Programming with Python Feb 10,11 2025

Note: Brief Summary of contents discussed.

Function: Reusable block of code that performs a specific task. Functions help in organizing code, making it modular and easy to maintain.

Exercise: Write a function that inputs a string and count the number of vowels in a given word.

```
def count_vowels(word)
    #word = input("Enter a word: ")
    vowels = "AEIOUaeiou"
    count = 0
    for letter in word:
        if letter in vowels:
            count += 1
    print("Number of vowels:", count)
```

Note: Notice how **`in`** is used with **`for`** and **`if`**. With **`for,`** it is used for accessing elements in a sequence one at a time. With **`if`**, it checks the membership.

Exercise: Write a function to check whether a given number is prime or not

```
def is_prime(n):
    #Code here
```

Note: use the above function to print all prime numbers till 100.

Built-in Functions: Functions that are already available in Python
Some Examples (Explore these on your own):

- input()
- eval()
- print()
- type()
- type conversion: int(), float(), str() …
- divmod()
- min(), max(),
- pow()
- round()

- Random number Generation (random module)

```
import random
print(random.random())  #[0, 1)
print(random.randint(2, 10) #[2, 10]
```

Exercise: Use the above random number generator to generate random numbers between [a, b) or integers [a, b]

- math module

```
import math
math.sqrt(24)
```

Other: ceil, floor, fabs, exp, log, log10, pow, sqrt, cos, sin, tan, degrees, radians etc

- `math.pi #constant pi in math module`

- `dir(__builtins__)` : to check complete list of built in functions

- `help`: to see documentation. E.g.

```
import math
help(math.sqrt)
```

- `import this` (zen of python is printed first time the module is imported)

Exercises (Complete the following functions):

1.

```
def factorial(n):
    # return factorial of n
print(factorial(5))  # Expected Output: 120
```

2.

```
def even_or_odd(n):
    # print even or odd
```

3.

```
def maximum(a, b, c):
    # return maximum of 3 numbers
print(maximum(3, 7, 5))  # Expected Output: 7
```

4.

```
def fibonacci(n):
    #print fibonacci upto n terms
print(fibonacci(6)) #Expected Output: 0, 1, 1,
2, 3, 5
```

5.

```
def gcd(a, b):
    #return gcd/hcf of two numbers
print(gcd(36, 60))  # Expected Output: 12
```

6. Write programs to find the sum of series below using functions and loops:

   a. $1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots \frac{x^n}{n!}$

   b. $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$

   c. $sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} \ldots$

7. Write a function that returns the sum of digits of a number, passed to it as an argument.

8. Write a function that takes two numbers as input parameters and returns their least common multiple.

9. Write a function to determine whether a given natural number is a perfect number. A number is perfect if its is the sum of its divisors. E.g. 6 = 1+ 2+3

10. A robot is placed at position (0,0) on a 2-dimensional grid. The robot can move **up, down, left, or right** based on user input. The goal is to allow the user to

control the robot's movement using commands and display the final position after a series of moves. Write a program to implement the above with following rules using while/if-else.

Rules:

a. The robot starts at (0,0).
b. The user inputs a series of moves:
    - "U" (Up) → Increases the y-coordinate by 1
    - "D" (Down) → Decreases the y-coordinate by 1
    - "L" (Left) → Decreases the x-coordinate by 1
    - "R" (Right) → Increases the x-coordinate by 1
c. The program should continue taking inputs until the user enters "STOP".
d. After the loop ends, display the final position of the robot.
e. (Optional) Update the above program to include the following condition.
    - Any move that makes the x or y-coordinate (-) negative is invalid and results in no move.
f. (New) Add a display_grid function to show the position of the robot on the grid. Use dots(.) and (R) to indicate positions in grid and robots' exact position in grid.