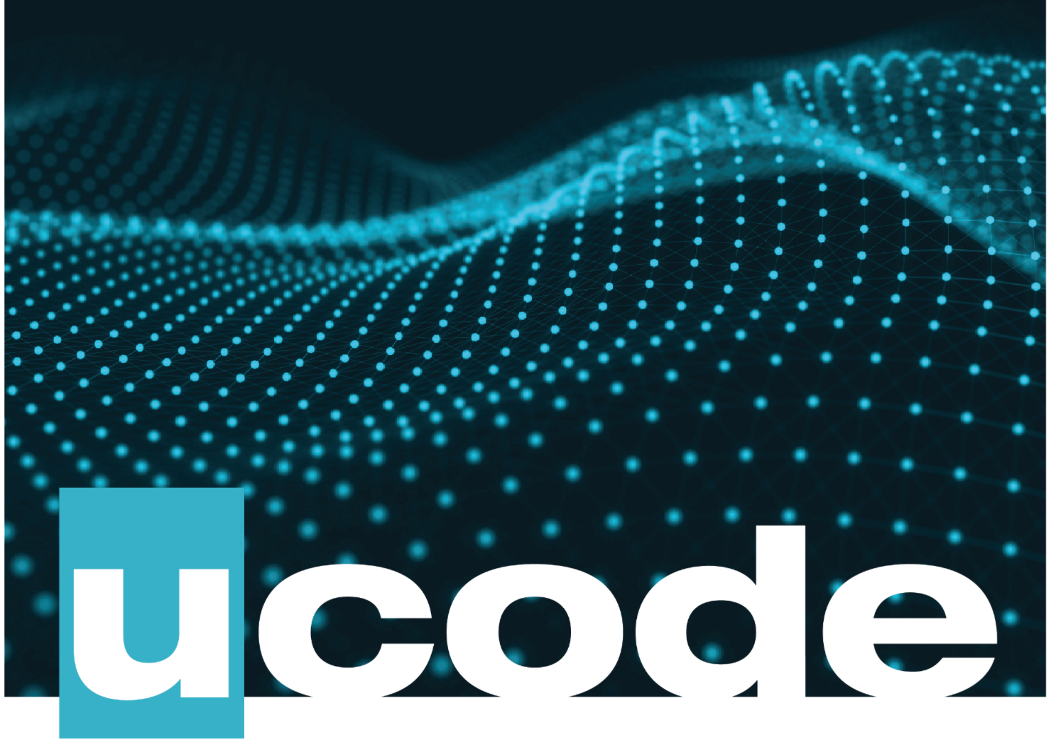
Sprint 02

Marathon Python

April 19, 2021



Contents

ucode

Engage

DESCRIPTION

Let's get started with Python data structures

Let's get started with Python data structures!

Data structures are a fundamental construct in programming. Each data structure provides a particular way of organizing data so it can be accessed efficiently, depending on use case.

For example, phone books make a decent real-world analog for dictionary objects. They allow you to quickly retrieve the information (phone number) associated with a given key (a person's name). Instead of having to read a phone book front to back to find someone's number, you can jump more or less directly to a name and look up the associated information.

Data structures provide us with a specific way of storing and organizing data, such that it can be easily accessed and worked with efficiently. There are quite a few data structures available. The Python built-in data structures are: lists, tuples, dictionaries, sets.

BIG IDEA

Data structures.

ESSENTIAL QUESTION

What are the ways to store data in Python?

CHALLENGE

Explore data structures and their appropriate uses.

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What is a data structure?
- What data structures exist in Python?
- What are the most popular sorting algorithms?
- What is a regular expression in Python?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Read about Python built-in datatypes.
- Find information about sorting and sorting algorithms.
- Learn about regular expressions and how to use them in Python.
- In this online regex visualizer, build a regex that recognizes all words that start with an uppercase letter.
- Attentively watch and investigate learning videos available on the challenge page. Try to repeat all actions.
- Clone your git repository issued on the challenge page in the LMS.
- Proceed with tasks.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story.
- All tasks are divided into Act Basic and Act Advanced. You need to complete all basic tasks to validate the Sprint. But to achieve maximum points, consider accomplishing advanced tasks also.
- Analyze all information you have collected during the preparation stages. Try to define the order of your actions.
- Perform only those tasks that are given in this document.
- Submit only those files that are described in the story. Only useful files allowed, garbage shall not pass!
- Run the scripts using `python3`.
- Make sure that you have `Python` with a `3.8` version, or higher.

- Use the standard library available after installing `Python`. You may use additional packages/libraries that were not previously installed only if they are specified in the task.
- To figure out what went wrong in your code, use PEP 553 -- Built-in breakpoint().
- Complete tasks according to the rules specified in the PEP8 conventions.
- The solution will be checked and graded by students like you. Peer-to-Peer learning.
- Also, the challenge will pass automatic evaluation which is called `Oracle`.
- If you have any questions or don't understand something, ask other students or just Google it.

ucode

Sprint 02 | Marathon Python > 4

Act Basic: Task 00

NAME

Tuple

DIRECTORY

`t00_tuple/`

SUBMIT

`song.py`

BEFORE YOU BEGIN

In order to complete this task, you must be able to answer the following questions:

- What is a `tuple` in Python?
- How is a tuple different from a list?
- What is the advantage of using a tuple over a list?
- How to create a tuple with just one element?
- How to add tuples?
- How to multiply tuples?

DESCRIPTION

Create a script with a function `song()` that generates a song from given verses and chorus. It takes two tuple arguments: `verses` and `chorus`, and returns a tuple of strings. The `verses` is a tuple of tuples of strings, and the `chorus` is a tuple of strings. Each string is meant to be a line of a song.

The formula for a song that your function must use is: add a chorus after every verse, and add one more chorus in the end of the song. So, for example, if the given arguments are:

- `verses` : `(('a', 'b'), ('c', 'd'))`
- `chorus` : `('x', 'y')`

Then the function must return `('a', 'b', 'x', 'y', 'c', 'd', 'x', 'y', 'x', 'y')`. You can see this example in the PYTHON INTERPRETER.

The script in the EXAMPLE tests your function. If everything is correct, it should generate output as seen in the CONSOLE VIEW. Pay attention that you must only submit the file song.py, not the test script.

You don't have to manage test cases where the incoming arguments are not tuples.

EXAMPLE

```
from song import song


simple_verses = (('verse1 line1', 'verse1 line2'),
                 ('verse2 line1', 'verse2 line2', 'verse2 line3'),
                 ('verse3 line1',))
simple_chorus = ('** CHORUS line1', '** CHORUS line2')
love_verses = (("There's nothing you can do that can't be done",
                "Nothing you can sing that can't be sung",
                'Nothing you can say but you can learn how to play the game',
                "It's easy\n"),
               ("Nothing you can know that isn't known",
                "Nothing you can see that isn't shown",
                "Nowhere you can be that isn't where you're meant to be",
                "It's easy\n"))
love_chorus = ('All you need is love', 'All you need is love',
               'All you need is love, love', 'Love is all you need\n')


def print_test_case(verses, chorus, test_description):
    print('\n' + f'test - {test_description}'.center(40, '_'))
    print('[start]', *song(verses, chorus), '[end]', sep='\n')


if __name__ == '__main__':
    print_test_case(simple_verses, simple_chorus, 'simple')
    print_test_case(love_verses, love_chorus, 'normal song')
    print_test_case((tuple(), ('verse2 line1', 'verse2 line2')),
                    simple_chorus, 'one verse = empty tuple')
    print_test_case(tuple(), simple_chorus, 'verses = empty tuple')
    print_test_case(simple_verses, tuple(), 'chorus = empty tuple')
    print_test_case(tuple(), tuple(), 'both arguments empty')
```

Sprint 02 | Marathon Python **> 6**

CONSOLE VIEW

```
>python3 s02t00_tuple_main.py

_____________test - simple______________
[start]
verse1 line1
verse1 line2
** CHORUS line1
** CHORUS line2
verse2 line1
verse2 line2
verse2 line3
** CHORUS line1
** CHORUS line2
verse3 line1
** CHORUS line1
** CHORUS line2
** CHORUS line1
** CHORUS line2
[end]

___________test - normal song___________
[start]
There's nothing you can do that can't be done
Nothing you can sing that can't be sung
Nothing you can say but you can learn how to play the game
It's easy

All you need is love
All you need is love
All you need is love, love
Love is all you need

Nothing you can know that isn't known
Nothing you can see that isn't shown
Nowhere you can be that isn't where you're meant to be
It's easy

All you need is love
All you need is love
All you need is love, love
Love is all you need

All you need is love
All you need is love
All you need is love, love
Love is all you need

[end]
```

```
_____test - one verse = empty tuple_____
[start]
** CHORUS line1
** CHORUS line2
verse2 line1
verse2 line2
** CHORUS line1
** CHORUS line2
** CHORUS line1
** CHORUS line2
[end]

______test - verses = empty tuple_______
[start]
** CHORUS line1
** CHORUS line2
[end]

______test - chorus = empty tuple_______
[start]
verse1 line1
verse1 line2
verse2 line1
verse2 line2
verse2 line3
verse3 line1
[end]

______test - both arguments empty_______
[start]
[end]
>
```

PYTHON INTERPRETER

```
>python3
>>> from song import song
>>> verses = (('a', 'b'), ('c', 'd'))
>>> chorus = ('x', 'y')
>>> song(verses, chorus)
('a', 'b', 'x', 'y', 'c', 'd', 'x', 'y', 'x', 'y')
>>>
```

SEE ALSO

Understanding Tuples in Python 3