playbook Documentation

Release latest

Feb 22, 2018

Contents

A Python playbook for common algorithms, interesting problems, handy tricks and easy referencing on the go.

This catalogue is a mixture of pre-baked Jupyter notebooks for demonstration and importable code for Python projects.

CHAPTER 1

Data Types

1.1 Data Types

A field briefing of Python's core data types and some useful tips

1.1.1 Sets

Definition

A set is an abstract data type that is a well-defined, unordered collection of elements. Sets contain only distinct elements. Sets are one of the most fundamental concept in mathematics since they were developed near the end of the 19th century.

Unlike various other data structures, elements are more often tested for membership within a set rather than the retrieval of a specific element for said set.

```
# To create a set in Python, we use curly parenthesis, quite similar to creating a␣
↪dictionary
# Let's create a set of numbers ranging from 1 to 7
A = {1, 2, 3, 4, 5, 6, 7}
print ('Set A: ', A)

# Since a set only holds distinct values, what would happen if we attempted to create␣
↪the following
B = {1, 2, 3, 3, 3, 4, 5, 5, 6, 6, 7}
print ('Set B: ', B)

# Assert that A is in fact equal to B since we discard the additional 3, 5 and 6␣
↪values
assert A == B
```

Operations

Union

A union of two sets is the combined set of all elements from a given set A and B. The union operation in Python is performed by either using the | operator or the instance method union()

Let's consider the same set A from above and create a new set C which contains additional values to demonstrate set union

```
A = {1, 2, 3, 4, 5, 6, 7}
B = {8, 9, 10}

# Let D equal the union of two sets A U C
method_union = A.union(B)
operator_union = A | B

assert method_union == {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

Intersection

Intersection of two sets is the elements of a given set A and B that are only available in both sets. For example, given set A and set B, both list of names.

Intersection is performed by the ampersand operator - & or alternatively the instance method intersection()

```
A = {1, 2, 3, 4, 5, 6, 7, 8}
B = {8, 9, 10}

method_intersection = A.intersection(B)
operator_intersection = A & B

assert method_intersection == {8}
```

Difference

Difference is the difference between sets A and B. This is the inverse of intersection, such that we will get the values only in one set and not in another.

Difference is performed using the hyphen operator -

And of course using the instance method difference()

```
A = {'john', 'mary', 'george'}
B = {'martha', 'john', 'thomas'}

method_difference = A.difference(B)
operator_difference = A - B

assert method_difference == ?
```

Symmetrical Difference

Symmetric Difference of two sets is the set of elements that exist in both a set A and a set B except elements that are common in both.

Symmetric Difference is performed using the hat operator - ^.

Or the instance method symmetric_difference()

CHAPTER 2

Data Structures

2.1 Data Structures

Contents of Data Structures

Located here are commonly used data structures in Python and some basic use cases to demonstrate their usage

CHAPTER 3

Algorithms

CHAPTER 4

Examples