Chapter 3: Python Data Types

1. Built-in Data Types

Python has the following data types built-in by default, in these categories:

Text Type:	`str`
Numeric Types:	`int, float`
Sequence Types:	`list, tuple, range`
Mapping Type:	`dict`
Set Types:	`set, frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes, bytearray`
None Type:	`NoneType`

Getting the Data Type

You can get the data type of any object by using the `type()` function:

```
x = 5
print(type(x))
```

2. Python Numbers

Variables of numeric types are created when you assign a value to them:

```
x = 1    # int
y = 2.8  # float
```

Int (integer) is a whole number, positive or negative, without decimals, of unlimited length.

```
x = 1
y = 35656222554887711
z = -3255522
```

Float is a number, positive or negative, containing one or more decimals.

```
x = 1.10
y = 1.0
z = -35.59
```

Float can also be scientific numbers with an “e” to indicate the power of 10.

```
x = 35e3
y = 12E4
z = -87.7e100
```

<u>Type Conversion</u>

You can convert with the `int()`, `float()` methods:

```
x = 1    # int
y = 2.8  # float

#convert from int to float:
a = float(x)

#convert from float to int:
b = int(y)

print(type(a))
print(type(b))
```

<u>Random Number</u>

Python does not have a `random()` function to make a random number, but Python has a built-in module called `random` that can be used to make random numbers:

```
import random

print(random.randrange(1, 10))
```

3. <u>Strings</u>

Strings in python are surrounded by either single quotation marks, or double quotation marks.

`'hello'` is the same as `”hello”`.

```
print(“Hello”)
print('Hello')
```

<u>Assign String to a Variable</u>

```
a = ”Hello”
print(a)
```

Multiline Strings

You can assign a multiline string to a variable by using three quotes:

```
a = ”““Lorem ipsum dolor sit amet,
       consectetur adipiscing elit,
        sed do eiusmod tempor incididunt
        ut labore et dolore magna aliqua.”““
print(a)
```

Or three single quotes:

```
a = '''Lorem ipsum dolor sit amet,
       consectetur adipiscing elit,
       sed do eiusmod tempor incididunt
       ut labore et dolore magna aliqua.'''
print(a)
```

Note: in the result, the line breaks are inserted at the same position as in the code.

Strings are Arrays

Strings in Python are arrays of characters (as unicode bytes). Python does not have a “char” data type, a single character is simply a string with a length of 1. The first character has the position 0:

Square brackets can be used to access elements of the string.

```
a = ”Hello, World!”
print(a[1])
```

Looping Through a String

Since strings are arrays, we can loop through the characters in a string, with a `for` loop.

```
for x in ”banana”:
  print(x)
```

String Length

To get the length of a string, use the `len()` function.

```
a = ”Hello, World!”
print(len(a))
```

Check String

To check if a certain phrase or character is present in a string, we can use the keyword `in`.

```
print(“free” in ”The best things in life are free!”)    #true
```

Use it in an `if` statement:

```
txt = ”The best things in life are free!”
x = ”free”
if x in txt:
  print(“Yes, it is present.”)
```

Check if NOT

To check if a certain phrase or character is NOT present in a string, we can use the keyword `not in`.

```
txt = ”The best things in life are free!”
print(“expensive” not in txt)
```

Use it in an `if` statement:

```
txt = ”The best things in life are free!”
if ”expensive” not in txt:
  print(“No, 'expensive' is NOT present.”)
```

Slicing

You can return a range of characters by using the slice syntax. Specify the start index and the end index to return a part of the string.

#Get the characters from position 2(included) to position 5 (not included):

```
b = ”Hello, World!”
print(b[2:5])
```

Slice From the Start

By leaving out the start index, the range will start at the first character:

#Get the characters from the start to position 5 (not included):

```
b = ”Hello, World!”
print(b[:5])
```

Slice To the End

By leaving out the *end* index, the range will go to the end:

```
b = ”Hello, World!”
print(b[2:])
```

Negative Indexing

Use negative indexes to start the slice from the end of the string (begin from write to left with index 1):

```
# Get the characters: From: “o” in “World!” (position -5)
# To, but not included: “d” in “World!” (position -2):

b = ”Hello, World!”
print(b[-5:-2])
```

Upper Case and Lower Case

```
a = ”Hello, World!”
print(a.upper())
print(a.lower())
```

Remove Whitespace

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

```
a = ” Hello, World! “
print(a.strip()) # returns “Hello, World!”
```

Replace String

```
a = ”Hello, World!”
print(a.replace(“H”, ”J”))
```

Split String

The `split()` method returns a list where the text between the specified separator becomes the list items.

```
a = ”Hello, World!”
print(a.split(“,”)) # returns ['Hello', ' World!']
```

String Concatenation

To concatenate, or combine, two strings you can use the + operator.

```
a = ”Hello”
b = ”World”
c = a + b
d = a + ” “ + b
print(c)
print(d)
```

String Format

We cannot combine strings and numbers like this:

```
age = 21
print(”My name is Jo, I am “ + age)    # error
```

But we can combine strings and numbers by using the `format()` method.

```
age = 21
txt = ”My name is Jo, and I am {}”
print(txt.format(age))
```

The `format()` method takes unlimited number of arguments, and are placed into the respective placeholders:

```
quantity = 3
itemno = 567
price = 49.95
myorder = ”I want {} pieces of item {} for {} dollars.”
print(myorder.format(quantity, itemno, price))
```

You can use index numbers `{0}` to be sure the arguments are placed in the correct placeholders:

```
quantity = 3
itemno = 567
price = 49.95
order = ”Want to pay {2} dollars for {0} pieces of item {1}.”
print(order.format(quantity, itemno, price))
```

4. **Booleans**

When you compare two values, the expression is evaluated and Python returns the Boolean answer `True` or `False`:

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

We can run a condition using an if statement:

```
a = 200
b = 33

if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Most Values are True

- Any string is `True`, except empty strings.
- Any number is `True`, except `0`.
- Any list, tuple, set, and dictionary are `True`, except empty ones.

```
# The following will return True:

bool("abc")
bool(123)
bool(["apple", "cherry", "banana"])
```

Functions can return Boolean

You can create functions that returns a Boolean Value:

```
def isTrue() :
  return True

print(isTrue())
```

You can execute code based on the Boolean answer of a function:

```
def isTrue() :
  return True

if isTrue():
  print("YES!")
else:
  print("NO!")
```