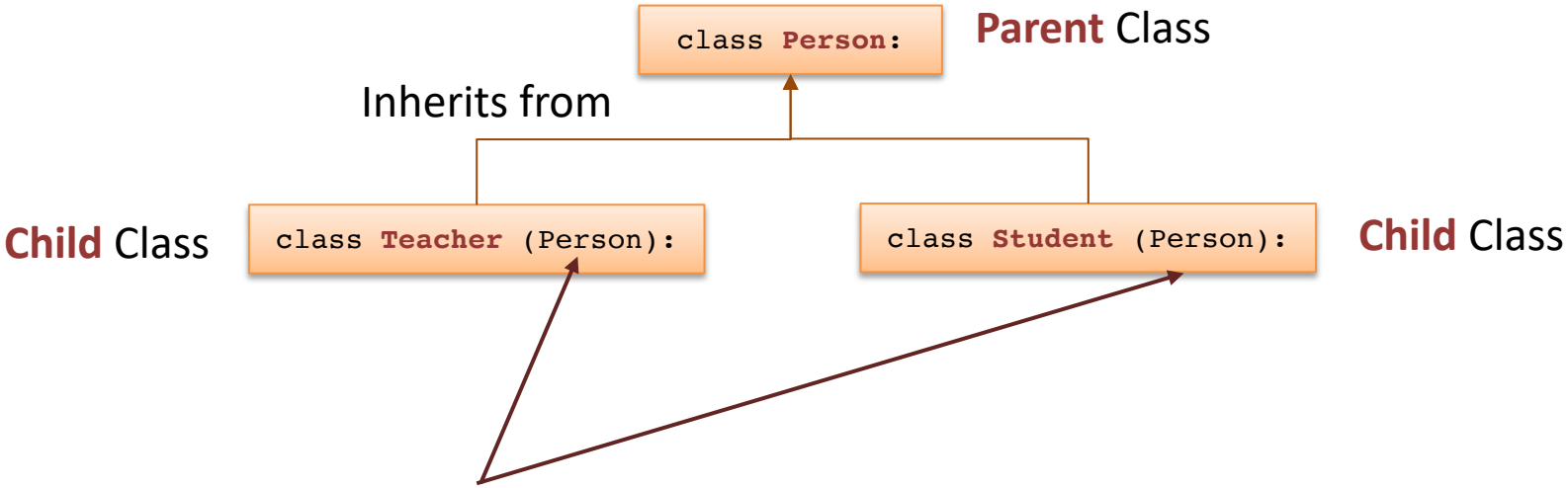
```
from Person import *

teacher1 = Teacher("Knut Nilsen", 44)
teacher1.displayPerson()

student1 = Student("Per Hamsun", 20)
student1.displayPerson()
```

Use the **pass** keyword when you do not want to add any other properties or methods to the class.

Inheritance



Send the parent class as a parameter when creating the child class

Inheritance cont.

Person.py

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def displayPerson(self):
        print("Person: " + self.name + ", " + str(self.age))


class Student(Person):
    def __init__(self, name, age, grade):
        self.name = name
        self.age = age
        self.grade = grade

    def displayPerson(self):
        print("Person: " + self.name + ", " + str(self.age) + ", " + self.grade)


class Teacher(Person):
    def __init__(self, name, age, course):
        self.name = name
        self.age = age
        self.course = course

    def displayPerson(self):
        print("Person: " + self.name + ", " + str(self.age) + ", " + self.course)
```

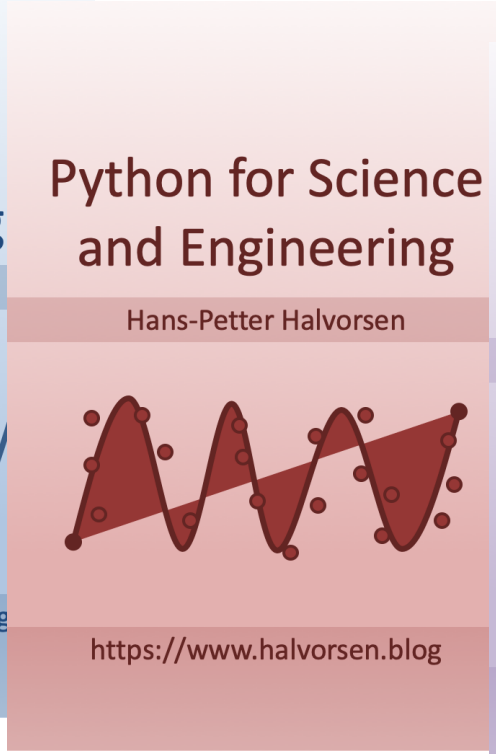
testPerson.py

```
from Person import *

teacher1 = Teacher("Knut Nilsen", 44, "Mathematics")
teacher1.displayPerson()

student1 = Student("Per Hamsun", 20, "B")
student1.displayPerson()
```

Additional Python Resources



https://www.halvorsen.blog/documents/programming/python/

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog