Classes and Objects (1A)

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Python Classes and Objects

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A **Class** is like an object constructor, or a "blueprint" for creating objects.

https://www.w3schools.com/python/python_classes.asp

Creating a class and an object

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

Create a class named MyClass,
with a property named x:

```
class MyClass:
    x = 5
```

Now we can use the class named **MyClass** to create objects:

Create an object named p1, and print the value of x:

```
p1 = MyClass()
print(p1.x)
```

https://www.w3schools.com/python/python_classes.asp

The __init__() Function (1-1)

To understand the meaning of classes
we have to understand the built-in **__init__()** function.

All classes have a function called **__init__()**,
which is always executed when the class is being initiated.

Use the **__init__()** function
to assign values to object properties,
or other operations that are necessary to do
when the object is being created:

https://www.w3schools.com/python/python_classes.asp

The __init__() Function (1-2)

Create a class named **Person**,
use the **__init__()** function
to assign values for **name** and **age**:

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

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

Note: The **__init__()** function is called *automatically*
every time the class is being used to create a new object.

https://www.w3schools.com/python/python_classes.asp

The __str__() Function

The __str__() function controls what should be returned
when the class object is represented as a string.

If the __str__() function is not set,
the string representation of the object is returned:

The string representation of an object
without the __str__() function:

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

p1 = Person("John", 36)

print(p1)
```

```
<__main__.Person object at 0x15039e602100>
```

https://www.w3schools.com/python/python_classes.asp

The __str__() Function

The string representation of an object *with* the __str__() function:

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

    def __str__(self):
        return f"{self.name}({self.age})"

p1 = Person("John", 36)

print(p1)
```

```
John(36)
```

https://www.w3schools.com/python/python_classes.asp

Object Methods

Objects can also contain methods.
Methods in objects are functions that belong to the object.

Insert a function that prints a greeting,
and execute it on the p1 object:

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

    def myfunc(self):
        print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

https://www.w3schools.com/python/python_classes.asp

The **self** parameter

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

It does not have to be named **self**,
you can call it whatever you like,
but it has to be the first parameter of *any* function in the class:

Use the words **xxx** and **abc** instead of **self**:

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

    def myfunc(abc):
        print("Hello my name is " + abc.name)

p1 = Person("John", 36)
p1.myfunc()
```

https://www.w3schools.com/python/python_classes.asp

Modify and Delete Object Properties

Set the **age** of **p1** to 40:

p1.age = 40

Delete the **age** property from the **p1 object**:

del p1.age

Delete the **p1 object**:

del p1

https://www.w3schools.com/python/python_classes.asp

Pass statement

class definitions cannot be empty,

but if you for some reason have

a class definition with no content,

put in the pass statement

to avoid getting an error.

```
class Person:
    pass
```

https://www.w3schools.com/python/python_classes.asp

Python Inheritance

Inheritance allows us to define a class

that inherits all the methods and properties

from another class.

Parent class is the class being inherited from,
also called **base class**.

Child class is the class
that *inherits* from another class,
also called **derived class**.

Any class can be a parent class,
so the syntax is the same as creating any other class:

To create a class that inherits
the functionality from another class,
send the parent class as a parameter
when creating the child class:

https://www.w3schools.com/python/python_classes.asp

Create a parent class

Create a class named **Person**,
with **firstname** and **lastname** properties,
and a **printname** method:

```
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname

    def printname(self):
        print(self.firstname, self.lastname)

#Use the Person class to create an object, and then execute the printname method:

x = Person("John", "Doe")
x.printname()
```

https://www.w3schools.com/python/python_classes.asp

Create a child class

Create a class named **Student**,
which will inherit the properties and methods
from the **Person** class:

```
class Student(Person):
    pass
```

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

Now the **Student** class has
the same properties and methods
as the **Person** class.

Use the **Student** class to create an object,
and then execute the **printname** method:

```
x = Student("Mike", "Olsen")
x.printname()
```

https://www.w3schools.com/python/python_classes.asp

Add the __**init**__() Function

So far we have created a child class
that inherits the properties and methods
from its parent.

We want to add the **__init__()** function
to the child class (instead of the **pass** keyword).

Note: The __init__() function is called automatically
every time the class is being used to create a new object.
Example

```
class Student(Person):
    def __init__(self, fname, lname):
        # add properties etc.
```

When you add the **__init__()** function,
the child class will no longer inherit
the parent's **__init__()** function.

Note: The child's **__init__()** function overrides
the inheritance of the parent's **__init__()** function.

https://www.w3schools.com/python/python_classes.asp

Add the __**init**__() Function

To keep the inheritance of the parent's __**init**__() function,
add a call to the parent's __**init**__() function:

```
class Student(Person):
    def __init__(self, fname, lname):
        Person.__init__(self, fname, lname)
```

Now we have successfully added the __**init**__() function,
and kept the inheritance of the parent class,
and we are ready to add functionality in the __**init**__() function.

https://www.w3schools.com/python/python_classes.asp

Use the **super**() Function

a **super**() function will make the child class
inherit all the methods and properties from its parent:

```
class Student(Person):
    def __init__(self, fname, lname):
        super().__init__(fname, lname)
```

By using the **super**() function,
you do not have to use the name of the parent element,
it will automatically inherit the methods and properties from its parent.

https://www.w3schools.com/python/python_classes.asp

Add Properties

Add a property called **graduationyear** to the **Student** class:

```
class Student(Person):
    def __init__(self, fname, lname):
        super().__init__(fname, lname)
        self.graduationyear = 2019
```

https://www.w3schools.com/python/python_classes.asp

Add Properties

the year 2019 should be a variable,
and passed into the **Student** class
when creating **Student** objects.

To do so, add another parameter in the **__init__**() function:

Add a **year** parameter,
and pass the correct year when creating objects:

```
class Student(Person):
        def __init__(self, fname, lname, year):
                super().__init__(fname, lname)
                self.graduationyear = year

x = Student("Mike", "Olsen", 2019)
```

https://www.w3schools.com/python/python_classes.asp

Add Methods

Add a method called welcome to the **Student** class:

```
class Student(Person):
    def __init__(self, fname, lname, year):
        super().__init__(fname, lname)
        self.graduationyear = year

    def welcome(self):
        print("Welcome",
              self.firstname,
              self.lastname,
              "to the class of",
              self.graduationyear)
```

If you add a method in the child class with the same name
as a function in the parent class,
the inheritance of the parent method will be overridden.

https://www.w3schools.com/python/python_classes.asp

References

[1] Essential C, Nick Parlante
[2] Efficient C Programming, Mark A. Weiss
[3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
[4] C Language Express, I. K. Chun