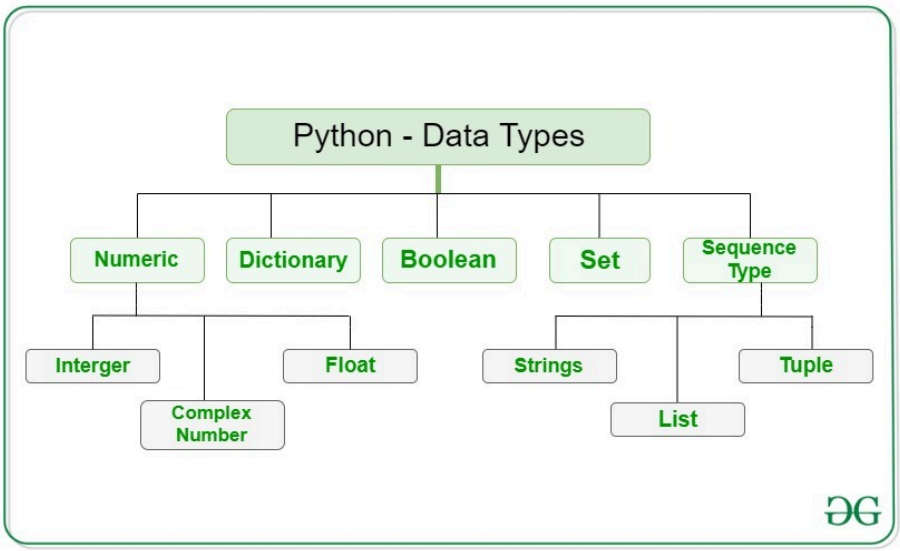
Python Data Types

Python Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, Python data types are classes and variables are instances (objects) of these classes. The following are the standard or built-in data types in Python:

- **Numeric**
- **Sequence Type**
- **Boolean**
- **Set**
- **Dictionary**
- **Binary Types(memoryview, bytearray, bytes)**



What is Python data types?

To define the values of various data types and check their data types we **use the type() function.** Consider the following examples.

This code assigns variable **'x'** different values of various Python data types. It covers **string**, **integer**, **float**, **complex**, **list**, **tuple**, **range**, **dictionary**, **set**, **frozenset**, **boolean**, **bytes**, **bytearray**, **memoryview**, and the special value

'None' successively. Each assignment replaces the previous value, making **'x'** take on the data type and value of the most recent assignment.

Python

```
x = "Hello World"
x = 50
x = 60.5
x = 3j
x = ["geeks", "for", "geeks"]
x = ("geeks", "for", "geeks")
x = range(10)
x = {"name": "Suraj", "age": 24}
x = {"geeks", "for", "geeks"}
x = frozenset({"geeks", "for", "geeks"})
x = True
x = b"Geeks"
x = bytearray(4)
x = memoryview(bytes(6))
x = None
```

Numeric Data Types in Python

The numeric data type in Python represents the data that has a numeric value. A numeric value can be an integer, a floating number, or even a complex number. These values are defined as Python int, Python float, and Python complex classes in Python.

- **Integers** – This value is represented by int class. It contains positive or negative whole numbers (without fractions or decimals). In Python, there is no limit to how long an integer value can be.
- **Float** – This value is represented by the float class. It is a real number with a floating-point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.
- **Complex Numbers** – A complex number is represented by a complex class. It is specified as *(real part) + (imaginary part)j*. For example – 2+3j

Note – type() function is used to determine the type of Python data type.

Example: This code demonstrates how to determine the data type of variables in Python using the **type() function**. It prints the data types of three variables**: a (integer)**, **b (float)**, and **c (complex)**. The output shows the respective Python data types for each variable.

Python

```
a = 5
print("Type of a: ", type(a))

b = 5.0
print("\nType of b: ", type(b))

c = 2 + 4j
print("\nType of c: ", type(c))
```

Output:

```
Type of a:  <class 'int'>
Type of b:  <class 'float'>
```

```
Type of c:  <class 'complex'>
```

Sequence Data Type in Python

The sequence Data Type in Python is the ordered collection of similar or different Python data types. Sequences allow storing of multiple values in an organized and efficient fashion. There are several sequence types in Python –

- Python String
- Python List
- Python Tuple

String Data Type

Strings in Python are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote, or triple-quote. In Python there is no character data type, a character is a string of length one. It is represented by str class.

Creating String

Strings in Python can be created using single quotes, double quotes, or even triple quotes.

Example: This Python code showcases various string creation methods. It uses single quotes, double quotes, and triple quotes to create strings with different content and includes a multiline string. The code also demonstrates printing the strings and checking their data types.

Python

```
String1 = 'Welcome to the Geeks World'
print("String with the use of Single Quotes: ")
print(String1)
String1 = "I'm a Geek"
print("\nString with the use of Double Quotes: ")
print(String1)
print(type(String1))
String1 = '''I'm a Geek and I live in a world of "Geeks"'''
print("\nString with the use of Triple Quotes: ")
print(String1)
print(type(String1))
```

```
String1 = '''Geeks
            For
            Life'''
print("\nCreating a multiline String: ")
print(String1)
```

Output:

```
String with the use of Single Quotes:
Welcome to the Geeks World
String with the use of Double Quotes:
I'm a Geek
<class 'str'>
String with the use of Triple Quotes:
I'm a Geek and I live in a world of "Geeks"
<class 'str'>
Creating a multiline String:
Geeks
            For
            Life
```

Accessing elements of String

In Python programming, individual characters of a String can be accessed by using the method of Indexing. Negative Indexing allows negative address references to access characters from the back of the String, e.g. -1 refers to the last character, -2 refers to the second last character, and so on.

Example: This Python code demonstrates how to work with a string named '**String1**'. It initializes the string with **"GeeksForGeeks"** and prints it. It then showcases how to access the first character (**"G"**) using an index of 0 and the last character (**"s"**) using a negative index of -1.

Python

```
String1 = "GeeksForGeeks"
print("Initial String: ")
print(String1)
print("\nFirst character of String is: ")
```

```
print(String1[0])
print("\nLast character of String is: ")
print(String1[-1])
```

Output:

```
Initial String:
GeeksForGeeks
First character of String is:
G
Last character of String is:
s
```

Note – To know more about strings, refer to Python String.

List Data Type

Lists are just like arrays, declared in other languages which is an ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.

Creating a List in Python

Lists in Python can be created by just placing the sequence inside the square brackets[].

Example: This Python code demonstrates list creation and manipulation. It starts with an empty list and prints it. It creates a list containing a single string element and prints it. It creates a list with multiple string elements and prints selected elements from the list. It creates a multi-dimensional list (a list of lists) and prints it. The code showcases various ways to work with lists, including single and multi-dimensional lists.

Python

```
List = []
print("Initial blank List: ")
print(List)
List = ['GeeksForGeeks']
print("\nList with the use of String: ")
print(List)
```

```
List = ["Geeks", "For", "Geeks"]
print("\nList containing multiple values: ")
print(List[0])
print(List[2])
List = [['Geeks', 'For'], ['Geeks']]
print("\nMulti-Dimensional List: ")
print(List)
```

Output:

```
Initial blank List:
[]
List with the use of String:
['GeeksForGeeks']
List containing multiple values:
Geeks
Geeks
Multi-Dimensional List:
[['Geeks', 'For'], ['Geeks']]
```

Python Access List Items

In order to access the list items refer to the index number. Use the index operator [ ] to access an item in a list. In Python, negative sequence indexes

the offset as in List[len(List)-3], it is enough to just write List[-3]. Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second-last item, etc.

Python

```
List = ["Geeks", "For", "Geeks"]
print("Accessing element from the list")
print(List[0])
print(List[2])
print("Accessing element using negative indexing")
print(List[-1])
print(List[-3])
```

Output:

```
Accessing element from the list
Geeks
Geeks
Accessing element using negative indexing
Geeks
Geeks
```

Note – To know more about Lists, refer to Python List.

Tuple Data Type

Just like a list, a tuple is also an ordered collection of Python objects. The only difference between a tuple and a list is that tuples are immutable i.e. tuples cannot be modified after it is created. It is represented by a tuple class.

Creating a Tuple in Python

In Python Data Types, tuples are created by placing a sequence of values separated by a 'comma' with or without the use of parentheses for grouping the data sequence. Tuples can contain any number of elements and of any datatype (like strings, integers, lists, etc.). **Note:** Tuples can also be created with a single element, but it is a bit tricky. Having one element in the parentheses is not sufficient, there must be a trailing **'comma'** to make it a tuple.

Example: This Python code demonstrates different methods of creating and working with tuples. It starts with an empty tuple and prints it. It creates a tuple containing string elements and prints it. It converts a list into a tuple and prints the result. It creates a tuple from a string using the **tuple()** function. It forms a tuple with nested tuples and displays the result.

Python

```python
Tuple1 = ()
print("Initial empty Tuple: ")
print(Tuple1)
Tuple1 = ('Geeks', 'For')
print("\nTuple with the use of String: ")
```

```
print(Tuple1)
list1 = [1, 2, 4, 5, 6]
print("\nTuple using List: ")
print(tuple(list1))
Tuple1 = tuple('Geeks')
print("\nTuple with the use of function: ")
print(Tuple1)
Tuple1 = (0, 1, 2, 3)
Tuple2 = ('python', 'geek')
Tuple3 = (Tuple1, Tuple2)
print("\nTuple with nested tuples: ")
print(Tuple3)
```

Output:

```
Initial empty Tuple:
()
Tuple with the use of String:
('Geeks', 'For')
Tuple using List:
(1, 2, 4, 5, 6)
Tuple with the use of function:
('G', 'e', 'e', 'k', 's')
Tuple with nested tuples:
((0, 1, 2, 3), ('python', 'geek'))
```

Note – The creation of a Python tuple without the use of parentheses is known as Tuple Packing.

Access Tuple Items

In order to access the tuple items refer to the index number. Use the index operator [ ] to access an item in a tuple. The index must be an integer. Nested tuples are accessed using nested indexing.

The code creates a tuple named '**tuple1'** with five elements: **1, 2, 3, 4, and 5**. Then it prints the first, last, and third last elements of the tuple using indexing.

Python

```
tuple1 = tuple([1, 2, 3, 4, 5])
```

```
print("First element of tuple")
print(tuple1[0])
print("\nLast element of tuple")
print(tuple1[-1])

print("\nThird last element of tuple")
print(tuple1[-3])
```

Output:

```
First element of tuple
1
Last element of tuple
5
Third last element of tuple
3
```

Note – To know more about tuples, refer to Python Tuples.

Boolean Data Type in Python

Python Data type with one of the two built-in values, True or False. Boolean objects that are equal to True are truthy (true), and those equal to False are falsy (false). However non-Boolean objects can be evaluated in a Boolean context as well and determined to be true or false. It is denoted by the class bool.

Note – True and False with capital 'T' and 'F' are valid booleans otherwise python will throw an error.

Example: The first two lines will print the type of the boolean values True and False, which is **<class 'bool'>.** The third line will cause an error, because true is not a valid keyword in Python. Python is case-sensitive, which means it distinguishes between uppercase and lowercase letters. You need to capitalize the first letter of true to make it a boolean value.

Python

```
print(type(True))
print(type(False))
```

```
print(type(true))
```

Output:

```
<class 'bool'>
<class 'bool'>


Traceback (most recent call last):
  File "/home/7e8862763fb66153d70824099d4f5fb7.py", line 8, in
    print(type(true))
NameError: name 'true' is not defined
```

Set Data Type in Python

In Python Data Types, a Set is an unordered collection of data types that is iterable, mutable, and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.

Create a Set in Python

Sets can be created by using the built-in set() function with an iterable object or a sequence by placing the sequence inside curly braces, separated by a **'comma'.** The type of elements in a set need not be the same, various mixed-up data type values can also be passed to the set.

Example: The code is an example of how to create sets using different types of values, such as **strings**, **lists**, and mixed values

---

Python

```
set1 = set()
print("Initial blank Set: ")
print(set1)
set1 = set("GeeksForGeeks")
print("\nSet with the use of String: ")
print(set1)
set1 = set(["Geeks", "For", "Geeks"])
print("\nSet with the use of List: ")
print(set1)
set1 = set([1, 2, 'Geeks', 4, 'For', 6, 'Geeks'])
```

```
print("\nSet with the use of Mixed Values")
print(set1)
```

Output:

```
Initial blank Set:
set()
Set with the use of String:
{'F', 'o', 'G', 's', 'r', 'k', 'e'}
Set with the use of List:
{'Geeks', 'For'}
Set with the use of Mixed Values
{1, 2, 4, 6, 'Geeks', 'For'}
```

Access Set Items

Set items cannot be accessed by referring to an index, since sets are unordered the items have no index. But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in the keyword.

Example: This Python code creates a set named **set1** with the values **“Geeks”**, **“For”** and **“Geeks”**. The code then prints the initial set, the elements of the set in a loop, and checks if the value **“Geeks”** is in the set using the ‘**in**’ operator

---

Python

```
set1 = set(["Geeks", "For", "Geeks"])
print("\nInitial set")
print(set1)
print("\nElements of set: ")
for i in set1:
    print(i, end=" ")
print("Geeks" in set1)
```

Output:

```
Initial set:
{'Geeks', 'For'}
Elements of set:
Geeks For
True
```

Note – To know more about sets, refer to Python Sets.

Dictionary Data Type in Python

A dictionary in Python is an unordered collection of data values, used to store data values like a map, unlike other Python Data Types that hold only a single value as an element, a Dictionary holds a key: value pair. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon : , whereas each key is separated by a 'comma'.

Create a Dictionary in Python

In Python, a Dictionary can be created by placing a sequence of elements within curly {} braces, separated by 'comma'. Values in a dictionary can be of any datatype and can be duplicated, whereas keys can't be repeated and must be immutable. The dictionary can also be created by the built-in function **dict().** An empty dictionary can be created by just placing it in curly braces{}. **Note** – Dictionary keys are case sensitive, the same name but different cases of Key will be treated distinctly.

Example: This code creates and prints a variety of dictionaries. The first dictionary is empty. The second dictionary has integer keys and string values. The third dictionary has mixed keys, with one string key and one integer key. The fourth dictionary is created using the **dict()** function, and the fifth dictionary is created using the **[(key, value)]** syntax

---

Python

```python
Dict = {}
print("Empty Dictionary: ")
print(Dict)
Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}
print("\nDictionary with the use of Integer Keys: ")
```

```
print(Dict)
Dict = {'Name': 'Geeks', 1: [1, 2, 3, 4]}
print("\nDictionary with the use of Mixed Keys: ")
print(Dict)
Dict = dict({1: 'Geeks', 2: 'For', 3: 'Geeks'})
print("\nDictionary with the use of dict(): ")
print(Dict)
Dict = dict([(1, 'Geeks'), (2, 'For')])
print("\nDictionary with each item as a pair: ")
print(Dict)
```

Output:

```
Empty Dictionary:
{}
Dictionary with the use of Integer Keys:
{1: 'Geeks', 2: 'For', 3: 'Geeks'}
Dictionary with the use of Mixed Keys:
{1: [1, 2, 3, 4], 'Name': 'Geeks'}
Dictionary with the use of dict():
{1: 'Geeks', 2: 'For', 3: 'Geeks'}
Dictionary with each item as a pair:
{1: 'Geeks', 2: 'For'}
```

Accessing Key-value in Dictionary

In order to access the items of a dictionary refer to its key name. Key can be used inside square brackets. There is also a method called **get()** that will also help in accessing the element from a dictionary.

Example: The code in Python is used to access elements in a dictionary. Here's what it does, It creates a dictionary Dict with keys and values as {**1: 'Geeks', 'name': 'For', 3: 'Geeks'}**. It prints the value of the element with the key **'name'**, which is **'For'**. It prints the value of the element with the key 3, which is **'Geeks'**.

Python

```
Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}
print("Accessing a element using key:")
```

```
print(Dict['name'])
print("Accessing a element using get:")
print(Dict.get(3))
```

Output:

```
Accessing a element using key:
For
Accessing a element using get:
Geeks
```

Python Data Type Exercise Questions

Below are two exercise questions on Python Data Types. We have covered list operation and tuple operation in these exercise questions. For more exercises on Python data types visit the page mentioned below.

Q1. Code to implement basic list operations

Python

```
fruits = ["apple", "banana", "orange"]
print(fruits)
fruits.append("grape")
print(fruits)
fruits.remove("orange")
print(fruits)
```

Output

```
['apple', 'banana', 'orange']
['apple', 'banana', 'orange', 'grape']
['apple', 'banana', 'grape']
```

Q2. Code to implement basic tuple operation

Python

```
coordinates = (3, 5)
```

```
print(coordinates)
print("X-coordinate:", coordinates[0])
print("Y-coordinate:", coordinates[1])
```

Output

```
(3, 5)
X-coordinate: 3
Y-coordinate: 5
```

Explore more Exercises: Python Data Type Exercise

Don't miss your chance to ride the wave of the data revolution! Every industry is scaling new heights by tapping into the power of data. Sharpen your skills and become a part of the hottest trend in the 21st century.

Dive into the future of technology - explore the Complete Machine Learning and Data Science Program by GeeksforGeeks and stay ahead of the curve.

Last Updated : 19 Mar, 2024

Previous

Python Tutorial | Learn Python Programming

Next

Python Arrays

Similar Reads

Python | Get tuple element data types

Python | Convert mixed data types tuple list to string list

How To Convert Data Types in Python 3?

Python - Extract rows with Complex data types

Python program to extract rows from Matrix that has distinct data types

Return True if Cast Between Data Types can Occur According to the Casting rule in Python

How to convert unstructured data to structured data using Python ?

Python - Convert Tick-by-Tick data into OHLC (Open-High-Low-Close) Data

How to convert categorical data to binary data in Python?

Kotlin Data Types

N nikhilagg...

Article Tags : Python-datatype , Python

Practice Tags : python