Docstring documents
Docstrings

Start of transcript. Skip to the end.

>> A vital component to

any good Python program is documentation.

In this section, we'll be

exploring how you can use Docstrings with

triple quotes to add

definition and details to all of your functions.

>> Let's look at how we can create and access

0:00 / 4:46 1.0x

Video
Download video file

Transcripts
Download SubRip (.srt) file
Download Text (.txt) file

Concepts

Documenting code is an essential part of software development. It is the way developers communicate ideas to users or other developers that will be using the code. Documenting code in Python is easily done using documentation strings or docstrings. Docstrings are just string literals that are used to document modules, classes, functions, and methods. Docstrings are defined as a string between triple quotes `""" some string """`. In this section the focus is on using docstrings to document functions.

In Python, documentation should communicate what your code does not how it works. There are different documentation conventions for every development community or company, the following is adapted from Python Enhancement Proposal (PEP 257), which can be found at https://www.python.org/dev/peps/pep-0257/

One-line docstrings

Used for simple functions with clear functionality:

```
def double(x):
    """Return doubled x."""
    return x * 2
```

General notes:

- Use triple quotes even though it's only a single line docstring.
- Describe the function as a command not as a description (i.e. return x, compute b...).
- The string should end with a period.

Multi-line docstrings

Used to describe functions more elaborately:

```
def vowel_count(word):
    """
    Count the number of vowels in word.

    args:
        word: string under test

    returns:
        count: number of counted vowels
    """

    count = 0
    for c in word:
        if c in "AEIOUYaeiouy":
            count = count + 1
    return count
```

General notes:

- Write a summary line like in the one-line docstring (as a command and ends with a period).
- Follow the summary by a blank line.
- You can write more description after the blank line, but this is optional.
- list the function arguments, return values, and exceptions raised (if any).
- There should be a blank line after the docstring.

Accessing docstrings

Each function contains an attribute `__doc__` that contains its docstring. A function's docstring can be accessed by accessing this attribute or using the `help()` function in a Python interpreter

---

Examples

In this example you will explore how to write a one-line and multi-line docstring and access them using `__doc__`

Celsius to Fahrenheit

```
def C2F(degrees_celsius):
    """ Convert Celsius to Fahrenheit"""
    return degrees_celsius * (9/5) + 32


print("Accessing docstrings using __doc__:\n")
print(C2F.__doc__)
```

```
def C2F(degrees_celsius):
    """ Convert Celsius to Fahrenheit"""
    return degrees_celsius * (9/5) + 32


print("Accessing docstrings using help:\n")
help(C2F)
```

Kilograms (Kg) to Pounds (lb)

```
def kg2lb(kilograms):
    """
    Convert Kilograms to Pounds

    args:
        Kilograms: weight in Kg

    returns:
        Pounds: weight in lb
    """

    pounds = Kilograms * 2.20462262185
    return pounds


print("Accessing docstrings using __doc__:\n")
print(kg2lb.__doc__)
```

```
def kg2lb(kilograms):
    """
    Convert Kilograms to Pounds

    args:
        Kilograms: weight in Kg

    returns:
        Pounds: weight in lb
    """

    pounds = Kilograms * 2.20462262185
    return pounds


print("Accessing docstrings using help:\n")
help(kg2lb)
```

Task 1

Docstrings

```
# [ ] The following function generates a single die roll.
# Document the function using a one-line docstring

from random import randint

def die_roller ():
    return (randint(1, 6))
```

```
# [ ] The following function computes the area of a circle.
# Document the function using a one-line docstring

from math import pi

def circle_area(r):
    return pi * (r ** 2)
```

```
# [ ] The following program counts the number of times the value i
# Document the function using a multi-line docstring

def count_occurrences(a, lst):
    count = 0
    for element in lst:
        if a == element:
            count = count + 1

    return count
```

```
# [ ] The following program prints out the date `d` number of days
# Document the function using a multi-line docstring

from datetime import date, timedelta
def future_date(d):
    today = date.today()
    td = timedelta(days = d)
    future = today + td
    print("Date {:d} from today is: {:s}".format(d, future.strftim


# date 10 days from today
future_date(10)
```

Learn About Verified Certificates