HyperionDev

Numerical Data Types

Task

Visit our website

Introduction

You are already very familiar with numbers as we encounter them daily and, in most cases, multiple times a day. It is likely that, through studying mathematics at school, you were introduced to different types of numbers such as integers and decimal numbers, as well as the operations we can perform on them, such as addition, subtraction, multiplication, and division. Computer programming languages like Python provide support for storing and manipulating many different types of numbers. In this task, you will learn how numbers are categorised based on their nature, as well as how to perform arithmetic operations in Python.

Extra resource

Software development is full of acronyms. On the **Hyperion Hub** you will find some frequently used software development terms you would do well to be familiar with, for when they are next thrown your way.

Numbers in our everyday lives

Whether you are fond of math or not, you cannot escape numbers. Numbers play an extremely important role in our daily lives. Probably not a day goes by without you interacting with numbers in some way. Think about it. Did you go shopping today and look at the prices of items? Did you change the volume or channel on your TV? Did you drive anywhere today and notice the speed limit signs or instrumentation on your vehicle's dashboard? Have you made a telephone call today? Chances are you have done at least one of these things and if you have not, you can probably think of numerous other instances where you have encountered numbers today.

You have also probably learnt from school math that there are different types of numbers. For example, whole numbers and decimal numbers. Whole numbers do not contain a fractional part and can be used to count the items in a list. However, decimal numbers do contain a fractional part. Decimal numbers are normally used when precision is required, such as when dealing with measurements or currency.

With numbers being so important in our daily lives, it comes as no surprise that they are equally important in programming. Every single programming language provides support for manipulating, storing, and defining different types of numbers.

Numbers in python

There are three distinct numeric types in Python 3:

- **Integers** can be either positive or negative, include the number 0, and do not include a fractional or decimal part (e.g., -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, etc. are all integers). Integers are normally used for counting or simple calculations. For example, you can store the number of items you wish to purchase from a store as an integer, e.g. num = int("-12").
- **Floating-point numbers** are decimal numbers or numbers which contain a fractional component. They are useful when you need more precision, for example, when storing measurements for a building or amounts of money. Floats may also be in scientific notation, with E or e indicating the power of 10, e.g. x = float("15.20"), y = float("-32.54e100").
- **Complex numbers** have a real and imaginary part, which are each a floating-point number, e.g. c = complex("45.j").

When you declare a variable in Python, it will already know if it is a float or an integer based on the characteristics used. If you use decimals, it will automatically be a float and if there are no decimals, then it will be an integer.

```
class_list = 25  # integer
interest_rate = 12.23  # float
```

To cast between numbers, make use of the **int()** or **float()** functions, depending on which is needed.

```
num1 = 12
num2 = 99.99
print(float(num1))
# Converting floats to ints, as below, causes data loss.
# int() removes values after the decimal point, returning only a whole number.
print(int(num2))
```

Arithmetic operations

Doing calculations with numbers in Python works similarly to the way they would in normal math.

```
total = 2 + 4
print(total)
# prints out 6
```

```
total_amount = 0.25 + 4.33
print(total_amount)
# prints out 4.58
```

The only difference between calculations in real mathematics and programming is the symbols you use:

Arithmetic operations	**Python symbol**
Addition: adds values on either side of the operator.	+
Subtraction: subtracts the value on the right of the operator from the value on the left.	-
Multiplication: multiplies values on either side of the operator.	*
Division: divides the value on the left of the operator by the value on the right.	/
Modulus: divides the value on the left of the operator by the value on the right and returns the remainder .	%
Exponent: performs an exponential calculation, i.e. calculates the answer of the value on the left to the power of the value on the right.	**

Mathematical functions

In Python, numerical data types play a crucial role in performing mathematical operations and calculations. To work with numerical data effectively, Python offers a variety of built-in functions (pre-written code) and libraries that can be utilised. Built-in functions are readily available within Python and provide fundamental mathematical operations, while the **math** module needs to be imported and offers an extensive collection of specialised mathematical functions.

Some of the most commonly used functions are listed in the tables below.

Built-in mathematical function

Function	Description	Example
round()	Rounds a floating-point number to the nearest whole number, or decimal places as specified by the second argument.	number = 66.6564544 print(round(number,2)) will output 66.66
min()	Returns the smallest value from an iterable, such as a list or tuple.	numbers_list = [6,4,66,35,1] print(min(numbers_list)) will output 1
max()	Returns the largest value from an iterable, such as a list or tuple.	numbers_list = [6,4,66,35,1] print(max(numbers_list)) will output 66
sum()	Calculates the total sum of all elements in iterable, such as a list or tuple.	numbers_list = [6,4,66,35,1] print(sum(numbers_list)) will output 112

Mathematical functions available through the math module

To be able to use the functions in the math module, add this line of code to the top of your program:

```
import math
```

Function	Description	Example
math.floor()	Rounds a number down.	print(math.floor(30.3333)) will output 30.0
math.ceil()	Rounds a number up.	print(math.ceil(30.3333)) will output 31.0
math.trunc()	Cuts off the decimal part of the float.	print(math.trunc(30.33333)) will output 30
math.sqrt()	Finds the square root of a number.	print(math.sqrt(4)) will output 2.0
math.pi()	Returns the value for pi where pi is the number used to calculate the area of a circle.	print(math.pi) will output 3.141592653589793

Feel free to explore some more functions in the **math** module in the **Python documentation**.

Extra resource

The further you progress on your Python journey, the more complex the programs you build will become. Here are the top five Python libraries that you could use to create these complex programs. It helps to be aware of them and know what they do so that you can call on them when needed and as you gain more experience.

- **Pillow**: A Python imaging library used for image processing.
- **NumPy**: A fundamental package for scientific computing using Python. It adds support for linear algebra, statistics, large multidimensional arrays, and matrices, along with a large library of high-level mathematical functions to operate on these arrays.
- **PyGame**. A cross-platform set of Python modules for creating video games and multimedia programs. It is highly portable and runs on nearly every platform and operating system.
- **Scrappy**. An open-source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.
- **Apache Libcloud**. A library interacting with many of the popular cloud service providers using a unified API. It was created to make it easy for developers to build products and work between any of the services that it supports.

Take note

Before you get started, we strongly suggest you use an editor such as VS Code to open all text files (.txt) and Python files (.py).

First, read the accompanying Python example files. These examples should help you understand some simple Python. You may run the examples to see the output. Feel free to also write and run your own example code before doing the compulsory tasks, to become more comfortable with Python.

Practical task 1

Follow these steps:

- Create a new Python file in this folder called **integermath.py**.
- Ask the user to enter three different integers.

Then print out:

  - The sum of all the numbers.
  - The first number minus the second number.
  - The third number is multiplied by the first number.
  - The sum of all three numbers divided by the third number.

Practical task 2

Follow these steps:

- Create a new Python file in this folder called **shopping.py**.
- Once this is done, ask the user to enter the names of three products
- Now ask for the price of each product. Each product must have two decimal values.
- Calculate the total sum of all three products.
- Calculate the average price of the three products. (Hint: look up the `round()` function)
- Then print out the following sentence after performing your calculations:

“The Total of [product1], [product2], [product3] is Rxx,xx and the average price of the items is Rxx,xx.”

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Challenge

Use this opportunity to extend yourself by completing an optional challenge activity.

Follow these steps:

- Create a new file called **challenge.py**.
- Write Python code to take the name of a user's favourite restaurant and store it in a variable called **string_fav**.
- Below this, write a statement to take in the user's favourite number. Use casting to store it in an integer variable called **int_fav**.
- Print out both of these using two separate print statements below what you have written. Be careful!
- Once this is working, try to cast **string_fav** to an integer. What happens? Add a comment in your code to explain why this is.

Important: To have this task reviewed, book a call with a mentor through your dashboard.

Take note

Make sure that you have installed and set up all programs correctly. You have set up GitHub correctly if you are reading this, but Python or your editor may not be installed correctly.

If you are not using Windows, please ask a mentor for alternative instructions.

Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.
