Python Training - Part 28

Note: these quiz questions are notes on the video available at: https://www.youtube.com/watch?v=snzi8jRMuq4

- 1. What is a **good example** of using a built-in function? [06:19]
- 2. Fill in the gaps: Here we are using a total of ____ built-in functions in our example. [07:15]
- 3. What have "they written in bold"? [17:10]
- 4. "When awaited, return the next item from the given..." what? [17:25]
- 5. Can you name some built-in functions whose name begins with a? [18:19]
- 6. What is an async iterator? [20:09]
- 7. An **async iterator** works with the... what? [21:22]
- 8. We look at an example of some code. Three arguments are passed into **def**__init__. What are they? [21:36]
- 9. What happens on the next two lines? [21:50]
- 10. Consider the special term **def** in Python code. Often, this is the first word on a line. Have you ever come across a case in which it is preceded by something?
- 11. What do we assign to self.current? [22:14]
- 12. What do we do if **self.current** is bigger than (or equal to) **self.high**? [22.35]
- 13. What do we assign to the word value? [22:42]
- 14. What is the meaning of the word "awaitable"? [23:43]
- 15. Sometimes we see words which have a double underscore on either side. Can you think of all the examples that we encounter in the video?

- 16. How does anext() fit in?
- 17. What do we assign to **counter**? [24:30]
- 18. Why?
- 19. What do we import? [24:30]
- 20. What do we describe as a shortcut?

ANSWERS

1. What is a good example of using a built-in function? [06:19]

```
destination_name = "Venkatanarasimharajuvaripeta"

# Built-in function: len()
length_of_destination = len(destination_name)

# Built-in function: print()
print(length_of_destination)
```

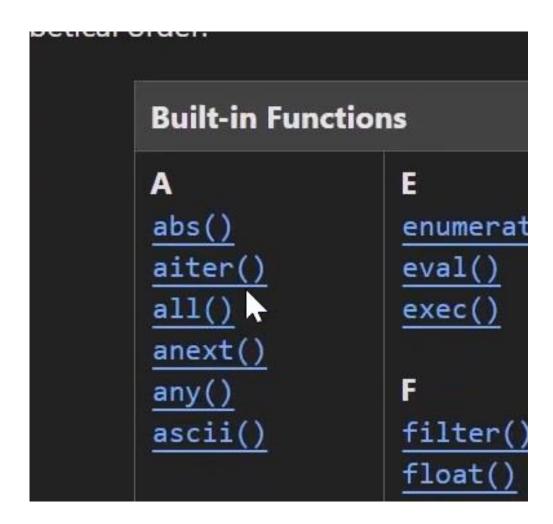
2. Fill in the gaps: Here we are using a total of ____ built-in functions in our example.

Here we are using a total of **two** built-in functions in our example.

- 3. What have "they written in bold"? [17:10]
 - anext
- 4. "When awaited, return the next item from the given..." what? [17:25]

When awaited, return the next item from the given asynchronous iterator.

5. Can you name some built-in functions whose name begins with a? [18:19]



6. What is an async iterator? [20:09]

In Python, an iterator is an object you can loop over (like a list or a generator).

Normally, you'd use the built-in next() function to get the next item from an iterator.

But async iterators are slightly different – they are used in asynchronous code (code that doesn't block your program while waiting for something, like downloading data from the internet).

7. An async iterator works with the... what? [21:22]

"An async iterator works with the async for loop"

8. We look at an example of some code. Three arguments are passed into def __init__. What are they? [21:36]

```
def __init__(self, low, high):
```

9. What happens on the next two lines? [21:50]

```
class AsyncCounter:
    def __init__(self, low, high):
        self.low = low
        self.high = high
```

10. Consider the special term **def** in Python code. Often, this is the first word on a line. Have you ever come across a case in which it is preceded by something?

Yes! Take a look at this:

```
async def __aiter__(self):
    self.current = self.low
    return self
```

11.	What do we assign to self.current?
	Self.current = self.low
12.	What do we do if self.current is bigger than (or equal to) self.high? [22.35] raise stopAsyncIteration
13.	What do we assign to the word value? value = self.current
14.	What is the meaning of the word "awaitable"? [23:43] The key word awaitable means that the result of calling anext() is something you must await.
15.	Sometimes we see words which have a double underscore on either side. Can you think of all the examples that we encounter in the video? init [21:37] aiter [22:04] anext [22:29]

16. How does anext() fit in?

The anext() function is a shortcut to get the next value from an async iterator.

Instead of writing this:

```
python

async for item in AsyncCounter(1, 4):

print(item)
```

You could manually get the next value like this:

```
import asyncio

async def main():
    counter = AsyncCounter(1, 4)
    iterator = await counter.__aiter__()
    print(await anext(iterator)) # 1
    print(await anext(iterator)) # 2
    print(await anext(iterator)) # 3

asyncio.run(main())
```

17. What do we assign to counter?

```
AsyncCounter(1, 4)
```

- 18. Why?
- 19. What do we import? [24:30]

asyncio

20. What do we describe as a shortcut?

The anext() function