Python Data Types

Built-in Data Types

In programming, data type is an important concept.
Variables can store data of different types, and different types can do different things.
Python has the following data types built-in default, in these categories :

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

Getting the Data Type

You can get the data type of any object by using type() function :
Example : Print the data type of the variable x :

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

Ans. <class 'int'>

Setting the Data Type

In Python, the data type is set when you assign a value to a variable:

Example	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "mango"]	list
x = ("apple", "banana", "mango")	tuple
x = range(6)	range
x = {"name": "John", "age": 36}	dict
x = {"apple", "banana", "mango"}	set
x = frozenset({"apple", "banana", "mango"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview
x = None	NoneType

Setting the Specific Data Type

If you want to specify the data type, you can use the following constructor functions:

Example	Data Type
x = str("Hello World")	str
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = list(("apple","banana","mango"))	list
x = tuple(("apple","banana","mango"))	tuple
x = range(6)	range
x = dict(name="John", age=36)	dict
x = set(("apple","banana","mango"))	set
x = frozenset(("apple","banana","mango"))	frozenset
x = bool(5)	bool
x = bytes(5)	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

Python Numbers

There are three numeric types in Python :

- int
- float
- complex

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

Example

```
x = 1      Print (type(x)) # int
y = 2.8    Print (type(y)) # float
z = 1j     print ( type(z)) # complex
```

Int

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

Example

Integers :

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

Float

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

Example
Float:

```
X = 1.10
Y = 1.0
Z = -35.59
```

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

Example
Floats:

```
X = 35e3
Y = 12E4
Z = -87.7e100
```

Complex

Complex numbers are written with a "j" as the imaginary part:

Example: Complex:

```
X = 3+5j
Y = 5j
Z = -5j
```

Type Conversion

You can convert from one type to another with the int(), float(), and complex() methods:

Example

Convert from one type to another:

```
x = 1   # int
y = 2.8 # float
z = 1j  # complex

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

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

# convert from int to complex:
c = complex(x)
```

Random Number

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:

Example: Import the random module, and display a random number between 1 and 9:

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

Ans:- 9

Python Casting

Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in Python is therefore done using constructor functions:

- int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number).
- float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer).
- str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals.

Example: Integers:

```
x = int(1) # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

Example: Float:

```
x = float(1)   # x will be 1.0
y = float(2.8) # y will be 2.8
z = float("3") # z will be 3.0
w = float("4.2") # w will be 4.2
```

Example: strings:

```
x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

Python Strings

Strings

Strings in Python are surrounded by either single quotation marks, or double quotation marks.
'hello' is the same as "hello".

You can display a string literal with the print() function:

Example:

```
Print("Hello") # Hello
Print('Hello') # Hello
```

Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

Example:

```
a = "Hello"
Print(a) # Hello
```

Multiline Strings

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

Example :

You can use three double quotes :

```
a= """ If you root yourself
in love
and kindness,
your heart will always bloom."""
Print (a)
```

Ans:→

```
If you root yourself
in love
and kindness,
your heart will always bloom.
```

Strings are Arrays

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1.

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

Example:

Get the character at position 1 (remember that the first character has the position 0):

```
a = "Hello, world!"
print(a[1])
```

Ans:➡ e

Looping Through a String

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

Example:

Loop through the letters in the word "banana":

```
for x in "banana":
  print(x)
```

Ans:➡
```
b
a
n
a
n
a
```

String Length

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

Example:

The len() function returns the length of a string:

```
a = "Hello, world"
print(len(a))
```

Ans:→ 13

Check String

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

Example:

Check if "free" is present in the following text:

```
txt = "The best things in life are free!"
print ("free" in txt)
```

Ans:→ True

Use it in an if statement:

Example:

Print only if "free" is present:

```
txt = "The best things in life are free!"
if "free" in txt
    Print ("yes", 'free' is present.")
```

Ans:→ yes, 'free' 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.

Example:

Check if "expensive" is NOT present in the following text:

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

Ans:➡ True

Use it in an if statement:

Example:

Print only if "expensive" is NOT present:

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

Ans:➡ No, 'expensive' is NOT present.