Assignment 2

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "##### Write a Calculator program in Python?"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {},
   "outputs": [],
   "source": [
    "class Calculator:\n",
    "    def add(self, a, b):\n",
    "        returna+b\n",
    "    def sub(self, a, b):\n",
    "        return a-b\n",
    "    defmult(self, a, b):\n",
    "        return a*b\n",
    "    def div(self, a, b):\n",
    "        return a/b"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "6\n"
     ]
    }
   ],
   "source": [
    "c = Calculator()\n",
    "print(c.add(2, 4))"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "##### Write a program to concatenate, reverse and slice a string?"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "metadata": {},
   "outputs": [],
   "source": [
    "class String:\n",
    "    defconcat(self, a, b):\n",
    "        returna+b\n",
    "    def reverse(self, s):\n",
  "        return s[::-1]\n",
  "    defslicestr(self, s, start, end):\n",
  "        return s[start:end+1]"
 ]
},
{
 "cell_type": "code",
 "execution_count": 6,
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "amku\n"
   ]
  }
 ],
 "source": [
  "s = String()\n",
  "print(s.slicestr('ramkumar', 1, 4))"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "#####  Why is Python a popular programming language?"
 ]
},
{
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "It uses a simplified syntax with an emphasis on natural language, for a much easier learning curve for beginners. And, because Python is free to use and is supported by an extremely large ecosystem of libraries and packages, it's often the first-choice language for new developers."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "##### What are the other Frameworks that can be used with python?"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Pyramid, TurboGears, Web2py, CherryPy, Flask, Sanic"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "##### Full form of WSGI"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "The Web Server Gateway Interface is a simple calling convention for web servers to forward requests to web applications or frameworks written in the Python programming language."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "##### Consider a list (list = []). You can perform the operations"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 15,
   "metadata": {},
   "outputs": [],
   "source": [
    "class List:\n",
    "    def __init__(self):\n",
    "        self.l = []\n",
    "    def insert(self, a, pos):\n",
    "        ifpos<= len(self.l):\n",
    "            self.l.insert(pos, a)\n",
    "        else:\n",
    "            print('position out of range')\n",
    "            return\n",
    "        returnself.l\n",
    "    def remove(self, a):\n",
    "        self.l.remove(a)\n",
    "        returnself.l\n",
    "    def append(self, a):\n",
    "        self.l.append(a)\n",
    "        returnself.l\n",
    "    def sort(self):\n",
    "        returnself.l\n",
    "    def pop(self):\n",
    "        returnself.l.pop()\n",
    "    def reverse(self):\n",
    "        returnself.l[::-1]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 25,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[11]\n",
      "[11, 12]\n",
      "[12]\n",
      "[12, 13]\n",
      "[12, 13, 15]\n",
      "[12, 13, 15]\n",
      "15\n",
      "[13, 12]\n"
     ]
    }
   ],
   "source": [
    "li = List()\n",
    "print(li.insert(11, 0))\n",
    "print(li.insert(12, 1))\n",
    "print(li.remove(11))\n",
    "print(li.append(13))\n",
    "print(li.append(15))\n",
    "print(li.sort())\n",
    "print(li.pop())\n",
    "print(li.reverse())"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "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": "3.8.5"
 }
},
"nbformat": 4,
"nbformat_minor": 4
}
```