Hyperiondev

TASK

Introduction to Python

Visit our website

Introduction

WELCOME TO THE INTRODUCTION TO PYTHON TASK!

In this task, you are introduced to the Python programming language. Python is a widely-used programming language and is used extensively in data science. Many familiar organisations make use of Python, such as Wikipedia, Google, Yahoo!, NASA and Reddit.

This bootcamp uses Python as the basis for the data science tools and packages introduced to the learner. This is on account of two main reasons. First, Python is easy to learn with its natural construct, intuitive coding style, and simple setup. Second, Python has several built-in libraries and data science libraries that make it an ideal choice for a beginner to start working with data.

This task is a gentle introduction to Python, where you will be asked to create a simple program and, in doing so, will become familiar with the structure of a Python program.

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 log in 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

HyperionDev Team

Hope you're excited to start learning such a popular and fun programming language!

An interesting fact about the Python language: it is named after the Monty Python comedy group! Python was created by 'Benevolent Dictator For Life' Guido van Rossum in 1991, who now works for Dropbox. Van Rossum incidentally happens to be a big Monty Python's Flying Circus fan. His inspiration for Python stemmed from the desire to create a simple scripting language and his experience with the ABC programming language.

Remember that Python is a high-level programming language, along with other popular languages such as Java, C and Ruby. High-level programming languages are closer to human languages than machine code. They're called "high-level" as they are several steps removed from the actual code that runs on a computer's processor.

WHY PYTHON?

Python is a powerful, widely used programming language. Unlike Java, Python is a more recent, efficient and arguably faster programming language. The syntax (the way the code is written) is similar to Java.

Python is a simple yet powerful language. Looking at languages like C++ and Java can flummox and scare the beginner. But Python is intuitive with a natural way of presenting code. Python's succinctness and economy of language allow for speedy development and less hassle over useful tasks. This makes Python easy on the eyes and mind.

If you need more convincing, here are some more reasons why it makes sense to learn Python:

- **Python is in-demand:** As of 2018, the demand for Python skills is only growing. Python boasts the highest year-on-year increase in terms of demand by employers (as reflected in job descriptions put up online) as well as popularity among developers and data scientists. The demand for Python

is only set to grow further with its extensive use in analytics, data science and machine learning.

- **From child's play to big business:** While Python is simple enough to be learned quickly (even by kids), it is also powerful enough to drive many big businesses. Python is used by some of the biggest tech firms such as Google, Yahoo!, Instagram, Spotify and Dropbox, which should speak volumes about the job opportunities out there for Python developers.

YOUR FIRST PYTHON PROGRAM

Installing the Programming Environment

Before you get started, make sure you start using IDLE or Notepad++ to open all text files (.txt) and Python files (.py). Do not use the normal Windows Notepad or Mac TextEdit for reading code files. IDLE is the integrated development environment in which you can code Python. An Integrated Development Environment (IDE) is the software that programmers use to write, debug and execute their code.

Your content folder contains a sub-folder named 'Installers' which will help you download and set up Python on your machine. Once that is done, you can access the IDLE through the Start Menu on a Windows computer or by typing "idle" at the command line on a Mac or Linux computer.

The actual program that reads and executes instructions written in Python is the Python interpreter. This interpreter is embedded by default in larger programs, such as the programming environment IDLE, and this makes it comparatively easier to develop and run Python programs.

The basics

Programmers write statements of code to create 'programs', which are executable files that do something. Code can be written in different programming languages, such as Python, Java or C++.

After writing Python commands or code, we save them in a Python file. A Python file has the following file naming format: *filename.py* (where *filename* can be any valid filename and *.py* is the file extension in Windows).

Once you save a Python file, you can 'run' it. On doing so, the Python program you have written is executed and displays the outcomes that may result based on what

the code statements say. Information about how to 'run' Python files is provided in a later section of this lesson.

Note: In programming, a 'command' is an instruction given by a user telling a computer to do something. An 'argument' can be thought of as the value that you want the command to act on. Together a command and an argument are known as a 'statement'.

Syntax

The syntax is the "spelling and grammar rules" of a programming language. Python has a very simple and straightforward syntax.

In Python, indentation is very important. Python uses indentation to indicate a block of code. In the earlier version of Python (Python 2) you could use both 'tab' and 'space' buttons to create an indentation, but mixing the two has been disallowed in the latest version, Python 3. In Python 3, spaces are the preferred indentation method, so each indentation level is created using four spaces (hitting the spacebar 4 times).

Programmers leave comments on their code in order to document their work for future reference or to make it easier to understand for other programmers. In Python, a hash sign (#) begins a single-line comment. If you type #, all characters after and up to the end of the physical line will be treated as part of the comment, and the Python interpreter will ignore them.

Writing your first Python program

The simplest directive in Python is `print()`.

You may want your program to display (or output) information to the user. The most common way to view program output is to use the print command followed by one or more arguments.

For instance:

```
print ("Hello, World!")
```

Note that the argument is enclosed in double quotes ("..."). This is because **`Hello, World!`** is a string or list of characters. We will learn more about strings in the next lesson and task.

When you run this program, the computer will output the argument `Hello, World!`.

Getting input from the user

Sometimes you want a user to enter something through the keyboard. The `input()` command is handy for such cases.

The following `input()` command will show the text "Enter your name:" in the output box of the program, and the program will halt until the user enters something with their keyboard and presses Enter. After that, the program will ask the user to input their age with an output box showing the text “Enter your age:”. Thereafter, the print function will output the name and age entered by the user. Note that if you did not use print(name) and print(age), the program will still seek and store the user’s input but not display it.

```
name = input("Enter your name: ")
age = input("Enter your age: ")
print(name)
print(age)
```

Running a Python program

There are different ways to 'run' Python files such as the example.py file in this task folder which contains further guiding material and examples on creating Python programs.

The easiest way to run Python files is through a GUI (Graphical User Interface).

Simply go to the Windows Start Button (at the bottom left corner of your screen) and enter 'IDLE' into your windows search (the 'Search for programs and files' box). The program 'IDLE (Python GUI)' should appear in the search box. After you click on it to open it, go to the top left corner of this program and click File >> Open. Browse to locate your Dropbox folder, open the relevant task folder, and open **example.py**.

Now your code will open in the Python GUI IDLE. Press F5 on your keyboard to run the Python file and the output will appear in a separate Python output (Python Shell) window. If there is an error in your code, the code won't run, and the error will be printed out in the Python Shell window.

You can use this method to run any Python file (file with a .py extension). We advise that you complete all tasks and open all example files in IDLE from now on. (You can create a shortcut to the IDLE on your desktop so that you can access it faster.) You can also use Notepad++ to view the example.py files, but Notepad++ can only be used to view the text of code. You can't run programs from within Notepad++.

Note: If you can't find the IDLE program on your computer, you probably did not install Python correctly. Contact your mentor for assistance.

A note from our coding mentor
Ridhaa

The Zen of Python, written in 1999 by Tim Peters, mentions all the software principles that influence the design of the Python language.

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
*Although never is often better than *right* now.*
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

Ever need to recall these principles? Try entering this into your Python interpreter!

```
>> import this
```

Now that you have grasped the basics of programming in Python, it's time to gain an understanding of variables: what they are, how to declare and assign values to them, and how to convert between different types of variables.

VARIABLES

In computer programming, a 'variable' refers to the storage location for data in a program.

Computer programs are able to solve complex problems by executing a lot of code in the background. In doing this, the computer needs a way to store values in its memory, and this is where variables come in. A variable is simply a way to store information. It can be thought of as a type of "container" that holds information.

Each variable has a name which can be used to refer to some stored information known as a 'value'. The content that a variable stores is the **data**, and the type of data is called the **data type**. The equal sign (=) is used to assign values to variables.

Unlike some other programming languages, variables do not need to be 'declared' in Python before being assigned a value. A variable is created as soon as you first assign a value to it.

In programming, we use variables to hold values that can later be changed. For example:

```
num_books = 5
num_books = 10
print(num_books)
# output: 10
```

Here, the variable numBooks is first assigned the value 5. Later, the value is substituted and becomes 10. When you print out the value of the variable numBooks, the program outputs 10.

In maths, variables have names of letters like x and y; in programming, variables can be given any name you desire (although there are **clean coding conventions on variable naming** which should be kept in mind). It is best to use meaningful names that are relevant to the program or calculation you are working on, for example, *num_learners*, which could contain the number of learners in a class or *total_amount,* which could store the total value of a calculation.

Remember that variable names are case-sensitive, which means *Apple* is not the same as *apple*. For instance, see the error message returned below.

```
num_books = 5
print(numbooks)
Traceback (most recent call last):
  File "script.py", line 2, in <module>
    print(numbooks)
NameError: name 'numbooks' is not defined
```

In maths, variables only deal with numbers but in programming, we have many different types of variables and each variable type deals with a specific set of information.

Types of Variables

There are 4 major variable types that we will learn about: String, Integer, Float and Boolean.

- **String:** A string consists of a combination of characters. It can be used to store, say, the surname, name, or address of a person.

- **Integer:** An integer is a whole number or number without a decimal or fractional part. For example, it can be used to store the number of items you would like to purchase, or the number of students in a class.

- **Float:** We make use of a Float data type when working with numbers that contain decimals. For example, it can be used to store measurements or monetary amounts.

- **Boolean:** A Boolean data type can only store one of two values, namely TRUE or FALSE.

Your requirements will determine which variable you need to use. For example, when dealing with money or mathematical calculations, you would likely use integers or floats. When dealing with sentences or displaying instructions to the user, we would make use of strings. When dealing with scenarios that have only two possible outcomes, we would use booleans, as the scenario would only either be True or False.

Take note:

Here are a few common questions from learners when they're starting out with variables:

Q: If I define a variable and assign it a 'string' value, can I assign another data type (say, an integer value) to that variable later?

A: Yes. In Python, a variable may be assigned a value of one type, and then later assigned a value of a different type.

Q: Do you declare the data type of a variable in advance?

A: No, the data type can be neglected when creating a variable in Python. This is because Python is 'dynamically typed'.

Q: How does Python know the difference between variables?

A: Python detects the type of a variable by reading how data is assigned to the variable.

- *Strings are detected by quotation marks (" ").*
- *Integers are detected by the lack of quotation marks and presence of digits (or another accepted number format) but no decimal.*
- *Floats are detected by the presence of decimal point numbers.*

Remember that types can be converted from one to another, and you need to take care when setting a string with numerical information. For example:

```
#Example A

num = 10
num - 2
8

#Example B

num_str = "10"
num_str - 2
Traceback (most recent call last):
TypeError: unsupported operand type(s) for -: 'str' and 'int'
```

Watch out in Example B here! Since you defined 10 within quotation marks, Python figures this is a String. It's not stored as an integer even though 10 is a number, as numbers can also be displayed as a String if you put them between

quotation marks. Now, because 10 is declared as a String here, we will be unable to do any arithmetic calculations with it.

Casting

Putting the 10 in quotation marks will automatically convert it into a string but there is a more formal way to change between variable types. This is known as **casting** or **type conversion**.

Casting in Python is pretty simple to do. All you need to know is which data type you want it to convert to and then use the corresponding function.

- `str()` converts variable to a string
- `int()` converts variable to a integer
- `float()` converts variable to a float

```
number = 10
number_str = str(number)
print(str(number) + number_str)
```

This example converts `number_str` into a string so that it can be easily printed without the need for quotation marks. The plus sign is not performing addition here, it is joining two strings together. Try running the code above in your IDLE to see the output.

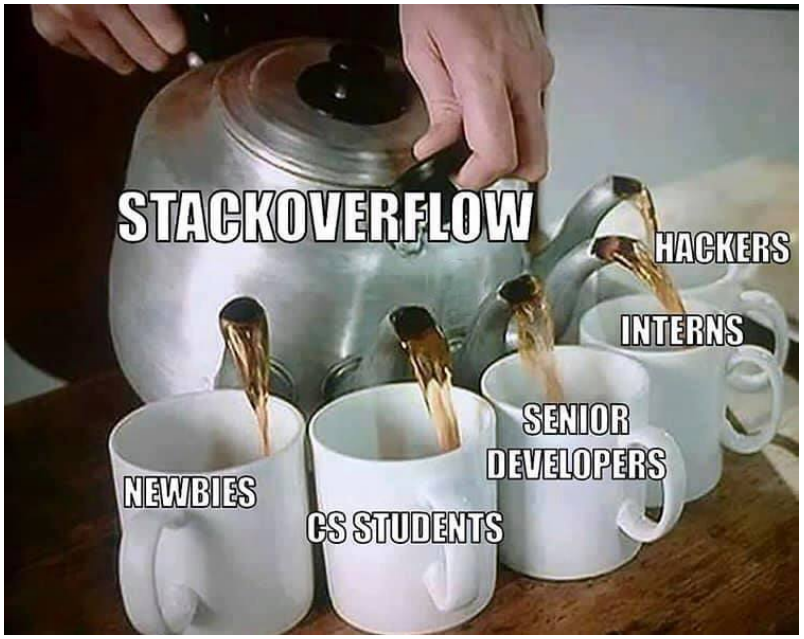
You can also convert the variable type entered via `input()`. By default, anything entered into an `input()` is a string and to convert it to a different data type we would simply need to use the desired function. Try running the code below in your IDLE to see the output.

```
no_days = int(input("How many days did you work this month?"))
pay_per_day = float(input("How much is your pay per day?"))
salary = no_days * pay_per_day
print("My salary for the month is USD " + str(salary))
```

Take note:

Coding can be an intimidating process, especially for a beginner. Be sure to look up anything you feel unsure about — the internet is full of very clever and helpful

*people! Remember that looking up your problems is something that ALL programmers do, from brand new coders to the most advanced senior developer. Don't believe me? Check out **this** post.*



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, 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.
- This may take some time at first as it is different from other (natural) languages, but persevere! Your mentor is here to assist you along the way.

Compulsory Task 1

Follow these steps:

- Create a new Python file in this folder called **details.py**
- Use the Input function to get the following information from the user.
    - Name
    - Age
    - House number
    - Street name
- Print out a single sentence containing all the details of the user. You will need to join strings to do this, using the plus symbol like in the example in the 'casting' section of this document.

  For example:

  This is John Smith who is 28 years old and lives at house number 42 on Hamilton Street.

Compulsory Task 2

Follow these steps:

- Create a new Python file in this folder called **conversion.py**
- Declare the following variables:
    - num1 = 99.23
    - num2 = 23
    - num3 = 150
    - string1 = "100"
- Convert them as follows:
    - num1 into an integer

    - num2 into a float
    - num3 into a String
    - string1 into an integer
- Print out all the variables on separate lines.

Completed the task(s)?

Ask your mentor to review your work!

Review work

Thing(s) to look out for:

1. Make sure that you have installed and set up Dropbox correctly on your machine, and that it is syncing.
2. Make sure that you have installed and are able to work on Python IDLE.
3. If you are not using Windows and facing trouble attempting the tasks, please ask your mentor for alternative instructions.

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.