PYTHON

Module 1: Advanced Program Of Choice

(Sudoku Puzzle Generator & Solver)

800894411 Madhu Ramachandra

800916727 Lucy Lopamudra ULN

800900744 Udayasri Buddhi

1. Objective

We are implementing a program to generate Sudoku puzzle and also provide the solution to it.

- Input : User will be provided a 9*9 Sudoku puzzle to solve
- Output: User will try different combinations to solve the puzzle
- If the User is entering multiple values of a particular number in a single row or column, the output of that cell will be set to '0' in those cells.
- User can also select the solution button to see the solution of the puzzle attempted.

2. Reasons for Chosen program

- Easy generation of the puzzle randomly
- Implementing the gaming and UI features in Python
- Easy Implementation of logic in Python

3. Major Data Structures Used

3.1 Python Dictionary

- Dictionary is an unordered collection of key-value pairs.
- It is used when we have huge data.
- We have to know the key to retrieve the value.
- Dictionaries are defined within {} and each item in it is a pair of key:value. Key and value can have any type.

```
>>> d = {60:'value','key':55}
>>> type(d)
<class 'dict'>
```

3.2 Python List

- List is an ordered sequence of items.
- All items of the list do not need to be of same type.
- Items in a list are separated by comma and the list is enclosed in [].
- Index in a list starts from 0.

```
>>> b = [1, 0, 'uncc']
>>> type(b)
<class 'list'>
```

3.3 Array

- Array is an ordered collection of items of same type.

3.4 Python Strings

- String is a sequence of characters.
- It can be represented by single quotes or double quotes
- Multiline strings can be represented by triple quotes.

```
>>> s = "Hi I am a string"
>>> type(s)
<class 'str'>
>>> s = '''Hi I am a multiline
```

```
... string'''
>>> s = 'Hello world!'
>>> s[4]
'o'
```

3.5 Python Tuple

- Set is an unordered collection of unique items.
- Set values are separated by comma inside braces { }.
- We can do Union and Intersection operation on it.
- Individual items cannot be accessed by [] as it is unordered.

4. Abstractions Used

4.1 Functions

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

Defining a function: Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.

Syntax:

```
def func_name():
        # body line 1
        #body line 2
        #body line 3
```

Calling a function:

```
Func_name()
```

4.2 Classes and objects

There is a construct in Python called a class that lets you structure your software in a particular way. Using classes, you can add consistence to your programs so that they can be used in a cleaner way. Python classes provide all the standard features of Object Oriented Programming: the class inheritance mechanism allows multiple base classes, a derived class can override any methods of its base class or classes, and a method can call the method of a base class with the same name. Objects can contain arbitrary amounts and kinds of data. As is true for modules, classes partake of the dynamic nature of Python: they are created at runtime, and can be modified further after creation.

Syntax:

```
class MyClass:
        <statement-1>
        <statement-2>
        <statement-N>
```

Class *instantiation* uses function notation (object creation). Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Syntax:

```
X=MyClass()
```

To access the variable inside of the newly created object, use
x.variable

Of course, the **__init__()** method may have arguments for greater flexibility. In that case, arguments given to the class instantiation operator are passed on to **__init__()**.
class Shape:

```
def _init__(self,x,y):
        self.x = x self.y
        = y
def area(self):
        return self.x * self.y def
perimeter(self):
        return 2 * self.x + 2 * self.y
```

4.3 Control abstractions

4.3.1 Conditional execution

4.3.1.1 The if statement

The if statement evaluates a condition. If it evaluates to true, the corresponding statements are executed.

In order to write useful programs, we almost always need the ability to check conditions and change the behaviour of the program accordingly. Conditional statements give us this ability. The simplest form is the **if** statement.

Syntax:

```
if True:

print "Yes"
```

4.3.1.2. The if else statement

When the if statement evaluates an expression to false, a different set of statements can be executed.

This is specified by the if/else construct.

Syntax:

```
if True:

print "Yes"

else:

print "No"
```

4.3.1.3 Nested Conditionals

The if else statements can be nested within each other to provide more conditions.
Syntax:

Example:

If 0 < x and x < 10:

Print “x is a positive single digit”

Python actually allows a short hand form for this:

If 0 < x < 10:

Print “x is a positive single digit”

4.3 .1.4 The for loop

The for loop iterates over the specified statements, a specific number of times. It processes each item in a sequence & can be used with Python’s sequence data types - strings, lists, and tuples.

Each item in turn is (re-)assigned to the loop variable, and the body of the loop is executed.

Example: To remove spaces from a sentence

```
sentence = raw_input("Please enter a sentence: ")

no_spaces = ''

for letter in sentence:

    if letter != ' ':

        no_spaces += letter

print("You sentence with spaces removed:")

print(no_spaces)
```

Output:

```
>>>

Please enter a sentence: hello! how are you? You sentence with

spaces removed: hello!howareyou?

>>>
```

4.3.1.4 The while statement

The while statement evaluates a condition first. If it is true, it executes the sequence of statements. At the end of every iteration, it evaluates the condition again & executes the sequence of statements, if the condition is true. This loops until the condition evaluates to false.

5. Major Types Used

5.1 Python Numbers

- It includes Integers, floating point numbers and complex numbers.
- They are defined as int, float and complex class.

- Integers can be of any length and is limited by the memory available. A floating-point number is accurate up to 15 decimal places. Integer and floating points are distinguished by decimal points. 2 is integer, 2.0 is floating point number.
- Complex numbers are written in the form, x + yj, where *x* is the real part and *y* is the imaginary part.

```
>>> a = 5
>>> type(a)
<class 'int'>
>>> type(2.0)
<class 'float'>
>>> c = 1+2j
>>> type(c)
<class 'complex'>
```

5.2 Python Strings

- String is a sequence of characters.
- It can be represented by single quotes or double quotes
- Multiline strings can be represented by triple quotes.

```
>>> s = "Hi I am a string"
>>> type(s)
<class 'str'>
>>> s = '"Hi I am a multiline
... string"'
>>> s = 'Hello world!'
>>> s[4]
‘o
```

5.3 Python Tuple

- Set is an unordered collection of unique items.
- Set values are separated by comma inside braces { }.
- We can do Union and Intersection operation on it.
- Individual items cannot be accessed by [] as it is unordered

6. Unique Syntax Used

6.1 Random Puzzle generation by using random function

- It generates random combinations from a given list .

```
disp_su = random.sample(currentGrid.keys(),20)
```

- This helped us in generating the puzzle randomly

6.2 List Comprehensions

- temp_grid = {k: currentGrid[k] if k in currentGrid else default for k in disp_su}

HELP GUIDE

How to install python

1. Download IDLE (Python) 3.5.0 from here https://www.python.org/downloads/

IDLE is the standard **Python** development environment. Its name is an acronym of "Integrated Development Environment". It works well on both Unix and Windows platforms. It has a **Python** shell window, which gives you access to the **Python** interactive mode.

2. Run the exe file
3. Choose Install Now
4. Open IDLE(Python GUI)
   Python shell will open

How to install pygame

1. Download appropriate version of pygame from here
http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame

2. Copy the .whl file to the python folder
3. in the command prompt, run:

```
pip install wheel

pip install pygame-1.9.2a0-cp34-none-win_amd64.whl
```

How to run the script

1. Choose File → New Window
2. Run the script Sudoku.py