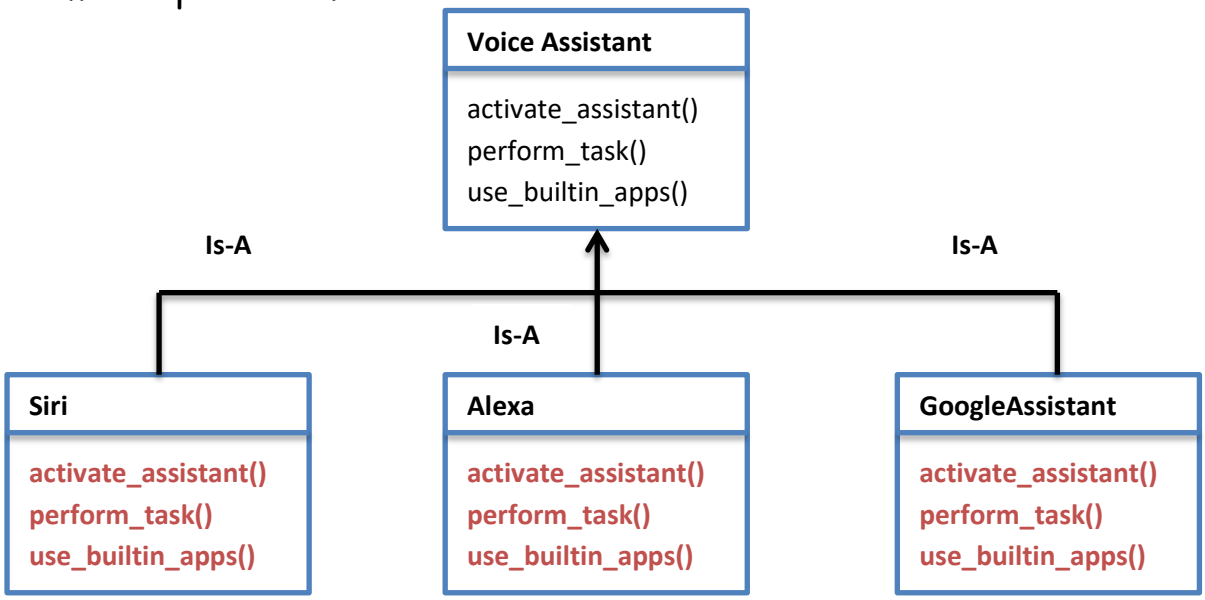
Python Fundamentals day 59

Today's Agenda

- Abstraction
- Rules of abstract class

Abstraction

Let us understand the abstraction by taking an example of voice assistant, we have several voice assistants built-in in our mobile phones like siri in iPhone, google assistant in android phone, alexa in amazon products.



Let us implement the code for the above UML

```
class VoiceAssistant:

    def activate_assistant(self):
        print('VA activated')

    def perform_task(self):
        print('VA is performing the task')

    def use_builtin_apps(self):
        print('VA is using built-in apps')

class Siri(VoiceAssistant):

    def activate_assistant(self):
        print('Hey Siri, activates Siri')

    def perform_task(self):
        print('Siri is performing task using apple servers')

    def use_builtin_apps(self):
        print('Siri uses the builtin apps of ios')

class Alexa(VoiceAssistant):

    def activate_assistant(self):
        print('Alexa, activates Alexa')

    def perform_task(self):
        print('Alexa is performing task using amazon servers')

    def use_builtin_apps(self):
        print('Alexa uses the builtin apps of fire-os')


class GoogleAssistant(VoiceAssistant):

    def activate_assistant(self):
        print('Ok Google, activates GA')

    def perform_task(self):
        print('GA is performing task using google servers')

    def use_builtin_apps(self):
        print('Google uses the builtin apps of android-os')

def main():
    s=Siri()
    a=Alexa()
    ga=GoogleAssistant()

    s.activate_assistant()
    s.perform_task()
    s.use_builtin_apps()

    a.activate_assistant()
    a.perform_task()
    a.use_builtin_apps()

    ga.activate_assistant()
    ga.perform_task()
    ga.use_builtin_apps()

main()
```

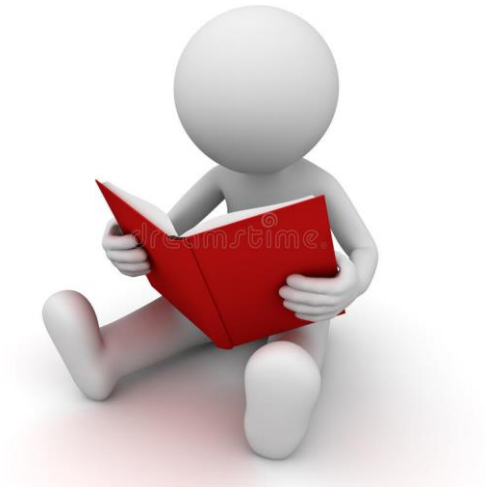
Output:

```
In [1]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Hey Siri, activates Siri
Siri is performing task using apple servers
Siri uses the builtin apps of ios
Alexa, activates Alexa
Alexa is performing task using amazon servers
Alexa uses the builtin apps of fire-os
Ok Google, activates GA
GA is performing task using google servers
Google uses the builtin apps of android-os
```

But in the above code we have just implemented inheritance. We could also implement polymorphism as below

```
class VoiceAssistant:

    def activate_assistant(self):
        print('VA activated')

    def perform_task(self):
        print('VA is performing the task')

    def use_builtin_apps(self):
        print('VA is using built-in apps')

class Siri(VoiceAssistant):

    def activate_assistant(self):
        print('Hey Siri, activates Siri')

    def perform_task(self):
        print('Siri is performing task using apple servers')

    def use_builtin_apps(self):
        print('Siri uses the builtin apps of ios')

class Alexa(VoiceAssistant):

    def activate_assistant(self):
        print('Alexa, activates Alexa')

    def perform_task(self):
        print('Alexa is performing task using amazon servers')

    def use_builtin_apps(self):
        print('Alexa uses the builtin apps of fire-os')
```

```
class GoogleAssistant(VoiceAssistant):

    def activate_assistant(self):
        print('Ok Google, activates GA')

    def perform_task(self):
        print('GA is performing task using google servers')

    def use_builtin_apps(self):
        print('Google uses the builtin apps of android-os')

def use_assistant(ref):
    ref.activate_assistant()
    ref.perform_task()
    ref.use_builtin_apps()

def main():
    s=Siri()
    a=Alexa()
    ga=GoogleAssistant()

    use_assistant(s)
    use_assistant(a)
    use_assistant(ga)

main()
```



Output:

```
In [3]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Hey Siri, activates Siri
Siri is performing task using apple servers
Siri uses the builtin apps of ios
Alexa, activates Alexa
Alexa is performing task using amazon servers
Alexa uses the builtin apps of fire-os
Ok Google, activates GA
GA is performing task using google servers
Google uses the builtin apps of android-os
```

We see in above code, all the inherited or the child classes have modified the inherited method as required. So can we remove the parent class methods and still get the output as expected? And if we

remove methods from parent class, we can change the names of methods in derived classes. Let us see if we can get the same output

```
class VoiceAssistant:
    pass

class Siri(VoiceAssistant):

    def start_assistant(self):
        print('Hey Siri, activates Siri')

    def perform_t(self):
        print('Siri is performing task using apple servers')

    def builtin_apps(self):
        print('Siri uses the builtin apps of ios')

class Alexa(VoiceAssistant):

    def initiate_assistant(self):
        print('Alexa, activates Alexa')

    def do_task(self):
        print('Alexa is performing task using amazon servers')

    def u_b_a(self):
        print('Alexa uses the builtin apps of fire-os')

class GoogleAssistant(VoiceAssistant):

    def activate_assistant(self):
        print('Ok Google, activates GA')

    def perform_task(self):
        print('GA is performing task using google servers')

    def use_builtin_apps(self):
        print('Google uses the builtin apps of android-os')
```

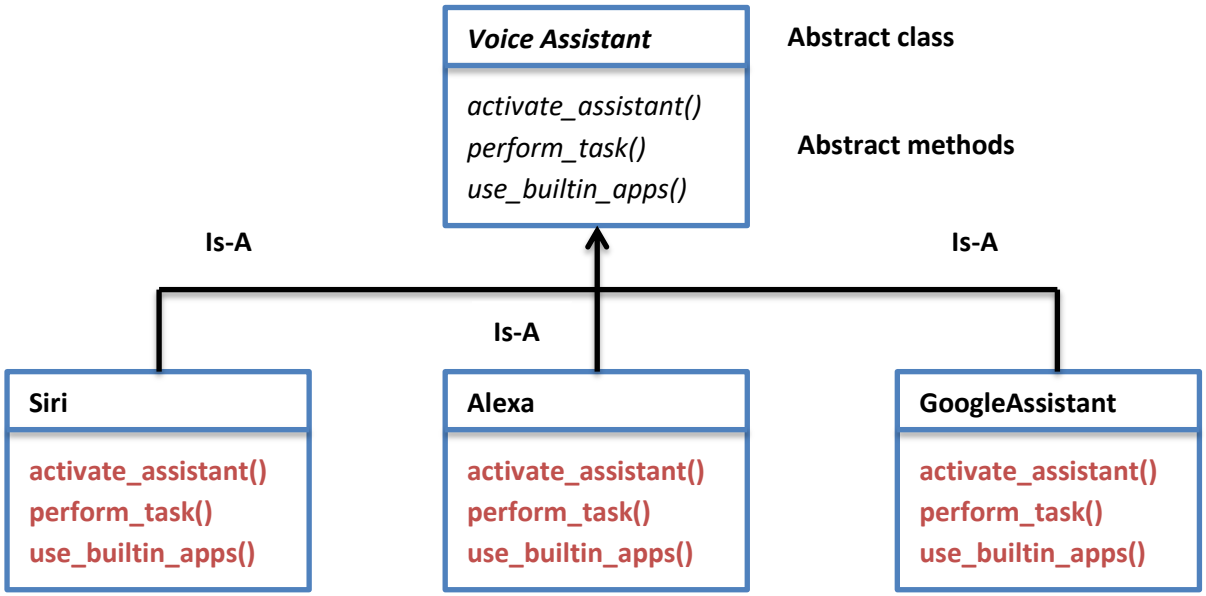
```
def use_assistant(ref):
    ref.activate_assistant()
    ref.perform_task()
    ref.use_builtin_apps()

def main():
    s=Siri()
    a=Alexa()
    ga=GoogleAssistant()

    use_assistant(s)
    use_assistant(a)
    use_assistant(ga)

main()
```

Output:

```
  File "C:/Users/rooman/OneDrive/Desktop/python/test.py",
line 47, in main
    use_assistant(s)

  File "C:/Users/rooman/OneDrive/Desktop/python/test.py",
line 38, in use_assistant
    ref.activate_assistant()

AttributeError: 'Siri' object has no attribute
'activate_assistant'
```

We are getting error because ref was previously pointing to the overridden methods, but now those methods have become specialised methods. So let us undo the changes done and as body is the only thing that is not being used by any of the derived or inherited or child class, let us remove the body of parent class methods.

The methods which contains only the name is called as abstract methods and a class which contains abstract methods are called as abstract class.



```
from abc import ABC,abstractmethod
class VoiceAssistant(ABC): #abstract base class

    @abstractmethod
    def activate_assistant(self):
        pass

    @abstractmethod
    def perform_task(self):
        pass
    @abstractmethod
    def use_builtin_apps(self):
        pass

class Siri(VoiceAssistant):

    def activate_assistant(self):
        print('Hey Siri, activates Siri')

    def perform_task(self):
        print('Siri is performing task using apple servers')

    def use_builtin_apps(self):
        print('Siri uses the builtin apps of ios')

class Alexa(VoiceAssistant):

    def activate_assistant(self):
        print('Alexa, activates Alexa')

    def perform_task(self):
        print('Alexa is performing task using amazon servers')

    def use_builtin_apps(self):
        print('Alexa uses the builtin apps of fire-os')

class GoogleAssistant(VoiceAssistant):

    def activate_assistant(self):
        print('Ok Google, activates GA')

    def perform_task(self):
        print('GA is performing task using google servers')

    def use_builtin_apps(self):
        print('Google uses the builtin apps of android-os')

def use_assistant(ref):
    ref.activate_assistant()
    ref.perform_task()
    ref.use_builtin_apps()

def main():
    s=Siri()
    a=Alexa()
    ga=GoogleAssistant()

    use_assistant(s)
    use_assistant(a)
    use_assistant(ga)

main()
```

Output:

```
In [5]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Hey Siri, activates Siri
Siri is performing task using apple servers
Siri uses the builtin apps of ios
Alexa, activates Alexa
Alexa is performing task using amazon servers
Alexa uses the builtin apps of fire-os
Ok Google, activates GA
GA is performing task using google servers
Google uses the builtin apps of android-os
```

Abstract methods are used when you are not sure about the body of method of the derived classes.

Rules of abstract classes

- An abstract class can contain both concrete as well as abstract methods.

```
from abc import ABC,abstractmethod
class VoiceAssistant(ABC): #abstract base class

    @abstractmethod
    def activate_assistant(self):
        pass

    @abstractmethod
    def perform_task(self):
        pass
    @abstractmethod
    def use_builtin_apps(self):
        pass

    def fun(self): #concrete method
        print('Iside fun()')
```

Output:

```
In [6]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')

In [7]:
```

Certainly we haven't called any method so we won't get anything on output screen. The point is to see if it works or throws error.

- If a class is abstract class, its objects cannot be created.

```
from abc import ABC,abstractmethod
class VoiceAssistant(ABC): #abstract base class

    @abstractmethod
    def activate_assistant(self):
        pass

    @abstractmethod
    def perform_task(self):
        pass
    @abstractmethod
    def use_builtin_apps(self):
        pass

    def fun(self): #concrete method
        print('Iside fun()')

va=VoiceAssistant() #object creation
```

Output:

```
  File "C:/Users/rooman/OneDrive/Desktop/python/test.py",
line 18, in <module>
    va=VoiceAssistant()

TypeError: Can't instantiate abstract class VoiceAssistant
with abstract methods activate_assistant, perform_task,
use_builtin_apps
```

This is because, if the behaviour of object is unknown what is the use of creating such object.

- A child class if inherited from a parent class which is abstract can only create an object if it overrides all the methods inherited from parent class.

```python
from abc import ABC,abstractmethod
class VoiceAssistant(ABC): #abstract base class

    @abstractmethod
    def activate_assistant(self):
        pass

    @abstractmethod
    def perform_task(self):
        pass
    @abstractmethod
    def use_builtin_apps(self):
        pass

    def fun(self): #concrete method
        print('Iside fun()')

class Siri(VoiceAssistant):

    def activate_assistant(self):
        print('Hey Siri')

    def perform_task(self):
        print('Performing task')

    def use_builtin_apps(self):
        print('Using builtin apps')

s=Siri() #object of child class
s.activate_assistant()
s.fun()
s.perform_task()
s.use_builtin_apps()
```

Output:

```
In [8]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Hey Siri
Iside fun()
Performing task
Using builtin apps
```