In this module we will be covering the concepts of how to design a program - life cycle. We will be looking at Input-process-output logic. We will learn some of the basic PYTHON commands, and will looking at "reading" input, producing output and performing calculations.

Program Development Life Cycle

One of the MOST important things a programmer needs to learn is the Program Development Life Cycle. Just like we should follow steps when building a house - we need to follow steps in building a computer program. The Program Development Life Cycle has 5 major steps:

1. Design the program - this step is done BEFORE starting to write your program. You need to a) understand the task that the program needs to perform, and 2) determine the steps (LOGIC) in their correct order that must be taken to perform the task. You need to break down the task into a series of steps - IN THE CORRECT ORDER - this is called an ALGORITHM!!! You then may create a "visual" to help you - sometimes you can use a computer flowchart (see chapter and link 1), or you can do pseudocode (see chapter and link 2). Either a flowchart or pseudocode can map your statements into what the computer needs to do. Many programmers try to skip this step - DON'T fall into that trap!!!!

   a) write out the steps in their correct sequence order

   b) create either a flowchart or pseudocode

2. Write the code - you will be using PYTHON as your programming language to write your computer programs

3. Interpret or compile your program - we will be using the interpreter; Correct any SYNTAX errors and put back through the interpreter until all SYNTAX errors are corrected.

4. Execute/Run the program - the text calls this "testing" - you are seeing if the correct Output is being produced. If the correct output is okay you are done, if not - go to step 5.

5. Correct logic errors - go back to your original program, make corrections in PYTHON language, put back through the interpreter and re-run (execute).

The Overall "logic" of all programs

All computer programs follow the same basic 3-step process: input-process-output. The computer program asks for input data to use in the program, the program processes the data (does to the data what you want it to do), then the program produces the outputs that you want (screen, printer, disc, etc.).

Functions

In PYTHON (and many other languages) your keywords are actually prewritten code that is stored as part of the PYTHON language. You have a one word "function" that will tell the computer to perform some type of action. The pre-written code is "called" by the computer program. In this case - to "call" a function is simply to tell the computer to follow that functions pre-written code.

Our first PYTHON function

The first PYTHON function will be "print". The print function will display something on the computer screen for you to see. The syntax for the print function statement is:

>>> print ('what you want to print')

example: >>> print ("My name is Mary Louise Jones') *********Notice that since you want actual letters printed you MUST put these letters in single quotation marks (this is part of the SYNTAX rules of PYTHON).

Strings and Literals

Sequences of letters, etc. are called "strings". A "string literal" is what we have in the single quotation marks. You can also use the double quotation marks also in PYTHON. When you need to use a quotation mark as part of your string literal - you use double quotation marks around the whole string and embed the single quotation mark where it needs to be in the string.

example: >>> print ("Don't forget to meet me at 1:00 today")

NOTE - you can use numbers inside any string literal and they are treated as "alphabetic types of numbers - not ones to do mathematical functions with".

Comments

You MUST use comments throughout your PYTHON program to explain to others who might look at your computer program what you intended to do in this part of the program. The PYTHON interpreter ignores comments (doesn't "execute" them).

A comment starts with the # character.

example: #This program prints out the date

print ('January 30, 2017')

Variables

A variable is a name that you create that will represent a value that will be stored in the computer's memory at the time the program is executing. Your program will use variables to use data that has previously been stored that you want to use for your program, or new data that you want to use for your program. REMEMBER - you come up with your own variable names for your program. Choose variable names that make sense. For example, if you have a number that represents someone's age - you would probably use "age" as your variable name.

We need to get actual data into the variable. Remember you just have the NAME of the variable - you need to put some data into it. This is called an Assignment statement - to put a value into a variable name.

variable = expression

example: age = 25 you CAN'T change this order - 25 = age is a SYNTAX error

A program example:

>>> # This is my first program

>>> width = 10 press the enter key on your keyboard

>>> length = 5 press the enter key on your keyboard

>>> print ('width') press the enter key on your keyboard

width (will be displayed on your screen)

>>> print (width) press the enter key on your keyboard - NOTICE NO quotation marks - you are asking to print the value inside the variable WIDTH

10

>>>

Variable Naming Rules

You must follow the rules when you create the names of your variables:

1. You can't use one of PYTHON's key words

2. A variable name cannot contain spaces

3. The first letter MUST be one of the letters A-Z, a-z or the underscore (_) character

4. After the first letter you may use the letters A-Z, a-z, digits 0-9 or the underscore

5. Uppercase and lowercase characters are DIFFERENT. This means the variable name Age and age are NOT the same!!!

Displaying Multiple Items with print Function

Within your parethesis for your print statement you can combine the literal part of the statement - things within quotation marks, and the actual name of the variable (so you get the value contained in the variable printed):

example: #This program prints out my room number

```
room = 1750

print ('I am staying in room number', room)
```

I am staying in room 1750

Reassigning values to the same variable in your program

Within a single program - you can reassign a different value to your variable after assigning the original value to the variable.

Numeric Data Types and Literals

In PYTHON we have several different "data types". These types of data include - int is used for integers, and float is used for real numbers (with decimals) for numbers.

room = 1750

dollars = 5.76

We also have the str for our string variables.

first_name (notice you can use the underscore character in your variable name) = 'Barry'

Reading Input from the Keyboard

We can get data "into" our programs by either typing is from the keyboard (direct entry), or from stored data.

The SYNTAX for reading data that is entered from the keyboard is: *variable* = input (*prompt*)

example: age = input('What is your age? ')

example: >>> age = input('What is your age? ') - notice I left some blank spaces before the last quotation mark -

press Enter

What is your age? **43** displays on the screen, type in 43 and press Enter

>>> print (age) press Enter

43 displayed on the screen

Reading numbers with Input Function

When data is typed from the screen as input it ALWAYS comes in as a string. Thus is we input the number 72 is actually comes in as the string '72'. If we wanted to do any type of math with this value, we need to first convert it to the integer or real number equivalent. There are two built in functions to accomplish this task.

int(*item)* will send the string value - this is called Passing an Argument to the function - and return the equivalent integer value

float (*item*) will send the string value - passing the argument to the function - and return the equivalent real number value

Calculations

Python has many mathematical operators that can be used:

+ addition

- subtraction

* multiplication

/ division gives result as a floating point value

// integer division gives result as an integer value

% remainder

** exponent

Mathematical Operator Precedence

You need to remember the precedence order for your math operators; these are the same that you learned in math classes.

1. exponentiation ** is done first

2. multiplication, division, and remainders *, /, //, % are done second

3. addition and subtraction done third

REMEMBER - you use parenthesis to change this order!!!!

Math formulas and programming statement

Remember that you have to follow the SYNTAX rules for everything. The normal way we write math formulas needs to be written so that the computer understands the order properly.

Escape Characters

An escape character is a special character that is preceded with a backslash (\) within a string literal. Please see page 67 to see the various escape types of characters. These allow you to better control your output from your print function.