... continued

Continues the discussion on native coroutines.

Below we present a table that displays the boolean values returned by different utility methods that test for a method to be a coroutine. Additionally, we also present the boolean values returned for **`isinstance()`** method when coroutines are tested to be instances of **`Iterable`**, **`Awaitable`**, **`Generator`** and **`Coroutine`**.

For example, a generator-based coroutine with a decorator tests false as an instance of **`Awaitable`** but the **`inspcet.isawaitable()`** returns true. The same coroutine returns true for **`asyncio.iscoroutine()`** and false for **`inspect.iscoroutine()`**. These differences came into being as the asynchronous programming model evolved in Python.

	Iterable	Awaitable	Generator	CoroutineType	asyncio.iscoroutine()	asyncio.iscoroutinefunction()	inspect.iscoroutine()	inspect.iscoroutinefunction()	inspect.isawaitable()

Ordinary Function with `@asyncio.coroutine`	T	F	T	F	T	T	F	F	T

Ordinary Function with `@types.coroutine`	T	F	T	F	T	F	F	F	T
Simple Generator	T	F	T	F	T	F	F	F	F

Simple Generator with `@asyncio.coroutine`	T	F	T	F	T	T	F	F	T

Simple Generator with `@types.-coroutine`	T	F	T	F	T	F	F	F	T
Generator-based Coroutine	T	F	T	F	T	F	F	F	F

Generator-based Coroutine	T	F	T	F	T	T	F	F	T
with `@asyncio.coroutine`	T	F	T	F	T	T	F	F	T
`async def`	T	F	T	F	T	T	F	F	T

Generator-based Coroutine with `@types.coroutine`	T	F	T	F	T	F	F	F	T
Native Coroutine	F	T	F	T	T	T	T	T	T

> Note the above table scrolls to its right which may not be obvious. All the values presented in the table above can be produced by running the code widget below.

```
import asyncio
import types
import inspect
from collections.abc import Iterable, Awaitable

# Ordinary Function
def ordinary_function():
    pass

# Ordinary Function with @asyncio.coroutine decorator
@asyncio.coroutine
def ordinary_function_with_asyncio_coroutine_dec():
    pass

# Ordinary Function with @types.coroutine decorator
@types.coroutine
def ordinary_function_with_types_coroutine_dec():
    pass

# Simple Generator
def simple_generator():
    assign_me = yield 0

# Simple Generator with @asyncio.coroutine decorator
@asyncio.coroutine
def simple_generator_with_asyncio_coroutine_dec():
    assign_me = yield 0

# Simple Generator with @types.coroutine decorator
@types.coroutine
def simple_generator_with_types_coroutine_dec():
    assign_me = yield 0

# Generator-based coroutine
def generator_based_coroutine():
    yield from asyncio.sleep(1)

# Generator-based coroutine with @asyncio.coroutine decorator
@asyncio.coroutine
def generator_based_coroutine_with_asyncio_coroutine_dec():
    yield from asyncio.sleep(1)

# Generator-based coroutine with @types.coroutine decorator
@types.coroutine
def generator_based_coroutine_with_types_coroutine_dec():
    yield from asyncio.sleep(1)

# Native coroutine
async def native_coroutine():
    pass
```

```
if __name__ == "__main__":
    of_aio_dec = ordinary_function_with_asyncio_coroutine_dec()
    print(of_aio_dec)
    print("simple generator instance of collections.abc.Iterable : " + str(isinstan
    print("simple generator instance of collections.abc.Awaitable : " + str(isinsta
    print("simple generator instance of types.Generator : " + str(isinstance(of_aio
    print("simple generator instance of types.CoroutineType : " + str(isinstance(of
    print("simple generator instance of asyncio.iscoroutine : " + str(asyncio.iscor
    print("simple generator instance of asyncio.iscoroutinefunction : " + str(
        asyncio.iscoroutinefunction(ordinary_function_with_asyncio_coroutine_dec)))
    print("simple generator instance of inspect.iscoroutine : " + str(inspect.iscor
    print("generator instance of inspect.iscoroutinefunction : " + str(
        inspect.iscoroutinefunction(ordinary_function_with_asyncio_coroutine_dec)))
    print("simple generator instance of inspect.isawaitable : " + str(inspect.isawa
    print("\n\n")

    of_types_dec = ordinary_function_with_asyncio_coroutine_dec()
    print(of_types_dec)
    print("simple generator instance of collections.abc.Iterable : " + str(isinstan
    print("simple generator instance of collections.abc.Awaitable : " + str(isinsta
    print("simple generator instance of types.Generator : " + str(isinstance(of_typ
    print("simple generator instance of types.CoroutineType : " + str(isinstance(of
    print("simple generator instance of asyncio.iscoroutine : " + str(asyncio.iscor
    print("simple generator instance of asyncio.iscoroutinefunction : " + str(
        asyncio.iscoroutinefunction(ordinary_function_with_types_coroutine_dec)))
    print("simple generator instance of inspect.iscoroutine : " + str(inspect.iscor
    print("generator instance of inspect.iscoroutinefunction : " + str(
        inspect.iscoroutinefunction(ordinary_function_with_types_coroutine_dec)))
    print("simple generator instance of inspect.isawaitable : " + str(inspect.isawa
    print("\n\n")


    sg = simple_generator()
    print(sg)
    print("simple generator instance of collections.abc.Iterable : " + str(isinstan
    print("simple generator instance of collections.abc.Awaitable : " + str(isinsta
    print("simple generator instance of types.Generator : " + str(isinstance(sg, ty
    print("simple generator instance of types.CoroutineType : " + str(isinstance(sg
    print("simple generator instance of asyncio.iscoroutine : " + str(asyncio.iscor
    print("simple generator instance of asyncio.iscoroutinefunction : " + str(
        asyncio.iscoroutinefunction(simple_generator)))
    print("simple generator instance of inspect.iscoroutine : " + str(inspect.iscor
    print("generator instance of inspect.iscoroutinefunction : " + str(
        inspect.iscoroutinefunction(simple_generator)))
    print("simple generator instance of inspect.isawaitable : " + str(inspect.isawa
    print("\n\n")

    sg_aio_dec = simple_generator_with_asyncio_coroutine_dec()
    print(sg_aio_dec)
    print("simple generator instance of collections.abc.Iterable : " + str(isinstan
    print("simple generator instance of collections.abc.Awaitable : " + str(isinsta
    print("simple generator instance of types.Generator : " + str(isinstance(sg_aio
    print("simple generator instance of types.CoroutineType : " + str(isinstance(sg
```

```
print("simple generator instance of types.CoroutineType : " + str(isinstance(sg
print("simple generator instance of asyncio.iscoroutine : " + str(asyncio.iscor
print("simple generator instance of asyncio.iscoroutinefunction : " + str(
    asyncio.iscoroutinefunction(simple_generator_with_asyncio_coroutine_dec)))
print("simple generator instance of inspect.iscoroutine : " + str(inspect.iscor
print("generator instance of inspect.iscoroutinefunction : " + str(
    inspect.iscoroutinefunction(simple_generator_with_asyncio_coroutine_dec)))
print("simple generator instance of inspect.isawaitable : " + str(inspect.isawa
print("\n\n")

sg_types_dec = simple_generator_with_types_coroutine_dec()
print(sg_types_dec)
print("simple generator instance of collections.abc.Iterable : " + str(isinstan
print("simple generator instance of collections.abc.Awaitable : " + str(isinsta
print("simple generator instance of types.Generator : " + str(isinstance(sg_typ
print("simple generator instance of types.CoroutineType : " + str(isinstance(sg
print("simple generator instance of asyncio.iscoroutine : " + str(asyncio.iscor
print("simple generator instance of asyncio.iscoroutinefunction : " + str(
    asyncio.iscoroutinefunction(simple_generator_with_types_coroutine_dec)))
print("simple generator instance of inspect.iscoroutine : " + str(inspect.iscor
print("generator instance of inspect.iscoroutinefunction : " + str(
    inspect.iscoroutinefunction(simple_generator_with_types_coroutine_dec)))
print("simple generator instance of inspect.isawaitable : " + str(inspect.isawa
print("\n\n")

gbc = generator_based_coroutine()
print(gbc)
print("generator instance of collections.abc.Iterable : " + str(isinstance(gbc,
print("generator instance of collections.abc.Awaitable : " + str(isinstance(gbc
print("generator instance of types.Generator : " + str(isinstance(gbc, types.Ge
print("generator instance of types.CoroutineType : " + str(isinstance(gbc, type
print("generator instance of asyncio.iscoroutine : " + str(asyncio.iscoroutine(
print("generator instance of asyncio.iscoroutinefunction : " + str(
    asyncio.iscoroutinefunction(generator_based_coroutine)))
print("generator instance of inspect.iscoroutine : " + str(inspect.iscoroutine(
print("generator instance of inspect.iscoroutinefunction : " + str(
    inspect.iscoroutinefunction(generator_based_coroutine)))
print("generator instance of inspect.isawaitable : " + str(inspect.isawaitable(
print("\n\n")

gbc_aio_dec = generator_based_coroutine_with_asyncio_coroutine_dec()
print(gbc_aio_dec)
print("generator instance of collections.abc.Iterable : " + str(isinstance(gbc_
print("generator instance of collections.abc.Awaitable : " + str(isinstance(gbc
print("generator instance of types.Generator : " + str(isinstance(gbc_aio_dec,
print("generator instance of types.CoroutineType : " + str(isinstance(gbc_aio_d
print("generator instance of asyncio.iscoroutine : " + str(asyncio.iscoroutine(
print("generator instance of asyncio.iscoroutinefunction : " + str(
    asyncio.iscoroutinefunction(generator_based_coroutine_with_asyncio_coroutin
print("generator instance of inspect.iscoroutine : " + str(inspect.iscoroutine(
print("generator instance of inspect.iscoroutinefunction : " + str(
    inspect.iscoroutinefunction(generator_based_coroutine_with_asyncio_coroutin
print("generator instance of inspect.isawaitable : " + str(inspect.isawaitable(
print("\n\n")
```

```
gbc_types_dec = generator_based_coroutine_with_types_coroutine_dec()
print(gbc_types_dec)

print("generator instance of collections.abc.Iterable : " + str(isinstance(gbc_
print("generator instance of collections.abc.Awaitable : " + str(isinstance(gbc
print("generator instance of types.Generator : " + str(isinstance(gbc_types_dec
print("generator instance of types.CoroutineType : " + str(isinstance(gbc_types
print("generator instance of asyncio.iscoroutine : " + str(asyncio.iscoroutine(
print("generator instance of asyncio.iscoroutinefunction : " + str(
    asyncio.iscoroutinefunction(generator_based_coroutine_with_types_coroutine_
print("generator instance of inspect.iscoroutine : " + str(inspect.iscoroutine(
print("generator instance of inspect.iscoroutinefunction : " + str(
    inspect.iscoroutinefunction(generator_based_coroutine_with_types_coroutine_
print("generator instance of inspect.isawaitable : " + str(inspect.isawaitable(
print("\n\n")

nc = native_coroutine()
print("native coro instance of collections.abc.Iterable : " + str(isinstance(nc
print("native coro instance of collections.abc.Awaitable : " + str(isinstance(n
print("native coro instance of types.Generator : " + str(isinstance(nc, types.G
print("native coro instance of types.CoroutineType : " + str(isinstance(nc, typ
print("native coro instance of asyncio.iscoroutine : " + str(asyncio.iscoroutin
print("native coro instance of asyncio.iscoroutinefunction : " + str(asyncio.is
print("native coro instance of inspect.iscoroutine : " + str(inspect.iscoroutin
print("generator instance of inspect.iscoroutinefunction : " + str(
    inspect.iscoroutinefunction(native_coroutine)))
print("native coro instance of inspect.isawaitable : " + str(inspect.isawaitabl
print(nc)
print("\n\n")
```

Back

Native Coroutines

Next

Mixing Native & Generator Based Cor...

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/-continued__asyncio__python-concurrency-for-senior-engineering-interviews)