Hyperiondev

TASK

Numerical Data Types

Visit our website

Introduction

Welcome to The Numerical Data Types Task!

You are probably already very familiar with numbers as we encounter them daily and, in most cases, multiple times a day. Through studying mathematics at school you were introduced to the 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, such as 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.

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!

A note from the
Hyperion Team

Software development is full of acronyms. Here's a link to the Hyperion Hub where you will find some frequently used software development terms you'd do well to be familiar with, for when it's 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 haven't, you can probably think of numerous other instances where you have encountered numbers today.

You have also probably learned 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, floating point numbers and complex numbers.

- **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, e.g. num = int("-12").

- **Floats**: Decimal numbers or numbers which contain a fractional component are stored as floats. 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**: Complex numbers have a real and imaginary part, which are each a floating point number, e.g. c = complex("45.j").

Declaring Numbers in Python

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
```

Casting Between Numeric Types

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))
print(int(num2))
```

Arithmetic Operations

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

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

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

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

Arithmetic Operations	Symbol used in Python
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 exponential calculation, i.e. calculates the answer of the value on the left to the power of the value on the right	**

Mathematical Functions

Python also has a library of pre-written code that you can access and use in your code. The *math* module contains many mathematical functions. Some of the most commonly used functions are listed in the table below:

Function	Description	Example
math.floor()	Rounds a number down	*print(math.floor(30.3333))* will print 30.0.

math.ceil()	Rounds a number up	*print(math.ceil(30.3333))* will print 31.0
math.trunc()	Cuts off the decimal part of the float	*print(math.trunc(30.33333))* will print 30.0
math.sqrt()	Finds the square root of a number	*print(math.sqrt(4))* will print 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 print 3.141592653589793

Explore some more functions in the *math* module here. To be able to use the functions in the *math* module, add this line of code to the top of your program:

import math

A note from our coding mentor
Valerie

The further you progress on your Python journey, the more complex programs you will be able to make. Here are the top 5 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.*

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 as it will be much harder to read.

First, read **example.py**, open it using Notepad++ (Right-click the file and select 'Edit with IDLE).

- **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.

- You may run **example.py** to see the output. Feel free to write and run your own example code before doing the Task to become more comfortable with Python.

Compulsory Task 1

Follow these steps:

- 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

Follow these steps:

- 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
- Now ask for 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."

Optional Bonus Task

Follow these steps:

- Create a new Python file in this folder called *optional_task.py*.
- Ask the user to enter the lengths of all three sides of a triangle.
- Calculate the area of the triangle.
- Print out the area.
- *Hint: If side1, side2 and side3 are the sides of the triangle:*
    - *s = (side1 + side2 + side3)/2 and*
    - *area = √(s(s-a)*(s-b)*(s-c))*

Thing(s) to look out for:

1. Make sure that you have installed and setup all programs correctly. You have setup **Dropbox** correctly if you are reading this, but **Python or Notepad++** may not be installed correctly.
2. If you are not using Windows, please ask your mentor for alternative instructions.

Rate us

Share your thoughts

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

Think 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.