High Impact Skills Development Program

AI & Data Science

Lab 04: Functions , Random Numbers & Math Functions

Lab 03: Functions

Introduction

This lab is designed to develop the understanding of students with User-defined functions.

Objectives

The objective of this lab is to design solutions using Library (Built-in) and User-defined functions in Python Scripted Mode.

Tools/Software Requirement

Python IDLE

Description

Python comes with many Library (Built-in) functions that perform common operations.

```
>>> #Absolute Function (abs())
>>> abs(-9) #Function Call
9
>>> abs(-3.3) #Function Call
3.3
```

```
>>> #Two arguments: pow(x,y)
>>> pow(2,4)
16
>>> #Three arguments: pow(x,y,z)
>>> pow(2,4,3)
1
```

- $pow(a, b) = a^b$
- $pow(a, b, c) = a^b \% c$

```
>>> #Round Function (round()), which rounds a
floating-point number to the nearest integer
>>> round(3.8)
4
>>> round(3.3)
3
>>> round(3.5)
4
>>> round(-3.5)
-4
>>> round(0.5) #Round towards even number
0
```

If you're not sure what a function does, try calling built-in function help, which shows documentation for any function:

```
>>> help(abs)
Help on built-in function abs in module built-ins:
abs(...)
abs(number) -> number
Return the absolute value of the argument
```

Apart from built-in (Library) functions, we can make functions (user-defined functions) and can use it many times.

Description:

- Function blocks begin with the keyword **def** followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

```
>>> def quadratic(a, b, c, x):
        first = a * x ** 2
        second = b * x
        third = c
        return first + second + third


>>> quadratic(2, 3, 4, 0.5)
6.0
>>> quadratic(2, 3, 4, 1.5)
13.0
```

Lab Tasks

Using only the programming techniques that you have learned so far, perform the following tasks:

Task 1: Define a function called `hypotenuse` that calculates the length of the hypotenuse of a right triangle when the other two sides are given. Use this function in a program to determine the length of the hypotenuse for each of the following triangles. The function should take two arguments of float type and return the hypotenuse as float type too. The sample output is as following:

```
Enter the sides of the triangle: 3.0 4.0
Hypotenuse:  5.0

Enter the sides of the triangle: 5.0 12.0
Hypotenuse:  13.0

Enter the sides of the triangle: 8.0 15.0
Hypotenuse:  17.0
```

Task 2: Write a function `distance` that calculates the distance between two points (x1, y1) and (x2, y2). All numbers and return values should be of floating type. Use this function in your program. The sample output is as following:

```
Enter the first point: 3 4
Enter the second point: 0 0
Distance between ( 3.00, 4.00 ) and ( 0.00, 0.00 ) is 5.00
```

Task 3: Write a function `reversed` that takes an integer value (between 1-9999) and returns the number with its digits reversed. For example, given the number 7631, the function should return 1367. Demonstrate the use of this function in your program with the following sample output.

```
Enter a number between 1 and 9999: 7631
The number with its digits reversed is: 1367
```

Task 4: Write a function `is_prime` that accepts an integer as argument and returns `True` if the number is prime and `False` otherwise. Use this function in your program to print on screen all the 4 digit prime numbers.

Task 5: Write a function `qualityPoints` that inputs a student's average and returns 4 if a student's average is 90-100, 3 if the average is 80-89, 2 if the average is 70-79, 1 if the average is 60-69, and 0 if the average is lower than 60. Use this function in your program that reads average from the student and prints on screen the corresponding GPA as shown below.

```
Enter the student's average: 92
92 on a 4 point scale is 4

Enter the student's average: 87
87 on a 4 point scale is 3

Enter the student's average: 75
75 on a 4 point scale is 2

Enter the student's average: 63
63 on a 4 point scale is 1

Enter the student's average: 22
22 on a 4 point scale is 0
```

Task 6: Write a function is_`bouncy` that accepts an integer as argument and returns `True` if the number is bouncy and `False` otherwise. Use this function in your program to print on screen all the 4 digit bouncy numbers along with the total count of the 4-digit bouncy numbers.

Task 7: Write a function called `number_of_factors` that takes an integer and returns how many factors the number has.

Task 8: Write a function called `binom` that takes two integers n `and` k and returns the binomial coefficient $\binom{n}{k}$. The definition is:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Task 9: Write a program that plays the game of "guess the number" as follows: Your program chooses the number to be guessed by selecting an integer at random in the range 1 to 1000. The program then types:

```
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
```

The player then types a first guess. The program responds with one of the following:

```
1. Excellent! You guessed the number!
   Would you like to play again (y or n)?
2. Too low. Try again.
3. Too high. Try again.
```

If the player's guess is incorrect, your program should loop until the player finally gets the number right. Your program should keep telling the player *Too high* or *Too low* to help the player "zero in" on the correct answer. The sample run of the game should be as following:

```
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too low. Try again.
? 750
Too high. Try again.
? 625
Too low. Try again.
? 687
Too high. Try again.
? 656
Too low. Try again.
? 671
Too low. Try again.
? 678
Too high. Try again.
? 675
Too high. Try again.
? 673
Too high. Try again.
? 672

Excellent! You guessed the number!
Would you like to play again?
Please type ( 1=yes, 2=no )? 2
```

- An integer number is said to be a perfect number if its factors, including 1 (but not the number itself), sum to the number. For example, 6 is a perfect number because $6 = 1 + 2 + 3$.

  Write a function `perfect` that determines if the parameter number is a perfect number or not. The function should take an integer argument and returns Boolean `True` if the integer is perfect and Boolean `False` otherwise.

  Use this function in a program that determines and prints all the perfect numbers between 1 and 1000.

- Write a function `int2binary` that accepts an integer as argument and returns its equivalent binary number using while loop and a list

  Use this function in a program that prints on screen the binary equivalent of numbers from 100 till 1000 in a tabular form.

  [Hint: Use the division-by-2 algorithm]

- Write a function called `root` that is given a number x and an integer n and returns $x^{1/n}$. In the function definition, set the default value of n to 2.

- An application of *`math.floor`* function is rounding a value to the nearest integer. The statement

  ```
  y = floor( x + .5 )
  ```

  will round the number x to the nearest integer, and assign the result to y. Write a program that reads several numbers and uses the preceding statement to round each of these numbers to the nearest integer. For each number processed, print both the original number and the rounded number.

```
Enter a floating point value: 1.5
1.500000 rounded is 2.0

Enter a floating point value: 5.55
5.550000 rounded is 6.0

Enter a floating point value: 73.2341231432
73.234123 rounded is 73.0

Enter a floating point value: 9.0
9.000000 rounded is 9.0

Enter a floating point value: 4
4.000000 rounded is 4.0
```

- Function `floor` may be used to round a number to a specific decimal place. The statement

  ```
  y = floor( x * 10 + .5 ) / 10
  ```

  rounds x to the tenths position (the first position to the right of the decimal point). The statement

  ```
  y = floor( x * 100 + .5 ) / 100
  ```

  rounds x to the hundredths position (i.e., the second position to the right of the decimal point). Write a program that defines four functions to round a number x in various ways

a) roundToInteger(number)
b) roundToTenths(number)
c) roundToHundreths(number)
d) roundToThousandths(number)

For each value read, your program should print the original value, the number rounded to the nearest integer, the number rounded to the nearest tenth, the number rounded to the nearest hundredth, and the number rounded to the nearest thousandth.

```
How many numbers do you want to process? 1
Enter number: 8.54739
8.547390 rounded to an integer is 9.000000
8.547390 rounded to the nearest tenth is 8.500000
8.547390 rounded to the nearest hundredth is 8.550000
8.547390 rounded to the nearest thousandth is 8.547000
```