TASK

Your First Computer Program

Visit our website

Introduction

WELCOME TO YOUR FIRST COMPUTER PROGRAM TASK!

In this task, you are introduced to the Python programming language. Python is a widely used programming language. It is consistently ranked in the top 10 most popular programming languages as measured by the TIOBE Programming Community Index. Many familiar organisations make use of Python, such as *Wikipedia*, *Google*, *Yahoo!*, *NASA* and *Reddit* (which is written entirely in Python).

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.

This task is a gentle introduction to Python where you will be asked to create a simple program. In doing so, you 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 an expert code reviewer to get support on any aspect of your course.

The best way to get help is to log in to Discord at **https://discord.com/invite/hyperdev** where our specialist team is ready to support you.

Our team 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

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

A brief history of Python:
Python is named after the Monty Python comedy group. It was created by '**Benevolent Dictator For Life**' Guido van Rossum in 1991, who now works for Dropbox. At the time when Guido van Rossum began implementing the Python language, he was also reading the published scripts from *Monty Python's Flying Circus* (a BBC comedy series from the seventies). His inspiration for Python stemmed from the desire to create a simple scripting language and his experience with the ABC programming language.

Guido van Rossum

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 very similar to Java.

Here are a few more reasons to use Python:

- **Simple, yet powerful:** 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 allows for speedy development and less hassle over useful tasks. This makes Python easy on the eyes and mind.

- **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.

- **Python is on the web:** python is a very appealing language of choice for web development. Sites such as *Pinterest* and *Instagram* make use of the versatility, rapidity and simplicity of Django (a web development framework written in Python).

- **Even *Dropbox* was built using Python:** *Dropbox* must save massive quantities of files while supporting massive amounts of user growth. 99.9% of *Dropbox* code is written in Python! Using Python has helped *Dropbox* gain more than a hundred million users. Using only a few hundred lines of Python code, they were able to scale up massively in user numbers. Learn from *Dropbox* and use Python!

ZEN OF PYTHON

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
```

A note from Masood...

🔥**Know this – Python is hot!**🔥

The demand for Python programmers 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. Python developers are one of the highest-paid categories of programmers! The demand for Python is only set to grow further with its extensive use in analytics, data science, and machine learning.

INSTALLING VIRTUAL STUDIO CODE

Before you get started, we suggest you use Virtual Studio (VS) Code to open all text files (.txt) and Python files (.py). Do not use the normal Windows notepad for reading code files.

VS Code is an IDE, which stands for Integrated Development Environment. An Integrated Development Environment 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.

SETTING UP YOUR DEVELOPMENT ENVIRONMENT

Please read the following instructions in full before following them:

1. Visit **https://code.visualstudio.com/**
2. Download the version of VS Code that matches your operating system (OS). Alternatively, you can follow the instructions stated at the following links for the corresponding operating system families:
    a. macOS: **https://code.visualstudio.com/docs/setup/mac**
    b. Linux: **https://code.visualstudio.com/docs/setup/linux**
    c. Windows: **https://code.visualstudio.com/docs/setup/windows**

3. Unix-like operating systems such as macOS and Linux often come with a pre-installed version of Python. It is generally discouraged to use the distributions of Python that are shipped with macOS as they may either be outdated or have customisations that might give you issues further down the line. Please follow the guidelines at:

   https://code.visualstudio.com/docs/python/python-tutorial.

   a. Ensure that if you are on Windows or macOS, you have installed the latest stable version of Python using the prescribed means on the link above.

   b. Ensure that if you are on Linux and the prepackaged Python version is not the latest stable version, you get a package provider for your operating system that uses the latest stable version. Being behind by 1 or 2 minor versions is fine on operating systems such as Fedora. However, on operating systems such as Ubuntu, we strongly recommend that you use a PPA (Personal Package Archive) such as:

      https://launchpad.net/~deadsnakes/+archive/ubuntu/ppa.

      i. Please avoid the flatpak or snap versions as they give you troubleshooting problems. Only proceed with flatpak if you are sure of how they work.

   c. For all operating systems, ensure that your environment paths are up-to-date with regards to your installation.

   d. Per the guidelines linked above, ensure that you install the latest stable version of Microsoft's Python extension available from **https://marketplace.visualstudio.com/items?itemName=ms-python.python** so that you get tooltips and other useful tooling that help you as your program.

4. Use the compulsory task to ensure that your setup is working as expected. If unsure, please check with an academic staff member or a peer.

5. Use **https://code.visualstudio.com/docs/languages/python** to learn how to use Visual Studio Code with Python. If you've never programmed before, we strongly recommend you watch the videos.

6. There are a range of other editors that you can use such as Vi, emacs, Notepad++, and PyCharm, but we cannot guarantee that your peers will be familiar enough with them to assist you with them or that the academic staff members will be able to consistently review your work.

7. If you’re concerned about opt-out telemetry with Visual Studio Code, please turn it off by using the instructions from:

   https://code.visualstudio.com/docs/getstarted/telemetry#_disable-telemetry-reporting.

WHAT IS PROGRAMMING?

Programmers write statements of code to create *programs*. Programs are executable files that perform the instructions given by the programmer.

Code can be written in different programming *languages*, such as Python, Java, and C++. In this course, you will start by learning Python.

After writing Python commands or code, you need to save them in a Python file. A Python file has the following file naming format:

*filename**.py***

The filename can be any valid filename and **.py** is the file extension.

You can then 'run' the Python file. In this process, 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 given in the example file (**example.py**) that accompanies this task. We will now show you how to write some basic code in Python, and perform some basic operations.

THE *PRINT()* FUNCTION

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* function. In this explanation, we’re going to talk about two new concepts, parameters and arguments. A **parameter** is a variable in a method definition. When a method is actually called, the **arguments** are the *data you pass into* the method's parameters.

To use *print*, we enter the *print* command followed by one or more arguments in brackets. In programming, a **command** is an instruction given by a user telling a computer to do something. Together a command and an argument are known as a **statement**. Consider the Python statement below:

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

When you run this program, the computer will output the argument "Hello, World!" that was passed in to the input parameter. Note that the argument is enclosed in double quotes ("..."). This is because "Hello, World!" is a string or a list of characters. These are data types that will be discussed in more detail in later tasks.

The Python Shell (the window that is displayed when you run a Python program) only shows the output of the program. Other statements in your code will be executed but not displayed in the Python Shell.

SYNTAX RULES

All programming languages have *syntax* rules. Syntax is the "spelling and grammar rules" of a programming language and determines how you write correct, well-formed statements.

A common syntax error you could make above is forgetting to add a closing quotation mark ("). Remember that all opening quotation marks (") require a closing one! Another common syntax error that you could make in the above example is forgetting to add a closing bracket ')'. Remember that all opening brackets '(' require a matching closing one, ')'!

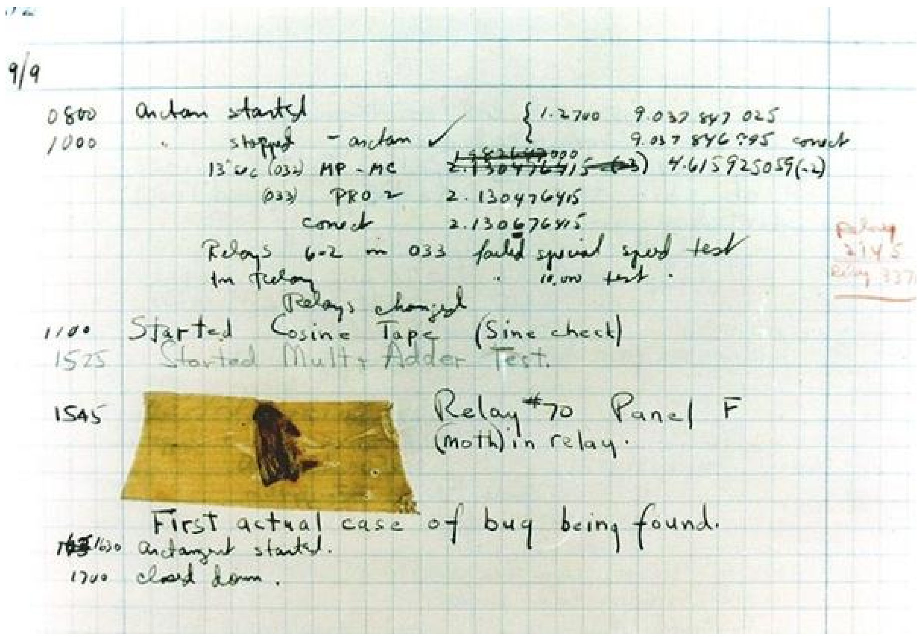
Any program you write must be exactly correct. All code is case sensitive. This means that *'Print'* is not the same as *'print'*. If you enter an invalid Python command, misspell a command, or misplace a punctuation mark, you will get a syntax error when trying to run your Python program.

Errors appear in the Python shell when you try to run a program and it fails. Be sure to read all errors carefully to discover what the problem is. Error reports in the Python shell will even tell you what line of your program had an error. The process of resolving errors in code is known as *debugging*.

A note from Riaz...

Sorry to interrupt, but did you know that the first computer "bug" was named after a real bug? Yes, you read that right! While the term "bug" in the meaning of a technical error was first coined by Thomas Edison in 1878, it was only 60 years later that someone else popularised the term.

In 1947, Grace Hopper, a US Navy admiral, recorded the first computer 'bug' in her logbook as she was working on a Mark II computer. A moth was discovered stuck in a relay and thus hindering the operation. She proceeded to remove the moth, thereby 'debugging' the system, and taped it in her logbook. In her notes, she wrote, "First actual case of bug being found."



9/9

0800 Anctan started
1000 " stopped - anctan ✓ {1.2700 9.037 847 025
9.037 846 795 conect
13" uc (032) MP - MC ~~1.982147000~~ 2.130476415 (-3) 4.615925059(-2)
(033) PRO 2 2.130476415
conect 2.130676415
Relays 6-2 in 033 failed special speed test
In relay 10,000 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult+ Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1630 Antangent started.
1700 closed down.

Relay 2145
Relay 3370

- **Riaz Moola,** *Founder and CEO*

HOW TO GET INPUT

Sometimes you want a user to enter data that will be used by your program through the keyboard. To do this, use the *input* command.

The *input* command, in the example below, will show the text "Enter your name: " in the output box of the program. The program will then halt until the user enters something with their keyboard and presses enter.

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

The variable *name* stores what the user entered into the box as a **string** (we'll cover strings in more depth in an upcoming task). Storing and declaring variables doesn't produce any output.

Instructions

This lesson is continued in the **example.py** file provided in this task folder. Open this file using VS Code. The context and examples provided in **example.py** should help you understand some simple basics of Python.

You may run **example.py** to see the output. The instructions on how to do this are inside the file. Feel free to write and run your own example code before attempting the task, to become more comfortable with Python.

Try to write comments in your code to explain what you are doing in your program (read the **example.py** file for more information).

Compulsory Task 1

Follow these steps:

- Create a new Python file in this folder called **hello_world.py**
- Inside this file, write Python code to take in a user's name using *input()* and then print out the name.
- Also, take in a user's age using the same method and print out their age.
- Finally, print out a new line and the string "Hello World!"

Compulsory Task 2

Follow these steps:

- Create a new Python file in this folder called **menu.py**
- Write a program that asks the user to order their 3 favourite food items on a menu using *input()*. Store them as variables *item1*, *item2* and *item3*.
- Print each item separately.
- E.g. your program could print:

  Order confirmation! You have ordered:

  Chicken nuggets

  Fish and chips

  Spaghetti bolognaise

Thing(s) to look out for:

1. Make sure that you have installed and set up all programs correctly. You have set up **Dropbox** correctly if you are reading this, but **Python or your editor** may not be installed correctly.
2. .If you are not using Windows, please ask one of our expert code reviewers for alternative instructions.

Completed the task(s)?

Ask an expert code reviewer to review your work!

Review work

Rate us

Share your thoughts

Hyperion 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.