# Iterators and generators

Total points 7/8

Theory test for iterators and generators in python.

✓ What is an iterator protocol in Python? * 1/1

- ○ A protocol for sending data between two computers
- ○ A protocol for transferring files over a network
- ○ A protocol for handling exceptions in Python
- ◉ A protocol that defines how objects can be iterated or looped over ✓

✓ To implement the iterator protocol in Python you have to overload the methods: *1/1

- ○ __next__ and __getitem__
- ○ __getitem__ and __iter__
- ◉ __next__ and __iter__ ✓
- ○ __setitem__ and __iter__

✓ Which of the following is true about iterators in Python? * 1/1

- (●) An iterator is an object that can be iterated over ✓
- ( ) An iterator is a function that returns an iterable object
- ( ) An iterator is a method that modifies an iterable object
- ( ) An iterator is a class that defines how objects can be iterated over

✓ What is a generator expression in Python? * 1/1

- ( ) A function that returns an iterator
- (●) An expression that generates a sequence of values lazily ✓
- ( ) An expression that generates random numbers
- ( ) An expression that returns a string

✓ What is the difference between an iterator and a generator in Python? * 1/1

- ( ) An iterator is a function that returns an iterable object, while a generator is an object that can be iterated over
- ( ) An iterator generates a sequence of values lazily, while a generator returns an iterator
- (●) An iterator is an object that can be iterated over, while a generator is a special type of iterator that is defined using a function with the yield keyword ✓
- ( ) An iterator is a class that defines how objects can be iterated over, while a generator is a function that returns an iterable object

✓ What is the purpose of the yield keyword in a generator function in Python? *1/1

- (●) It returns a value from a function and suspends the execution of the function ✓
- ( ) It terminates the execution of a function and returns a value
- ( ) It raises an exception in a function
- ( ) It defines the start of a loop in a function

✓ Which of the following is an example of an iterable in Python? * 1/1

- ( ) A list
- ( ) A dictionary
- ( ) A string
- (●) All of the above ✓

✗ What is the difference between a list comprehension and a generator expression in Python? *0/1

- ○ A list comprehension generates a sequence of values eagerly, while a generator expression generates a sequence of values lazily
- ○ A list comprehension uses the yield keyword, while a generator expression uses the return keyword
- ◉ A list comprehension can only be used to generate lists, while a generator expression can be used to generate any iterable object ✗
- ○ None of the above

Correct answer

- ◉ A list comprehension generates a sequence of values eagerly, while a generator expression generates a sequence of values lazily