```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "9818ffc8",
   "metadata": {},
   "source": [
    "# Python Practice Problems"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "92c96f2f",
   "metadata": {},
   "source": [
    "### 1. Find the Largest of Two Numbers\n",
    "**Task:** Given two numbers, find the larger one.  \n",
    "**Demo Input:**  5 and 6 <br>\n",
    "**Output:** 6\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "15c848cf",
   "metadata": {},
   "outputs": [],
   "source": [
    "a = int(input(\"6\"))\n",
    "b = int(input(\"5\"))\n",
    "if a > b:\n",
    "    print(\"The larger number is:\", a)\n",
    "else:\n",
    "    print(\"The larger number is:\", b)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "c8c1c683",
   "metadata": {},
   "source": [
    "### 2. Print Numbers from 1 to N\n",
    "**Task**: Given a number `n`, print all numbers from 1 to `n`. <br>
\n",
    "**Demo Input**: `5`  <br>\n",
    "**Demo Output**:  1 2 3 4 5\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "id": "adbb8035-9ccd-4a87-b319-5a77166c3e8f",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
```

```
      "1\n",
      "2\n",
      "3\n",
      "4\n",
      "5\n"
     ]
    }
   ],
   "source": [
    "n = 5\n",
    "for i in range(1, n+1):\n",
    "    print(i)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "899df468",
   "metadata": {},
   "source": [
    "### 3. Check if a Number is Positive or Negative\n",
    "**Task**: Determine if the input number is positive, negative, or zero.  \n",
    "**Demo Input**: `-3`  \n",
    "**Demo Output**: `The number is Negative`"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "id": "56ae285e",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "The number is Negative\n"
     ]
    }
   ],
   "source": [
    "n = -3\n",
    "if n < 0:\n",
    "    print( \"The number is Negative\")\n",
    "    \n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "9d865d12",
   "metadata": {},
   "source": [
    "### 4. Calculate the Sum of Digits\n",
    "**Task**: Given a number, find the sum of its digits.  \n",
    "**Demo Input**: `123`  \n",
    "**Demo Output**: `6`  \n",
    "(Explanation: `1+2+3=6`)"
   ]
```

```
},
{
 "cell_type": "code",
 "execution_count": 1,
 "id": "c28b2c6c",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "6\n"
   ]
  }
 ],
 "source": [
  "def sum_of_digits(n):\n",
  "    sum = 0\n",
  "    while(n != 0):\n",
  "        sum = sum + (n % 10)\n",
  "        n = n//10\n",
  "    return sum\n",
  "\n",
  "number = 123\n",
  "result = sum_of_digits(number)\n",
  "print(result)"
 ]
},
{
 "cell_type": "markdown",
 "id": "a970046f",
 "metadata": {},
 "source": [
  "### 5. Find Factorial of a Number\n",
  "**Task**: Calculate the factorial of a given number.  \n",
  "**Demo Input**: `4`  \n",
  "**Demo Output**: `24`  \n",
  "(Explanation: `4! = 4*3*2*1 = 24`)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 17,
 "id": "fd5966d2",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "The factorial of 4 is 24\n"
   ]
  }
 ],
 "source": [
  "def factorial(n):\n",
  "  \"\"\"Calculates the factorial of a non-negative integer.\n",
  "\n",
```

```
    "  Args:\n",
    "    n: The non-negative integer for which to calculate the
factorial.\n",
    "\n",
    "  Returns:\n",
    "    The factorial of n.\n",
    "  \"\"\"\n",
    "  if n == 0:\n",
    "    return 1  # Base case: factorial of 0 is 1\n",
    "  else:\n",
    "    result = 1\n",
    "    for i in range(1, n + 1):\n",
    "      result *= i\n",
    "    return result\n",
    "\n",
    "# Get input from the user\n",
    "number = 4\n",
    "# Calculate the factorial\n",
    "result = factorial(number)\n",
    "\n",
    "# Print the result\n",
    "print(f\"The factorial of {number} is {result}\")"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "c8798099",
   "metadata": {},
   "source": [
    "### 6. Count Occurrences of a Digit\n",
    "**Task**: Count how many times a specific digit appears max in a
number.  \n",
    "**Demo Input**:  1233321 <br>\n",
    "**Demo Output**: `3`  <br>\n",
    "(Explanation: `3` appears 3 times in `1233321`)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 18,
   "id": "5772f0e6",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "3\n"
     ]
    }
   ],
   "source": [
    "number = \"1233321\"\n",
    "digit_to_count = '3'\n",
    "count = number.count(digit_to_count)\n",
    "print(count)"
   ]
  },
```

```
  {
   "cell_type": "markdown",
   "id": "b7f985bc",
   "metadata": {},
   "source": [
    "### 7. Find the GCD of Two Numbers\n",
    "**Task**: Find the greatest common divisor (GCD) of two numbers.
\n",
    "**Demo Input**: 8 and 12 <br>\n",
    "**Demo Output**: `4`"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 20,
   "id": "a467887d",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "The GCD of 8 and 12 is: 4\n"
     ]
    }
   ],
   "source": [
    "import math\n",
    "int1 = 8\n",
    "int2 = 12\n",
    "gcd_value = math.gcd(8, 12)\n",
    "print(f\"The GCD of {8} and {12} is: {gcd_value}\")"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "14c6a2fd",
   "metadata": {},
   "source": [
    "### 8. Reverse a String\n",
    "**Task**: Reverse the given string.  \n",
    "**Demo Input**: `\"aiquest\"`  \n",
    "**Demo Output**: `\"tseuqia\"`"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 22,
   "id": "bd22e158",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "tseuqia\n"
     ]
    }
```

```
   ],
   "source": [
    "input_string = \"aiquest\"\n",
    "reversed_string = input_string[::-1]\n",
    "\n",
    "print(reversed_string)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "88317dfd",
   "metadata": {},
   "source": [
    "### 9. Check Armstrong Number\n",
    "**Task**: Check if a number is an Armstrong number (the sum of its digits raised to the power of the number of digits equals the number). \n",
    "**Demo Input**: `153`  \n",
    "**Demo Output**: `153 is an Armstrong Number`  \n",
    "(Explanation: \\(1^3 + 5^3 + 3^3 = 153\\))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 30,
   "id": "2602ff94",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "True\n"
     ]
    }
   ],
   "source": [
    "def is_armstrong(num):\n",
    "    num_str = str(num)\n",
    "    n = len(num_str)\n",
    "    sum = 0\n",
    "    for digit in num_str:\n",
    "        sum += int(digit)**n\n",
    "    if sum == num:\n",
    "        return True\n",
    "    else:\n",
    "        return False\n",
    "num=153\n",
    "print(is_armstrong(num))"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "0905a3cc",
   "metadata": {},
   "source": [
    "### 10. Generate a Pattern\n",
    "**Task**: Print a pyramid pattern with `n` rows.  \n",
```

```
    "**Demo input:** 4 <br>\n",
    "**Output:** You will see the pyramid pattern with 4 rows"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 31,
   "id": "83d8b483",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "   *\n",
      "  ***\n",
      " *****\n",
      "*******\n"
     ]
    }
   ],
   "source": [
    "# Function to print full pyramid pattern\n",
    "def full_pyramid(n):\n",
    "    for i in range(1, n + 1):\n",
    "        # Print leading spaces\n",
    "        for j in range(n - i):\n",
    "            print(\" \", end=\"\")\n",
    "        \n",
    "        # Print asterisks for the current row\n",
    "        for k in range(1, 2*i):\n",
    "            print(\"*\", end=\"\")\n",
    "        print()\n",
    "   \n",
    "full_pyramid(4)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "ec723917",
   "metadata": {},
   "source": [
    "# Good Luck!"
   ]
  }
 ],
"metadata": {
 "kernelspec": {
  "display_name": "Python 3 (ipykernel)",
  "language": "python",
  "name": "python3"
 },
 "language_info": {
  "codemirror_mode": {
   "name": "ipython",
   "version": 3
  },
  "file_extension": ".py",
```

```
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.11.0"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```