Python Programming

Jian Zhang

Dec. 07, 2023@PHBS

https://jianzhang.tech/

Python Classes/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.

The __init__() Function

```
class Person:
    def __init__(self, name="Jack", age=34):
        self.name = name
        self.age = age

p1 = Person("John", 36)
print(p1.name)
print(p1.age)

p2 = Person()
print(p2.name)

print(type(p1))
print(type(Person))
```

Object Methods

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

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

p1 = Person("John", 36)
p1.printname()

p2 = Person("Jack", 35)
p2.printname()
```

Object Methods

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

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

    def printname_age(self):
        print("Hello, my name is " + self.name + " and my age is " + str(self.age))

    def print_hello(self):
        print(self)
        print('Hello!')

p3 = Person("PKUSZ", 20)
p3.printname_age()
```

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.

The self Parameter

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

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

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

Modify Object Properties

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

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

p1 = Person("John", 36)
p1.printname()

print(p1.age)
p1.age = 40
print(p1.age)
```

Delete Object Properties/ Objects

```
del p1.age

print(p1.age)

del p1

print(p1)
```

The pass Statement

```
class Person:
    pass

class Person:
```

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.

Create a Child Class

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

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

x = Person("Jian", "Zhang")
x.printname()

class Student(Person):
    pass

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

Add the __init__() Function

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

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

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

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

Add the __init__() Function

```
class Student(Person):
    def __init__(self, fname, lname):
        Person.__init__(self, fname, lname)
        self.firstname = 'Mr. ' + self.firstname

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

Use the super() Function

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

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

Add Properties

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

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

x.printname()

print(x.graduationyear)
```

Add Methods

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

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

x = Person("Jian", "Zhang")
x.printname()
```

Add Methods

```
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)

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

Python Modules

- What is a Module?
- Consider a module to be the same as a code library.
- A file containing a set of functions you want to include in your application.

Create a Module

```
# To create a module just save the code you want in a file with the file extension .py:

def greeting(name):
    print("Hello, " + name)

# Save this code in a file named mymodule.py
```

Use a Module

```
import mymodule

mymodule.greeting("PKUSZ")

mymodule.greeting("PHBS")

# Note: When using a function from a module, use the syntax: module_name.function_name.
```

Variables in Module

```
# Save this code in the file mymodule.py

person1 = {
  "name": "San Zhang",
  "age": 36,
  "country": "China"
}
```

```
import mymodule

a = mymodule.person1["name"]
print(a)

print(mymodule.student_num)
```

Re-naming a Module

```
# You can create an alias when you import a module, by using the as keyword:

import mymodule as mx

a = mx.person1["age"]
print(a)
```

Built-in Modules

```
import platform

x = platform.system()
print(x)

x = dir(platform)
print(x)

import math
print(dir(math))
```

Import From Module

```
# You can choose to import only parts from a module, by using the from keyword.

from mymodule import person1

print(person1["age"])

print(person1["name"])

mymodule.person1["age"]
```

Import From Module

```
from mymodule import *

greeting("PKUSZ")

print(student_num)

def greeting(x):
    print(f"Hi {x}, Welcome to Python Class!")

greeting("PKUSZ")
```

Import From Module

```
from modules.test1 import person2

person2['name’]

from modules import test1

print(test1.person2)
```

Python Datetime

```
import datetime

x = datetime.datetime.now()
print(x)

print(x.strftime("%c"))
```

RegEx in Python

```
# A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

# RegEx can be used to check if a string contains the specified search pattern.

import re

txt = "The rain in Spain"
x = re.findall("ai", txt)
print(x)

txt = "The rain in Spain"
x = re.search("rai", txt)

print("The first white-space character is located in position:", x.start())
```

Homework

Given H1.xls, extract all the hyperlinks into one column, as illustrated in New_H1.xls.

Questions?