

Code Introspection and Testing

Session 12

Session Overview

In this session, you will be able to:

- Explain the significance of introspection
- Explain the use of help utility in Python
- Identify the functions and modules used for introspecting built-in Python code
- Identify the functions and attributes used for introspecting Python objects
- Explain Selenium Python bindings API



Introduction to Introspection

The ability to identify the type or properties of objects during execution is referred as introspection.

Provides flexibility and control to the developers.

Python offers great introspection support.



Examining built-in Python Entities Using the Basic Help Utility

Interpreter responds in no time while starting the Python Shell, to inform about the Python version in detail.

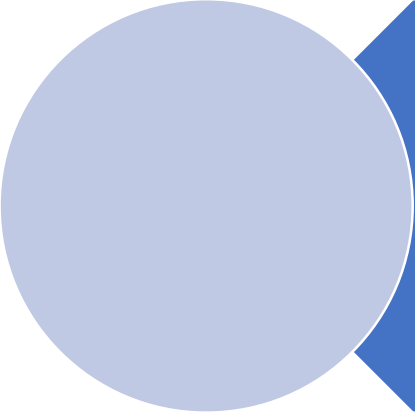
It is obvious to check out which words Python recognizes as keywords, once a developer is at the prompt (`>>>`).

The `help()` function provides more information about any built-in entity.

For example, It is used to know the keywords in Python.



Functions and Modules for Introspecting Built-in Python Code



Code introspection is important in respect to keywords, modules, classes, or objects in order to gain information about them for controlling or manipulating them.



Many built-in functions and utilities are offered by Python for the purpose of introspection.



keyword Module

- Keyword list can be obtained directly from a module in the standard library of Python.

Code Snippet:

```
>>> help('modules keywords')  
Here is a list of modules whose name or  
summary contains 'keywords'.  
If there are any, enter a module name to  
get more help.  
keyword - Keywords (from "graminit.c")
```



sys Module

- How to use the `sys` module in interactive shell?

Code Snippet :

```
>>> import sys
>>> sys.executable
'/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6'
```



dir () Function

- How to use the `dir ()` function to obtain all attributes of `keyword` module?

Code Snippet:

```
>>> dir(keyword)
['__all__', '__builtins__',
 '__cached__', '__doc__', '__file__',
 '__loader__', '__name__',
 '__package__', '__spec__',
 'iskeyword', 'kwlist', 'main']
```

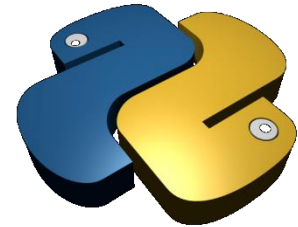


Interrogating Python Objects Using Built-in Attributes and Functions

Each object in computer has few similarities with each other, retaining some of its uniqueness.

For instance, each string in Python has attributes that the `dir()` function reveals.

Thus, in the field of programming, objects are entities having identity and value, some characteristics and are of certain type.



`__name__` Attribute

- Objects that do have names are stored in their `__name__` attribute.
- This name is obtained from the object and not from the variable that refers it.
- While executing a Python module from command prompt, its `__name__` attribute has the `__main__` value instead of actual module name.
- This is how modules are capable of looking at their own `__name__` value.



`type ()` Function

The `type ()` function assists in finding out whether an object is an integer, a string, or of some other type.

It returns a type object, which is comparable to the types existing in `types` module.



id () Function

Each object features a type, an identity, and a value.

More than one variable can refer to the same object. Similarly, variables can refer to objects that look similar, though have isolated, distinct identities.

The object identity is significant while modifying objects, such as removing an item from a list and change a value.



hasattr() and getattr() Functions

If an object consists of the targeted attribute, a developer needs to retrieve it.

Here, `hasattr()` and `getattr()` functions are useful.

`hasattr()` function reveals whether the targeted attribute exists or not, whereas, `getattr()` function retrieves that attribute.



`isinstance ()` Function

- `type ()` function returns the object's type.
- Developer can find object is an instance of a specific type, or a custom class with the help of `isinstance ()` function.



`issubclass()` Function

- To know about the parent classes, the `issubclass()` function is useful.
- User-defined class inherit their attributes from the another class.



Python and Selenium

Binding of Selenium and Python offers a simple Application Programming Interface (API) for coding functional or acceptance tests using the WebDriver version.

Features and functionality of WebDriver can be accessed through API.



Selenium Prerequisites

Selenium is a testing framework that supports Python coding.

Developers can download Python bindings for Selenium from the Python Software Foundation Website,
`https://pypi.python.org/pypi/selenium`.

Alternatively, developers can use the `pip` command for installing Selenium, which is `pip install selenium`.



Comprehending a Basic Selenium Test Script

- Once Selenium Python bindings API is installed, a developer can start coding from Python, as shown in following Code Snippet:

Code Snippet:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
driver = webdriver.Firefox()
driver.get("http://www.python.org")
assert "Python" in driver.title
element = driver.find_element_by_name("q")
element.clear()
element.send_keys("pyton")
element.send_keys(Keys.RETURN)
assert "Not found." Absent in driver.page_source
driver.close()
```

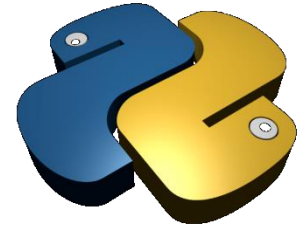


Writing a Python Test Script Using Selenium

The most popular use of Selenium is to code test cases.

However, the installed selenium package itself has no testing framework or tool.

Thus, a developer should use the `unittest` module of Python for coding test cases.



Using Selenium with Remote WebDriver

How to interact with the remote WebDriver?

Code Snippet:

```
from selenium import WebDriver
from selenium.WebDriver.common.desired_capabilities import
DesiredCapabilities
driverObj = WebDriver.Remote(
    cmd_executor='http://127.0.0.1:4444/wd/hub',
    desired_capabilities=DesiredCapabilities.CHROME)
driverObj = WebDriver.Remote(
    cm_executor='http://127.0.0.1:4444/wd/hub',
    desired_capabilities=DesiredCapabilities.HTMLUNITWITHJS)
```



Summary (1-4)

- Introspection is the ability to scrutinize something to find out what it is and what it is capable of doing.
- Code introspection refers to the Python's ability to examine keywords, functions, and classes to find out what they are, what they are aware of, and what they do.
- Introspection helps in identifying the type or properties of objects during execution.



Summary (2-4)

- Python provides the `help()` function to obtain information about any built-in entity.
- The `keyword.kwlist` command reveals a list of Python keywords.
- The `sys` module contains a myriad of variables and functions that expose helpful information about the current interpreter.



Summary (3-4)

- The `dir()` function fetches and returns a sorted list of all names of attributes for any type of object.
- The `__name__` attribute reveals the name of an object.
- As interpreter is the main or top-level module, the local `__name__` variable has the `__main__` value while running Python interactively.
- The `type()` function reveals the type of an object, while the `isinstance()` function whether an object is an instance or not.



Summary (4-4)

- The `hasattr()` and `getattr()` functions allow checking the existence of an attribute and retrieving a specific attribute, respectively.
- Selenium Python bindings offers an API for coding functional or acceptance tests in Python and testing them using Selenium WebDriver.
- Selenium automates the execution of a test script using one of its drivers.

