Hyperion Development

Task: Python Basics Application

www.hyperiondev.com

Introduction

Let's make use of what you know!

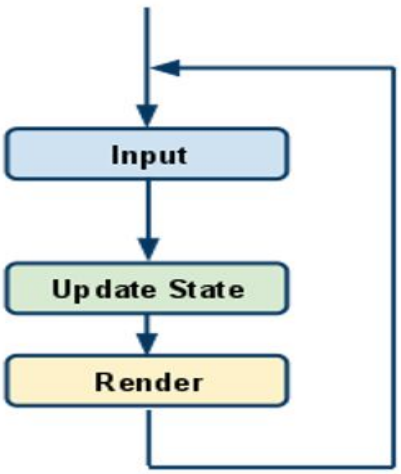
Now that you know some Python it's time to write a small program. You may wondering how you could possibly write a program such as Microsoft Word with the few concepts that we have covered so far, but let's not get ahead of ourselves. Baby steps. With that in mind, let's see what we CAN do!

-The Hyperion Team

Programming games is a great way to learn how to code. Look at any game and think to yourself: How can I do that? What functions will I need? What data types and structures would I use?

For example, using control structures you could perhaps make a turn-based strategy game. What about games like Sudoku or Hangman even?

The workings of a game can be represented as follows:



The above diagram represents a 'game loop'. This loop is executed many times a second. Inside the loop the game logic is executed and the screen is updated and rendered accordingly.

Even if you do not have all the programming tools at your disposal yet, it is often useful to prime the mind to think in this way.

Using the pygame module is one way to make games in Python. Visit the pygame website to find out more information about pygame.

- **The Hyperion Team**

A little bit about interest:

No no, not interest as in if you care for something, but interest as in financial interest! Interest occurs in almost all financial "happenings", whether it be on a loan which ends up with you paying more to the bank, or on an investment which ends up with you earning more. There are two main streams of interest, compound and simple interest.

Simple interest is continually calculated on the initial amount invested, and is only calculated once per year. This interest amount is then added to the amount that you initially added (known as the Principal amount).
An example of this is if you invest R1000 and 10%, the first year you will earn R100 interest (R1000 * 0.10) giving you R1100. The next year the interest is still calculated on the principal amount (R1000) giving you another R100, making a total of R1200.

Compound interest is different in that the interest is calculated on the current total, known as the Accumulated amount.
To use the above example, imagine you invest R1000 at 10% compounded once a year. The first year you will earn R100, giving you an Accumulated amount of R1100. The second year you will earn interest on the Accumulated amount (1100 * 0.10), to earn R110 interest, giving you R1210.

If this doesn't make a huge amount of sense, don't worry too much as the above is just a brief background to what you will be doing in the compulsory tasks. If you'd like to find out more feel free to do a bit more research! The needed formulae are given below.

Instructions

Feel free at any point to refer back to previous material if you get stuck. Remember that if you require more assistance our tutors are always more than willing to help you!

- We are going to create an investment calculator to see what return you will get if you were to put money into an investment at a certain interest rate for a certain amount of time.
- Complete the compulsory task below to progress to the next task.
- Open the example.py files using Notepad++ (Right click the file and select 'Edit with Notepad++') to read them.
- **example.py** has recap content for you to aid with the Compulsory Task and remind you of key concepts, as well as a new structure known as a "while loop". This is a widely used concept in programming and something definitely worth 'while' getting used to using!
- As a bonus treat open the **example.py** file in the **game** folder.
    - example.py contains a very simple game coded with pygame. The player can move up and down with the arrow keys and must avoid the programming challenges to win.
    - Do not worry if you do not understand some of the concepts yet as they will be covered in upcoming tasks.
    - Most of the concepts should, however, be familiar from the previous tasks.
    - This should show you how powerful the tools are that you have learnt thus far.
    - You will need to download pygame. Instructions on how to do so are inside the example.py file.
    - You will get to add more functionality and complexity to the game in later tasks as you learn more programming concepts. For example more enemies, lives and cool effects.

Compulsory Task

Follow these steps:

- Create a new Python file in this folder called **InvestmentCalculator.py**.
- At the top of the file include the line:
    - import math
- Ask the user to input:
    - The amount that they are depositing, stored as 'P'.

- The interest rate (as a percentage), stored as 'i'.
- The number of years of the investment. stored as 't'.
- Then ask the user to input whether they want "simple" or "compound" interest, and store this in a variable called 'interest'.

- Only the number of the interest rate should be entered - don't worry about having to deal with the added '%', e.g. The user should enter 8 and not 8%.
- Depending on whether they typed "simple" or "compound", output the appropriate amount that they well get after the given period at the interest rate. Look below in "additional information" for the formulae to be used.
- Print out the answer!
- Try enter 20 years and 8 (%) and see what a difference there is depending on the type of interest rate!

Additional Information:

- Don't forget to cast the input to ints for the calculations!
- 'r' below is the interest entered above divided by 100, e.g. if 8% is entered, then r is 0.08.
- Simple interest rate is calculated as follows:
  - A = P(1 + r * t)
  - The Python equivalent is very similar:
    - A =P*(1+r*t)
- Compound interest rate is calculated as follows:
  - A = P(1 + r) ^ t
  - The Python equivalent is slightly different:
    - A = P* math.pow((1+r),t)

Still need help?

Just write your queries in your comments.txt file and your tutor will respond.

Task Statistics

Last update to task: 06/01/2015.
Course Content Manager: Riaz Moola.
Task Author: Brandon Haschick and Sinead Urisohn.
Task Feedback link: Hyperion Development Feedback.