TASK

Numerical Data Types

Visit our website

Introduction

WELCOME TO THE NUMERICAL DATA TYPES TASK!

We encounter numbers daily. At school, you were likely introduced to different types of numbers such as integers and decimal numbers, as well as the mathematical operations we can perform on numbers, such as addition, subtraction, multiplication and division.

Computer programming languages such as Python provide support for storing and manipulating various kinds of numbers. In this task, you will learn how numbers are categorised based on their nature, and how to perform arithmetic operations in Python.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

NUMBERS IN EVERYDAY LIFE

Whether you are fond of math or not, you cannot escape numbers. Not a day goes by without 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 somewhere and notice the speed limit signs on the road? You can probably think of numerous other instances where you encountered numbers today.

You probably learned from high 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 items in a list. Decimal numbers contain a fractional part and 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

In Python, numbers are generally stored as either integers or floats.

Integers

These are synonymous with whole numbers. Numbers which are stored as this type do not contain a fractional part or decimal.

Integers can either be positive or negative and 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.

Floats

Decimal numbers or numbers which contain a fractional component are stored as floats. Examples of numbers represented as floats are 0.75 and -6.593. Floats are useful when you need more precision, for example, when storing measurements for a building or amounts of money.

Floats can be created directly by creating a variable and assigning to it a number with a decimal point, or by using mathematical operations (such as division). When an arithmetic operation is performed on two floats, or on a float and an integer, the resulting output is a float. See the difference in output between the two examples below.

```
#Example A

num1 = 3.0
num2 = 2
print(num1 + num2)
#Output: 5.0

#Example B

num1 = 3
num2 = 2
print(num1 + num2)
#Output: 5
```

Declaring Numbers in Python

When you create a variable in Python, it is able to figure out whether the number stored is a Float or an Integer based on the characteristics used. So, if you use decimals, the variable will automatically be a classified as a Float and if there is no decimal then as an Integer.

```
class_list = 25
interest_rate = 12.23
```

Casting Between Numeric Types

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

```
num1 = 12
num2 = 99.99

print(float(num1))
#Output: 12.0

print(int(num2))
#Output: 99
```

ARITHMETIC OPERATIONS

Python can carry out arithmetic calculations similar to how you would do it in 'regular math'.

Keep in mind the symbols you use while performing operations in programming:

- Addition uses +
- Subtraction uses -
- Multiplication uses *
- Division uses /

```
sum = 2+4
print(sum)
#Output: 6
```

You can use parentheses to signify which operations are performed first, similar to math. In the example below, using brackets ensures that the addition part is carried out before the division. (The result would be different if there were no brackets.)

```
conversion = 5 / (1.33 + 0.25)
print(conversion)
#Output: 3.16455696203
```

You can also perform operations on negative numbers. Dividing by zero results in an error, as can be expected.

Exponentiation

Python also supports exponentiation (raising a number to a power) using the symbol **.

```
num = 2**4
print(num)
#Output: 16
```

Division quotient and remainder

It is useful to know two more operators connected with division.

- **Quotient:** 'Floor division' is performed using two forward slashes (`//`) and helps determine the whole number quotient that results from dividing one number by another (rounded down to the nearest integer)
- **Remainder:** The remainder is obtained by carrying out the 'modulo operation' using the percent symbol (`%`).

For instance, when you divide 17 by 5, the quotient is 3, and the remainder is 2.

```
print(17//5)
#Output: 3

print(17%5)
#Output: 2
```

The modulo operator (`%`) used for determining the remainder is extremely useful when dealing with problems that require determining **whether a number is odd or even**. For instance, a program may require you to print out only the 'even' numbers in a list. In such cases, you can use the `if` function (which we will learn about later) along with `x%2==0` (if x divided by 2 leaves no remainder, it must be an even number).

Instructions

Before you get started, we strongly suggest you start using Notepad++ or IDLE to open all text files (.txt) and Python files (.py). Do not use the normal Windows Notepad or Mac TextEdit as it will be much harder to read.

First, make sure you've read *example.py* in this task folder by opening it with your IDLE or Notepad++ (right-click the file and select 'Edit with Notepad++').

- The file **example.py** should help you understand some simple Python. Every task will have example code to help you get started. Make sure you read all of example.py and try your best to understand.
- Do run example.py to see the output. Feel free to write and run your own example code before doing the tasks to become more comfortable with Python.

Compulsory Task 1

- Create a new Python file in this folder called **numbers.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 multiplied by the first number
    - The sum of all three numbers divided by the third number.

Compulsory Task 2

- Create a new Python file in this folder called **shopping.py**.
- Once this is done, ask the user to enter the name of three products.
- The price of each product. Each product must have two decimal values.
- Calculate the total of all three products.
- Calculate the average price of the three products.
- 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 are Rxx,xx."

Completed the task(s)?

Ask your mentor to review your work!

Review work

Thing(s) to look out for:

Make sure that you have installed and set up **Dropbox** correctly on your machine.

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.