100 DAYS OF PYTHON CODE BY ANGELA YU

DAY-1

FOR DOUTS HEAD OVER TO: https://www.python.org/ AND https://www.askpython.com/

PRINTING PYTHON STRINGS

In [1]:
```
#Printing the string
print("Hello World")
```

```
Hello World
```

STRING MANIPULATION AND CODE INTELLIGENCE

In [2]:
```
#New line helps to make the word after to next line
print("Hello World \nHello World")
```

```
Hello World
Hello World
```

CONCATENATION

-->COMBAINING TWO STRINGS

In [3]:
```
print("Hello"+" Srenath")
```

```
Hello Srenath
```

GETTING INPUT FOR THE CODE:

In [5]:
```
a=input("What is your name")
a
```

Out[5]:
```
'Srenath kumar'
```

In [7]:
```
print("Hello "+input("What is your name"))
```

```
Hello Srenath kumar
```

Len() Function

-->Python Built In function to count no of characters in the variable

In [8]:
```
print(len("Hello"))
```

```
5
```

Variable

-->Storing the data's

Day-1:Project Band Name Generator

In [5]:
```
a="Hello"
city=input("Enter your city:")
pet=input("Enter your pet:")
print(a+" "+city+" "+pet)
```

```
Hello Chennai Rabbit
```

DAY-2

PYTHON PRIMITIVE DATA TYPES

-->Pulling out some of the elements from the words are known as subscript

In [4]:
```
print("Hello"[0])
```

```
H
```

-->Here,it is called as subscriptInt str bool float

Type Conversion

-->Process of converting one data type to another

In [6]:
```
print(type(str(123))) #type conversion      #conversion of int to str
```

```
<class 'str'>
```

Mathematical Operations

+ Addition - Subtraction / Division * Multiplication ** Exponentiation operator // Floor Division

PEMDAS

P ->PARANTHESES E ->EXPONENTS M ->MULTIPLICATION D ->DIVISION A ->ADDITION S ->SUBTRACTION

NUMBER MANIPULATION

ROUND() FUNCTION

-->Function to round the decimal value

In [1]:
```
print(round(8/3))
```

```
3
```

In [2]:
```
print(round(8/3,2))
```

```
2.67
```

-->Rounding two decimal places

Day-2:Project-Tip calculator

In [3]:
```
print("Welcome to the tip calculator!")
bill = float(input("What was the total bill? $"))
tip = int(input("How much tip would you like to give? 10, 12, or 15? "))
people = int(input("How many people to split the bill?"))

tip_as_percent = tip / 100
total_tip_amount = bill * tip_as_percent
total_bill = bill + total_tip_amount
bill_per_person = total_bill / people
final_amount = round(bill_per_person, 2)

print(f"Each person should pay: ${final_amount}")
```

```
Welcome to the tip calculator!

Each person should pay: $22.0
```

DAY-3

IF/ELSE CONDITION

SYNTAX: if condition: do this else: do this

In [7]:
```
a=3
if a<4:
    print("The number is lesser")
else:
    print("The number is greater")
```

```
The number is lesser
```

ELIF CONDITION

It is done if there is more than two conditionSYNTAX: if condition: do this elif condition: do this else: do this

In [9]:
```
b=43
if b==50:
    print("The number is 50")
elif b==43:
    print("The number is 43")
else:
    print("Invalid")
```

```
The number is 43
```

NESTED IF/ELSE

SYNTAX: if condition: if condition: do this else: do this else: do this

In [10]:
```
num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

```
Positive number
```

DAY-3:PROJECT-LOVE CALCULATOR

In [1]:
```
print("Welcome to the Love Calculator!")
name1 = input("What is your name? \n")
name2 = input("What is their name? \n")

combined_names = name1 + name2
lower_names = combined_names.lower()
t = lower_names.count("t")
r = lower_names.count("r")
u = lower_names.count("u")
e = lower_names.count("e")
first_digit = t + r + u + e

l = lower_names.count("l")
o = lower_names.count("o")
v = lower_names.count("v")
e = lower_names.count("e")
second_digit = l + o + v + e

score = int(str(first_digit) + str(second_digit))

if (score < 10) or (score > 90):
    print(f"Your score is {score}, you go together like coke and mentos.")
elif (score >= 40) and (score <= 50):
    print(f"Your score is {score}, you are alright together.")
else:
    print(f"Your score is {score}.")
```

```
Welcome to the Love Calculator!

Your score is 73.
```

DAY-4

Random functions()

-->Python use Mersenne Twister algorithm for the random number generator-->module for random numbers is random

In [3]:
```
import random
```

In [7]:
```
#randint is used for producing random integer numbers of the given range
random_integer=random.randint(1,10)
random_integer
```

Out[7]: 2

File:my_module

In [8]:
```
pi=3.1432323
```

File:2

import my_module # Importing from the file my_module print(my_module.pi) #printing the pi from the file my_module which is taken as module Output:31432323

In [13]:
```
#this random() function is used to generate random numbers between 0 and 1
random_float=random.random()
random_float
```

Out[13]: 0.8226574896266053

In [21]:
```
#Generating floating random numbers between 1 to 5
#Here we multiply the value with 5 to get the value more than 1
#Tricks to get the value
random_float=random.random()
random_float*5
```

Out[21]: 2.5220979184962156

Random Exercise

-->You are going to write a virtual coin toss program -->It will randomly tell the user "Heads" or "Tails".

In [25]:
```
import random

heads_or_tails=random.randint(0,1)
if heads_or_tails == 1:
    print("Heads")
else:
    print("Tails")
```

```
Tails
```

Lists

-->Lists are used to store multiple items in a single variable. -->Lists are one of 4 built-in data types in Python used to store collections of data -->Lists are created using square brackets:[]

In [38]:
```
thislist1=["apple","banana","cherry"]
thislist1
```

Out[38]: ['apple', 'banana', 'cherry']

Lists Indexing

In [39]:
```
#This gets printed according to the given index
#Remember that always indexes starts from 0
thislist1[1]
```

Out[39]: 'banana'

In [40]:
```
#When we give the value negative it takes form the reverse
thislist1[-1]
```

Out[40]: 'cherry'

In [41]:
```
#Also the lists values can be changes
#The value from banana changes to orange
thislist1[1]="orange"
thislist1
```

Out[41]: ['apple', 'orange', 'cherry']

In [42]:
```
#To add data in the list we use the function called append()
#which is a inbuilt function
thislist1.append("mango")
thislist1
```

Out[42]: ['apple', 'orange', 'cherry', 'mango']

In [43]:
```
# We use extend function in the list to add more than one datas
thislist1.extend(["pumpkin","guava","pomagranete"])
```

In [44]:
```
thislist1
```

Out[44]: ['apple', 'orange', 'cherry', 'mango', 'pumpkin', 'guava', 'pomagranete']

Split Function

-->This function is used to split the words according to the given keyword

In [1]:
```
r="Hello,World,Srenath"
lists3=r.split(",")
lists3
```

Out[1]: ['Hello', 'World', 'Srenath']

CODING-EXERCISE

BANKER-ROUETTE-WHO WILL PAY THE BILL

In [1]:
```
# NAMES are inputted in the names which gets splited by comma
import random

# Split string method
names_string = input("Give me everybody's names, seperated by a comma. ")
names = names_string.split(", ")

#Get the total number of items in list.
num_items = len(names)
#Generate random numbers between 0 and the last index.
random_choice = random.randint(0, num_items - 1)
#Pick out random person from list of names using the random number.
person_who_will_pay = names[random_choice]
print(person_who_will_pay + " is going to buy the meal today!")
```

```
SREJA  is going to buy the meal today!
```

DUAL LISTS

In [2]:
```
# Creating dual lists by combining two seperate lists
# It is also called as nested Lists
fruits=["orange","mango","apple","banana","guava"]
vegetables=["carrot","beans","onions","betroot","tomatoes"]
fruits_vegetables=[fruits,vegetables]
```

In [3]:
```
fruits_vegetables
```

Out[3]:
```
[['orange', 'mango', 'apple', 'banana', 'guava'],
 ['carrot', 'beans', 'onions', 'betroot', 'tomatoes']]
```

CODING EXERCISE-TREASURE ISLAND MAP

-->You are going to write a program which will mark a spot with an X. -->In the starting code, you will find a variable called map.

In [4]:
```
row1 = ["⬜","⬜","⬜"]
row2 = ["⬜","⬜","⬜"]
row3 = ["⬜","⬜","⬜"]
map = [row1, row2, row3]
print(f"{row1}\n{row2}\n{row3}")

position = input("Where do you want to put the treasure? ")

horizontal = int(position[0])
vertical = int(position[1])

map[vertical - 1][horizontal - 1] = "X"

print(f"{row1}\n{row2}\n{row3}")
```

```
['⬜', '⬜', '⬜']
['⬜', '⬜', '⬜']
['⬜', '⬜', '⬜']

['⬜', '⬜', 'X']
['⬜', '⬜', '⬜']
['⬜', '⬜', '⬜']
```

DAY-4-PROJECT-CHALLANGE-ROCK,PAPER,SCISSORS

In [13]:
```
rock = '''
    _______
---'   ____)
      (_____)
      (_____)
      (____)
---.__(___)
'''

paper = '''
    _______
---'   ____)____
          ______)
          _______)
         _______)
```

```
---._________)
'''

scissors = '''
    _______
---'   ____)____
          ______)
       __________)
      (____)
---.__(___)
'''
import random
person_choose=int(input("Select 0 for rock,1 for paper,2 for scissors:"))
computer_choose=random.randint(0,2)
print("You_choose:")
if person_choose == 0:
    print(rock)
elif person_choose ==1:
    print(paper)
elif person_choose ==2:
    print(scissors)
else:
    print("Invalid Input")

print("Computer_choose:")
if computer_choose == 0:
    print(rock)
elif computer_choose ==1:
    print(paper)
elif computer_choose ==2:
    print(scissors)
else:
    print("Invalid Input")

if person_choose==0 and computer_choose==0:
    print("Tie")
if person_choose==0 and computer_choose==1:
    print("You win the game")
if person_choose==0 and computer_choose==2:
    print("You win the game")

if person_choose==1 and computer_choose==0:
    print("You lose the game")
if person_choose==1 and computer_choose==1:
    print("Tie")
if person_choose==1 and computer_choose==2:
    print("You lose the game")

if person_choose==2 and computer_choose==0:
    print("You lose the game")
if person_choose==2 and computer_choose==1:
    print("You win the game")
if person_choose==2 and computer_choose==2:
    print("Tie")
```

```
You_choose:

    _______
---'   ____)____
          ______)
```

```
      __________)
     (____)
---.__(___)
```

```
Computer_choose:
```

```
    _______
---'   ____)____
          ______)
          _______)
         _______)
---.__________)
```

```
You win the game
```

Method-2

In [2]:

```
import random

rock = '''
    _______
---'   ____)
      (_____)
      (_____)
      (____)
---.__(___)
'''

paper = '''
    _______
---'   ____)____
          ______)
          _______)
         _______)
---.__________)
'''

scissors = '''
    _______
---'   ____)____
          ______)
       __________)
      (____)
---.__(___)
'''
game_images = [rock, paper, scissors]

user_choice = int(input("What do you choose? Type 0 for Rock, 1 for Paper or 2 for Scis
print(game_images[user_choice])

computer_choice = random.randint(0, 2)
print("Computer chose:")
print(game_images[computer_choice])

if user_choice >= 3 or user_choice < 0:
    print("You typed an invalid number, you lose!")
elif user_choice == 0 and computer_choice == 2:
    print("You win!")
elif computer_choice == 0 and user_choice == 2:
    print("You lose")
elif computer_choice > user_choice:
```

```
    print("You lose")
elif user_choice > computer_choice:
    print("You win!")
elif computer_choice == user_choice:
    print("It's a draw")

####### Debugging challenge: #########
#Try running this code and type 5.
#It will give you an IndexError and point to line 32 as the issue.
#But on line 38 we are trying to prevent a crash by detecting
#any numbers great than or equal to 3 or less than 0.
#So what's going on?
```

```
    _______
---'   ____)
      (_____)
      (_____)
      (____)
---.__(___)

Computer chose:

    _______
---'   ____)____
          ______)
       __________)
      (____)
---.__(___)

You win!
```

DAY-5

For Loops

In [6]:
```
# Assigining the fruits in the list
fruits=["apple","orange","grapes"]
# For loop used to print the datas in every iterations
for i in fruits:
    print(i)
```

```
apple
orange
grapes
```

In [7]:
```
fruits=["apple","orange","grapes"]
for i in fruits:
    print(i)
#on adding the string in the loop it prints with the strings
    print(i+" pies")
```

```
apple
apple pies
orange
orange pies
grapes
grapes pies
```

CODING EXERCISE:AVERAGE HEIGHT

In [11]:
```
student_heights = input("Input a list of student heights ").split()
for n in range(0, len(student_heights)):
    student_heights[n] = int(student_heights[n])

    total_height = 0
for height in student_heights:
    total_height += height
print(f"total height = {total_height}")

number_of_students = 0
for student in student_heights:
    number_of_students += 1
print(f"number of students = {number_of_students}")

average_height = round(total_height / number_of_students)
print(average_height)
```

```
total height = 1167
number of students = 3
389
```

CODING CHALANGE-CALCULATING HIGHEST SCORE

In [12]:
```
students_score=[32,543,32,54,65,43]
hightst_scores=0
for scores in students_score:
    if scores>hightst_scores:
        hightst_scores=scores
print(hightst_scores)
```

```
543
```

FOR LOOPS WITH RANGE FUNCTION

In [15]:
```
#Printing the number with the range between 1 to 10
#It prints with n-1 i,e it does not print last number
for i in range(1,11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

To add sum of numbers between 1 to 100

In [17]:
```
total=0
for i in range(1,101):
    total += i
print(total)
```

```
5050
```

CODING EXERCISE:TO ADD UP SUM OF ALL EVEN NUMBERS WITH FOR LOOPS

In [21]:
```
total=0
#RANGE=(START,STOP,STEPOVER) INDEXES
for i in range(2,101,2):
    total +=i
print(total)
```

```
2550
```

In [23]:
```
total=0
for i in range(1,101):
    if i % 2==0:
        total += i
print(total)
```

```
2550
```

CODING EXERCISE:FIZZ-BUZZ GAME

In [27]:
```
for i in range(1,101):
    if i %5==0 and i%3==0:
        print("FizzBuzz")
    elif i % 5 == 0:
        print("Buzz")
    elif i%3==0:
        print("Fizz")
    else:
        print(i)
```

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
Fizz
22
23
Fizz
Buzz
26
Fizz
28
```

```
29
FizzBuzz
31
32
Fizz
34
Buzz
Fizz
37
38
Fizz
Buzz
41
Fizz
43
44
FizzBuzz
46
47
Fizz
49
Buzz
Fizz
52
53
Fizz
Buzz
56
Fizz
58
59
FizzBuzz
61
62
Fizz
64
Buzz
Fizz
67
68
Fizz
Buzz
71
Fizz
73
74
FizzBuzz
76
77
Fizz
79
Buzz
Fizz
82
83
Fizz
Buzz
86
Fizz
88
89
FizzBuzz
91
92
Fizz
```

```
94
Buzz
Fizz
97
98
Fizz
Buzz
```

DAY-5 PROJECT-RANDOM_PASWORD_GENERATOR

In [1]:

```
#Password Generator Project
import random
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
           'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
           'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D',
           'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
           'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
           'Y', 'Z']
numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
symbols = ['!', '#', '$', '%', '&', '(', ')', '*', '+']

print("Welcome to the PyPassword Generator!")
nr_letters = int(input("How many letters would you like in your password?\n"))
nr_symbols = int(input(f"How many symbols would you like?\n"))
nr_numbers = int(input(f"How many numbers would you like?\n"))


password_list = []

for char in range(1, nr_letters + 1):
    password_list.append(random.choice(letters))

for char in range(1, nr_symbols + 1):
    password_list += random.choice(symbols)

for char in range(1, nr_numbers + 1):
    password_list += random.choice(numbers)

print(password_list)
random.shuffle(password_list)
print(password_list)

password = ""
for char in password_list:
    password += char

print(f"Your password is: {password}")
```

```
Welcome to the PyPassword Generator!

['s', 's', 'X', 'q', 'b', ')', '(', '*', '7', '2', '2', '6', '7']
['6', 's', 'q', '2', '2', '7', '*', ')', 'X', 'b', 's', '(', '7']
Your password is: 6sq227*)Xbs(7
```

In [2]:

```
import random
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
           'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', '
           'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', '
numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
symbols = ['!', '#', '$', '%', '&', '(', ')', '*', '+']
```

```
print("Welcome to the PyPassword Generator!")
nr_letters = int(input("How many letters would you like in your password?\n"))
nr_symbols = int(input(f"How many symbols would you like?\n"))
nr_numbers = int(input(f"How many numbers would you like?\n"))


password_list = []

for char in range(1, nr_letters + 1):
    password_list.append(random.choice(letters))

for char in range(1, nr_symbols + 1):
    password_list += random.choice(symbols)

for char in range(1, nr_numbers + 1):
    password_list += random.choice(numbers)

print(password_list)
random.shuffle(password_list)
print(password_list)

password = ""
for char in password_list:
    password += char

print(f"Your password is: {password}")

password = ""

for char in range(1, nr_letters + 1):
    password += random.choice(letters)

for char in range(1, nr_symbols + 1):
    password += random.choice(symbols)

for char in range(1, nr_numbers + 1):
    password += random.choice(numbers)

print(password)
```

```
Welcome to the PyPassword Generator!

['z', 'V', 'f', 'i', 'd', '(', '%', '&', '+', ')', '2', '2', '7', '5', '9', '5']
['z', '9', '2', '2', '5', 'd', 'f', ')', '5', '%', '(', '7', '&', '+', 'V', 'i']
Your password is: z9225df)5%(7&+Vi
OPxzq$&&&(865730
```

DAY-6

USER-DEFINED FUNCTIONS

-->SYNTAX: def own_function_name(): do this

In [2]:

```
#User defined Functions
def ones():
    print("Ones")
#Calling the functions
ones()
```

```
Ones
```

WHILE LOOP

SYNTAX: while condition_true: do this

In [5]:
```
i = 1
while i < 6:
    print(i)
    i += 1
```

```
1
2
3
4
5
```

DIFFERENCE BETWEEN FOR LOOP AND WHILE LOOP

The 'for' loop used only when we already knew the number of iterations. The 'while' loop used only when the number of iteration are not exactly known. If the condition is not put up in 'for' loop, then loop iterates infinite times. In 'while' loop, the iteration statement can be written anywhere in the loop.

DAY-7

PROJECT HANGMAN

CHALLANGE-1-->PICKING UP THE RANDOM WORDS AND CHECKING ANSWERS

In [7]:
```
#Step 1

word_list = ["aardvark", "baboon", "camel"]

#TODO-1 - Randomly choose a word from the word_list and assign it to a variable called
import random
chosen_word = random.choice(word_list)

#TODO-2 - Ask the user to guess a letter and assign their answer to a variable called g
guess = input("Guess a letter: ").lower()

#TODO-3 - Check if the letter the user guessed (guess) is one of the leters in the chos
for letter in chosen_word:
    if letter == guess:
        print("Right")
    else:
        print("Wrong")
```

```
Wrong
Right
Wrong
Wrong
Wrong
Wrong
```

CHALLANGE-2-->HOW TO REPLACE THE BLANKS

In [8]:
```
#Step 2

import random
```

```
word_list = ["aardvark", "baboon", "camel"]
chosen_word = random.choice(word_list)

#Testing code
print(f'Pssst, the solution is {chosen_word}.')

#TODO-1: - Create an empty List called display.
#For each letter in the chosen_word, add a "_" to 'display'.
#So if the chosen_word was "apple", display should be ["_", "_", "_", "_", "_"] with
#5 "_" representing each letter to guess.
display = []
word_length = len(chosen_word)
for _ in range(word_length):
    display += "_"

guess = input("Guess a letter: ").lower()

#TODO-2: - Loop through each position in the chosen_word;
#If the letter at that position matches 'guess' then reveal that letter in the display
#e.g. If the user guessed "p" and the chosen word was "apple", then display should be [
for position in range(word_length):
    letter = chosen_word[position]
    #print(f"Current position: {position}\n Current letter: {letter}\n Guessed letter:
    if letter == guess:
        display[position] = letter

#TODO-3: - Print 'display' and you should see the guessed letter in the correct positio
#other letter replace with "_".
#Hint - Don't worry about getting the user to guess the next letter. We'll tackle that
print(display)
```

```
Pssst, the solution is aardvark.
```

```
['a', 'a', '_', '_', '_', 'a', '_', '_']
```

CHALLANGE-3-->CHECKING IF THE PLAYER HAS WON

In [9]:

```
#Step 3

import random
word_list = ["aardvark", "baboon", "camel"]
chosen_word = random.choice(word_list)
word_length = len(chosen_word)

#Testing code
print(f'Pssst, the solution is {chosen_word}.')

#Create blanks
display = []
for _ in range(word_length):
    display += "_"

#TODO-1: - Use a while loop to let the user guess again. The loop should only stop
#once the user has guessed all the letters in the chosen_word and 'display' has no more
#Then you can tell the user they've won.
end_of_game = False

while not end_of_game:
    guess = input("Guess a letter: ").lower()

    #Check guessed letter
```

```
    for position in range(word_length):
        letter = chosen_word[position]
        #print(f"Current position: {position}\n Current letter: {letter}\n Guessed lett
        if letter == guess:
            display[position] = letter

    print(display)

    #Check if there are no more "_" left in 'display'. Then all letters have been guess
    if "_" not in display:
        end_of_game = True
        print("You win.")
```

```
Pssst, the solution is baboon.
['_', 'a', '_', '_', '_', '_']
['_', 'a', '_', '_', '_', '_']
['_', 'a', '_', '_', '_', '_']
['_', 'a', '_', '_', '_', '_']
['_', 'a', '_', '_', '_', '_']
['b', 'a', 'b', '_', '_', '_']
['b', 'a', 'b', '_', '_', 'n']
['b', 'a', 'b', 'o', 'o', 'n']
You win.
```

CHALLANGE-4-->KEEPING TRACK OF THE PLAYER'S LIVES

In [2]:

```
#Step 4

import random

stages = ['''
  +---+
  |   |
  O   |
 /|\  |
 / \  |
      |
=========
''', '''
  +---+
  |   |
  O   |
 /|\  |
 /    |
      |
=========
''', '''
  +---+
  |   |
  O   |
 /|\  |
      |
      |
=========
''', '''
```

```
  +---+
  |   |
  O   |
 /|   |
      |
      |
=========''', '''
  +---+
  |   |
  O   |
  |   |
      |
      |
=========
''', '''
  +---+
  |   |
  O   |
      |
      |
      |
=========
''', '''
  +---+
  |   |
      |
      |
      |
      |
=========
''']

end_of_game = False
word_list = ["ardvark", "baboon", "camel"]
chosen_word = random.choice(word_list)
word_length = len(chosen_word)

#TODO-1: - Create a variable called 'lives' to keep track of the number of lives left.
#Set 'lives' to equal 6.
lives=6

#Testing code
print(f'Pssst, the solution is {chosen_word}.')

#Create blanks
display = []
for _ in range(word_length):
    display += "_"

while not end_of_game:
    guess = input("Guess a letter: ").lower()

    #Check guessed letter
    for position in range(word_length):
        letter = chosen_word[position]
        # print(f"Current position: {position}\n Current letter: {letter}\n Guessed let
        if letter == guess:
            display[position] = letter


    #TODO-2: - If guess is not a letter in the chosen_word,
```

```
    #Then reduce 'lives' by 1.
    #If lives goes down to 0 then the game should stop and it should print "You lose."
    if guess not in chosen_word:
        lives -=1

        if lives==0:
            print("You lose")
            end_of_game=True

    #Join all the elements in the list and turn it into a String.
    print(f"{' '.join(display)}")

    #Check if user has got all letters.
    if "_" not in display:
        end_of_game = True
        print("You win.")

    #TODO-3: - print the ASCII art from 'stages' that corresponds to the current number
    print(stages[lives])
```

```
Pssst, the solution is camel.

_ a _ _ _

  +---+
  |   |
      |
      |
      |
      |
=========


c a _ _ _

  +---+
  |   |
      |
      |
      |
      |
=========


c a _ _ _

  +---+
  |   |
  O   |
      |
      |
      |
=========


c a _ _ _

  +---+
  |   |
  O   |
  |   |
      |
      |
```

```
=========


c a m _ _

  +---+
  |   |
  O   |
  |   |
      |
      |
=========


c a m e _

  +---+
  |   |
  O   |
  |   |
      |
      |
=========


c a m e l
You win.

  +---+
  |   |
  O   |
  |   |
      |
      |
=========
```

CHALLANGE-5-->IMPROVING THE USERS EXPERIENCE

file 1: hangman_art.py stages = [''' +---+ | | O | /|\ | / \ | | ========= ''', ''' +---+ | | O | /|\ | / | | ========= ''', ''' +---+ | | O | /|\ | | | ========= ''', ''' +---+ | | O | /| | | | ========='', ''' +---+ | | O | | | | ========= ''', ''' +---+ | | O | | | | ========= ''', ''' +---+ | | | | | | ========= '''] logo = ''' _ | | | |__ __ _ _ __ __ _ _ _ __ ___ __ _ _ __ | '_ \ / _` | '_ \ / _` | '_ ` _ \ / _` | '_ \ | | | (_| | | | | (_| | | | | | | (_| | | | | |_| |_|__,_|_| |_|__, |_| |_| |_|__,_|_| |_| __/ | |___/ ''' file 2: hangman_words.py word_list = ['abruptly', 'absurd', 'abyss', 'affix', 'askew', 'avenue', 'awkward', 'axiom', 'azure', 'bagpipes', 'bandwagon', 'banjo', 'bayou', 'beekeeper', 'bikini', 'blitz', 'blizzard', 'boggle', 'bookworm', 'boxcar', 'boxful', 'buckaroo', 'buffalo', 'buffoon', 'buxom', 'buzzard', 'buzzing', 'buzzwords', 'caliph', 'cobweb', 'cockiness', 'croquet', 'crypt', 'curacao', 'cycle', 'daiquiri', 'dirndl', 'disavow', 'dizzying', 'duplex', 'dwarves', 'embezzle', 'equip', 'espionage', 'euouae', 'exodus', 'faking', 'fishhook', 'fixable', 'fjord', 'flapjack', 'flopping', 'fluffiness', 'flyby', 'foxglove', 'frazzled', 'frizzled', 'fuchsia', 'funny', 'gabby', 'galaxy', 'galvanize', 'gazebo', 'giaour', 'gizmo', 'glowworm', 'glyph', 'gnarly', 'gnostic', 'gossip', 'grogginess', 'haiku', 'haphazard', 'hyphen', 'iatrogenic', 'icebox', 'injury', 'ivory', 'ivy', 'jackpot', 'jaundice', 'jawbreaker', 'jaywalk', 'jazziest', 'jazzy', 'jelly', 'jigsaw', 'jinx', 'jiujitsu', 'jockey', 'jogging', 'joking', 'jovial', 'joyful', 'juicy', 'jukebox', 'jumbo', 'kayak', 'kazoo', 'keyhole', 'khaki', 'kilobyte', 'kiosk', 'kitsch', 'kiwifruit', 'klutz', 'knapsack', 'larynx', 'lengths', 'lucky', 'luxury', 'lymph', 'marquis', 'matrix', 'megahertz', 'microwave', 'mnemonic', 'mystify', 'naphtha', 'nightclub', 'nowadays', 'numbskull', 'nymph', 'onyx', 'ovary', 'oxidize', 'oxygen', 'pajama', 'peekaboo', 'phlegm', 'pixel', 'pizazz', 'pneumonia', 'polka', 'pshaw', 'psyche', 'puppy', 'puzzling', 'quartz', 'queue', 'quips', 'quixotic', 'quiz', 'quizzes', 'quorum', 'razzmatazz', 'rhubarb', 'rhythm', 'rickshaw', 'schnapps', 'scratch', 'shiv', 'snazzy', 'sphinx', 'spritz', 'squawk', 'staff', 'strength', 'strengths', 'stretch', 'stronghold', 'stymied', 'subway', 'swivel', 'syndrome', 'thriftless', 'thumbscrew', 'topaz', 'transcript', 'transgress', 'transplant', 'triphthong', 'twelfth', 'twelfths', 'unknown', 'unworthy', 'unzip', 'uptown', 'vaporize', 'vixen', 'vodka', 'voodoo', 'vortex', 'voyeurism', 'walkway', 'waltz', 'wave', 'wavy', 'waxy', 'wellspring', 'wheezy', 'whiskey', 'whizzing', 'whomever', 'wimpy', 'witchcraft', 'wizard', 'woozy', 'wristwatch', 'wyvern', 'xylophone', 'yachtsman', 'yippee', 'yoked', 'youthful', 'yummy', 'zephyr', 'zigzag', 'zigzagging', 'zilch', 'zipper', 'zodiac', 'zombie',]file3:main.py #Step 5 import random import hangman_art import hangman_words

#TODO-1: - Update the word list to use the 'word_list' from hangman_words.py chosen_word = random.choice(hangman_words.word_list) word_length = len(chosen_word) end_of_game = False lives = 6 #TODO-3: - Import the logo from hangman_art.py and print it at the start of the game. print(hangman_art.logo) #Testing code print(f'Pssst, the solution is {chosen_word}.') #Create blanks display = [] for _ in range(word_length): display += "_" while not end_of_game: guess = input("Guess a letter: ").lower() #TODO-4: - If the user has entered a letter they've already guessed, print the letter and let them know. if guess == display: print("The letter is already guessed") #Check guessed letter for position in range(word_length): letter = chosen_word[position] #print(f"Current position: {position}\n Current letter: {letter}\n Guessed letter: {guess}") if letter == guess: display[position] = letter #Check if user is wrong. if guess not in chosen_word: #TODO-5: - If the letter is not in the chosen_word, print out the letter and let them know it's not in the word. print("Sorry the letter is not in the word.Try another letter") lives -= 1 if lives == 0: end_of_game = True print("You lose.") #Join all the elements in the list and turn it into a String. print(f"{' '.join(display)}") #Check if user has got all letters. if "_" not in display: end_of_game = True print("You win.") #TODO-2: - Import the stages from hangman_art.py and make this error go away. print(hangman_art.stages[lives]) Output: _ | | | |__ __ _ _ __ __ _ _ _ __ ___ __ _ _ _ __ | '_ \ / _` | '_ \ / _` | '_ ` _ \ / _` | '_ \| | | | (_| | | | | (_| | | | | | | (_| | | | | |_| |_|__,_|_| |_|__, |_| |_| |_|__,_|_| |_| __/ | |___/ Pssst, the solution is quizzes. Guess a letter: q q _ _ _ _ _ _ +---+ | | | | | | ========= Guess a letter: a Sorry the letter is not in the word_list.Try another word q _ _ _ _ _ _ +---+ | | O | | | | ========= Guess a letter: q q _ _ _ _ _ _ +---+ | | O | | | | ========= Guess a letter: q q _ _ _ _ _ _ +---+ | | O | | | | ========= Guess a letter: q q _ _ _ _ _ _ +---+ | | O | | | | ========= Guess a letter: e q _ _ _ _ e _ +---+ | | O | | | | ========= Guess a letter: t Sorry the letter is not in the word_list.Try another word q _ _ _ _ e _ +---+ | | O | | | | ========= Guess a letter: r Sorry the letter is not in the word_list.Try another word q _ _ _ _ e _ +---+ | | O | /| | | | ========= Guess a letter: r Sorry the letter is not in the word_list.Try another word q _ _ _ _ e _ +---+ | | O | /|\ | | | ========= Guess a letter: y Sorry the letter is not in the word_list.Try another word q _ _ _ _ e _ +---+ | | O | /|\ | / | | ========= Guess a letter: r Sorry the letter is not in the word_list.Try another word You lose. q _ _ _ _ e _ +---+ | | O | /|\ | / \ | | =========

HANGMAN'S FINAL CODE:

import random from hangman_art import stages, logo from hangman_words import word_list from replit import clear print(logo) game_is_finished = False lives = len(stages) - 1 chosen_word = random.choice(word_list) word_length = len(chosen_word) display = [] for _ in range(word_length): display += "_" while not game_is_finished: guess = input("Guess a letter: ").lower() #Use the clear() function imported from replit to clear the output between guesses. clear() if guess in display: print(f"You've already guessed {guess}") for position in range(word_length): letter = chosen_word[position] if letter == guess: display[position] = letter print(f"{' '.join(display)}") if guess not in chosen_word: print(f"You guessed {guess}, that's not in the word. You lose a life.") lives -= 1 if lives == 0: game_is_finished = True print("You lose.") if not "_" in display: game_is_finished = True print("You win.") print(stages[lives])

DAY-8

FUNCTIONS WITH CALLING IT

In [1]:
```
def greet():                                # User defined Function
    print("Hello world")
    print("Hello world")
    print("Hello world")

greet() #Calling the function
```

```
Hello world
Hello world
Hello world
```

FUNCTIONS WITH INPUTS

Syntax: def function(something given): do this finally do this function(sothing given)--->Here name is Parameter and "Srenath" is argument

In [5]:
```
def hellooo(name):
    print("Hello",name)
    print("How are you",name)

hellooo("Srenath")
```

```
Hello Srenath
How are you Srenath
```

FUNCTIONS WITH MORE INPUTS

In [6]:
```
def city(name,location):
    print("Hello",name)
    print("I am from",location)
city(input("Enter your name"),input("Enter your location"))
```

```
Hello Srenath kumar
I am from Chennai
```

POSITIONAL ARGUMENTS

In [9]:
```
def numbers(a,b,c):
    print(a)
    print(b)
    print(c)

numbers(1,2,3)
```

```
1
2
3
```

In [10]:
```
def numbers(a,b,c):
    print(a)
    print(b)
    print(c)
numbers(3,1,2)
```

```
3
1
2
```

-->Here when the postion of the argument is changed the values are also changed

Keyword Arguments

In [11]:
```
def numbers(a,b,c):
    print(a)
    print(b)
    print(c)
numbers(c=3,a=1,b=2)
```

```
1
2
3
```

-->Here even thought we change the position of the values we get the same and required results

CODING EXERCISE-PAINT AREA CALCULATOR

In [12]:
```
#Write your code below this line 👇
import math
def paint_calc(height,width,cover):
    area=(height*width)
    #Here we use ciel instead of round because it helps to round off +2
    #Eg is the nunmber is 4.3 it converts it into 5
    num_of_cans=math.ceil(area/cover)
    print("You would need",num_of_cans,"of paint")


#Write your code above this line 👆
# Define a function called paint_calc() so that the code below works.

# 📬 Don't change the code below 👇
test_h = int(input("Height of wall: "))
test_w = int(input("Width of wall: "))
coverage = 5
paint_calc(height=test_h, width=test_w, cover=coverage)
```

```
You would need 6 of paint
```

CODING EXERCISE:PRIME NUMBER CHECKER

In [13]:
```
def prime(num):
    if num>1:
        s=int(num/2)
        for i in range(2,s+1):
            if num%i==0:
                return("not prime")
                break
        return("prime")
print(prime(239))
```

```
prime
```

DAY-8-PROJECT-CEASAR CIPHER

CHALLANGE-1-->ENCRYPTION

In [14]:
```
alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
            'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f',
            'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',

direction = input("Type 'encode' to encrypt, type 'decode' to decrypt:\n")
text = input("Type your message:\n").lower()
shift = int(input("Type the shift number:\n"))

#Don't change the code above 👆

#TODO-1: Create a function called 'encrypt' that takes the 'text' and 'shift' as inputs
def encrypt(plain_text, shift_amount):
  #TODO-2: Inside the encrypt function, shift each letter of the text forwards in the a
  #shift amount and print the encrypted text.
  #e.g.
  #plain_text = "hello"
  #shift = 5
  #cipher_text = "mjqqt"
  #print output: "The encoded text is mjqqt"
    #Creating an empty list
```

```
    cipher_text = ""
    for letter in plain_text:
        #To get the position of the alphabet in the list
        position = alphabet.index(letter)
        #getting new position by adding the old position with the shift number
        new_position = position + shift_amount
        #getting the new letter with alphabet new index
        new_letter = alphabet[new_position]
        #adding the text to cipher_text list
        cipher_text += new_letter
        #finally printing the text
        print(f"The encoded text is {cipher_text}")

#TODO-3: Call the encrypt function and pass in the user inputs. You should be able to t
encrypt(plain_text=text, shift_amount=shift)
```

```
The encoded text is m
The encoded text is mj
The encoded text is mjq
The encoded text is mjqq
The encoded text is mjqqt
```

CHALLANGE-2-->DECRYPTION

In [16]:

```
alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
            'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
            'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
            'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

direction = input("Type 'encode' to encrypt, type 'decode' to decrypt:\n")


def encrypt(plain_text, shift_amount):
    cipher_text = ""
    for letter in plain_text:
            position = alphabet.index(letter)
            new_position = position + shift_amount
            cipher_text += alphabet[new_position]
            print(f"The encoded text is {cipher_text}")

#TODO-1: Create a different function called 'decrypt' that takes the 'text' and 'shift'
def decrypt(plain_text,shift_amount):
        original_text=""
        for letter in plain_text:
            position=alphabet.index(letter)
            #Reducing the shift number
            new_position = position-shift_amount
            original_text +=alphabet[new_position]
            print(f"The decoded message is {original_text}")



  #TODO-2: Inside the 'decrypt' function, shift each letter of the 'text' *backwards* i
  #alphabet by the shift amount and print the decrypted text.
  #e.g.
  #cipher_text = "mjqqt"
  #shift = 5
  #plain_text = "hello"
  #print output: "The decoded text is hello"
```

```
#TODO-3: Check if the user wanted to encrypt or decrypt the message by checking the
#'direction' variable. Then call the correct function based on that 'drection' variable
#You should be able to test the code to encrypt *AND* decrypt a message.
if direction == "encode":
    text = input("Type your message:\n").lower()
    shift = int(input("Type the shift number:\n"))
    encrypt(plain_text=text, shift_amount=shift)

elif direction == "decode":
    text = input("Type your message:\n").lower()
    shift = int(input("Type the shift number:\n"))
    decrypt(plain_text=text,shift_amount=shift)
else:
    print("Invalid Input")
```

```
The decoded message is h
The decoded message is he
The decoded message is hel
The decoded message is hell
The decoded message is hello
```

CHALLANGE-3-->REORGANISING OUR CODE

In [19]:

```
#Creating a function to combine the encrypt and decrypt function into a single function

alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
            'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
            'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
            'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

direction = input("Type 'encode' to encrypt, type 'decode' to decrypt:\n")
text = input("Type your message:\n").lower()
shift = int(input("Type the shift number:\n"))

#TODO-1: Combine the encrypt() and decrypt() functions into a single function called ca
def ceasor(plain_text,shift_amount):
    ceasor_text=""
    if direction == "decode":
            shift_amount *= -1
    for letter in plain_text:
        position=alphabet.index(letter)
        new_position=position+shift_amount
        ceasor_text += alphabet[new_position]
        print(f"The encoded text is {ceasor_text}")




#TODO-2: Call the caesar() function, passing over the 'text', 'shift' and 'direction' v
ceasor(text,shift)
```

```
The encoded text is h
The encoded text is he
The encoded text is hel
The encoded text is hell
The encoded text is hello
```

CHALLANGE-4-->USER EXPERIENCE IMPROVEMENTS AND FINAL CODE

File1: art.py logo = """ ,adPPYba, ,adPPYYba, ,adPPYba, ,adPPYba, ,adPPYYba, 8b,dPPYba, a8" "" "" `Y8 a8P_____88 I8[ "" "" `Y8 88P' "Y8 8b ,adPPPPP88 8PP""""""" `"Y8ba, ,adPPPPP88 88 "8a, ,aa 88, ,88 "8b, ,aa aa ]8I 88, ,88 88 `"Ybbd8"' `"8bbdP"Y8 `"Ybbd8"' `"YbbdP"' `"8bbdP"Y8 88 88 88 "" 88 88 ,adPPYba, 88 8b,dPPYba, 88,dPPYba, ,adPPYba, 8b,dPPYba, a8" "" 88 88P' "8a 88P' "8a a8P_____88 88P' "Y8 8b 88 88 d8 88 88 8PP""""""" 88 "8a, ,aa 88 88b, ,a8" 88 88 "8b, ,aa 88 `"Ybbd8"' 88 88`YbbdP"' 88 88 `"Ybbd8"' 88 88 88 """ FILE 2: alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'] def caesar(start_text, shift_amount, cipher_direction): end_text = "" if cipher_direction == "decode": shift_amount *= -1 for char in start_text: #TODO-3: What happens if the user enters a number/symbol/space? #Can you fix the code to keep the number/symbol/space when the text is encoded/decoded? #e.g. start_text = "meet me at 3" #end_text = "•••• •• •• 3" if char in alphabet: position = alphabet.index(char) new_position = position + shift_amount end_text += alphabet[new_position] else: end_text += char print(f"Here's the {cipher_direction}d result: {end_text}") #TODO-1: Import and print the logo from art.py when the program starts. from art import logo print(logo) #TODO-4: Can you figure out a way to ask the user if they want to restart the cipher program? #e.g. Type 'yes' if you want to go again. Otherwise type 'no'. #If they type 'yes' then ask them for the direction/text/shift again and call the caesar() function again? #Hint: Try creating a while loop that continues to execute the program if the user types 'yes'. should_end = False while not should_end: direction = input("Type 'encode' to encrypt, type 'decode' to decrypt:\n") text = input("Type your message:\n").lower() shift = int(input("Type the shift number:\n")) #TODO-2: What if the user enters a shift that is greater than the number of letters in the alphabet? #Try running the program and entering a shift number of 45. #Add some code so that the program continues to work even if the user enters a shift number greater than 26. #Hint: Think about how you can use the modulus (%). shift = shift % 26 caesar(start_text=text, shift_amount=shift, cipher_direction=direction) restart = input("Type 'yes' if you want to go again. Otherwise type 'no'.\n") if restart == "no": should_end = True print("Goodbye") OUTPUT: ,adPPYba, ,adPPYYba, ,adPPYba, ,adPPYba, ,adPPYYba, 8b,dPPY ,adPPYba, ,adPPYYba, ,adPPYba, ,adPPYba, ,adPPYYba, 8b,dPPYba, a8" "" "" `Y8 a8P_____88 I8[ "" "" `Y8 88P' "Y8 8b ,adPPPPP88 8PP" `"Y8ba, ,adPPPPP88 88 "8a, ,aa 88, ,88 "8b, ,aa aa ]8I 88, ,88 88 `"Ybbd8"' `"8bbdP"Y8 `"Ybbd8"' `"YbbdP"' `"8bbdP"Y8 88 88 88 "" 88 88 ,adPPYba, 88 8b,dPPYba, 88,dPPYba, ,adPPYba, 8b,dPPYba, a8" "" 88 88P' "8a 88P' "8a a8P_____88 88P' "Y8 8b 88 88 d8 88 88 8PP" 88 "8a, ,aa 88 88b, ,a8" 88 88 "8b, ,aa 88 `"Ybbd8"' 88 88`YbbdP"' 88 88 `"Ybbd8"' 88 88 88 Type 'encode' to encrypt, type 'decode' to decrypt: encode Type your message: hellos Type the shift number: 6 Here's the encoded result: nkrruy Type 'yes' if you want to go again. Otherwise type 'no'. yes Type 'encode' to encrypt, type 'decode' to decrypt: decode Type your message: nkrruy Type the shift number: 6 Here's the decoded result: hellos Type 'yes' if you want to go again. Otherwise type 'no'. no Goodbye

DAY-9

DICTIONARIES

SYNTAX: dict={key1:value1,key2=value2}

In [11]:
```
dictionary={"fruits":"apple","vegetables":"onions"}
dictionary["fruits"]
```

Out[11]:
```
'apple'
```

ADDING NEW ITEMS TO DICTIONARY

In [12]:
```
dictionary["green leafs"]="spinach"
```

In [13]:
```
dictionary
```

Out[13]:
```
{'fruits': 'apple', 'vegetables': 'onions', 'green leafs': 'spinach'}
```

CREATING EMPTY DICTIONARY

In [6]:
```
DICTIONARY1={}
```

In [8]:
```
DICTIONARY1
```

Out[8]: {}

WIPING OUT ENTIRE EXISTING DICTIONARY

In [10]:
```
#It totally wipes out everything
dictionary={}
dictionary
```

Out[10]: {}

EDITING AN ITEM IN DICTIONARY

In [14]:
```
dictionary["fruits"]="mango"
```

In [15]:
```
dictionary
```

Out[15]: {'fruits': 'mango', 'vegetables': 'onions', 'green leafs': 'spinach'}

LOOP THROUGH DICTIONARY TO PRINT KEYS

In [16]:
```
#It prints out all the keys in the dictionary
for thing in dictionary:
    print(thing)
```

```
fruits
vegetables
green leafs
```

LOOP THROUGH DICTIONARY TO PRINT VALUES

In [20]:
```
for key in dictionary:
    print(key)
    print(dictionary[key])
```

```
fruits
mango
vegetables
onions
green leafs
spinach
```

CODING EXERCISE-GRADING PROGRAM

In [21]:
```
student_scores = {
  "Harry": 81,
  "Ron": 78,
  "Hermione": 99,
```

```
  "Draco": 74,
  "Neville": 62,
}
# 📥 Don't change the code above 👆

#TODO-1: Create an empty dictionary called student_grades.
student_grades={}

#TODO-2: Write your code below to add the grades to student_grades.👇
for i in student_scores:
    if student_scores[i] >=91 and student_scores[i]<=100:
        student_grades[i]="Outstanding"
    elif student_scores[i]>=81 and student_scores[i]<=90:
        student_grades[i]="Exceeds Expetations"
    elif student_scores[i]>=71 and student_scores[i]<=80:
        student_grades[i]="Acceptable"
    elif student_scores[i]<=70:
        student_grades[i]="Fail"


# 📥 Don't change the code below 👇
print(student_grades)
```

```
{'Harry': 'Exceeds Expetations', 'Ron': 'Acceptable', 'Hermione': 'Outstanding', 'Drac
o': 'Acceptable', 'Neville': 'Fail'}
```

NESTING LISTS AND DICTIONARIES

IT MEANS THAT ADDING LISTS AND DICTIONARIES IN ANOTHER DICTIONATIES IS NESTING LISTS AND DICTIONARIES SYNTAX: dictionary={key1:[list], key2:{dictionary1} }

NESTING A LIST IN DICTIONARY

In [1]:
```
dictionary2={"France":["Parris","Djion","Lilee"],
             "Germany":["Berlin","Hamburg"]}
dictionary2
```

Out[1]:
```
{'France': ['Parris', 'Djion', 'Lilee'], 'Germany': ['Berlin', 'Hamburg']}
```

NESTING DICTIONARY IN A DICTIONARY

In [7]:
```
travelog={"France":{"city_visited":["Parris","Djion","Lilee"],"Total_no_visits":1},
          "Germany":{"city_visited":["Berlin","Hamburg"],"Total_no_visits":4}}
```

In [8]:
```
travelog
```

Out[8]:
```
{'France': {'city_visited': ['Parris', 'Djion', 'Lilee'],
  'Total_no_visits': 1},
 'Germany': {'city_visited': ['Berlin', 'Hamburg'], 'Total_no_visits': 4}}
```

NESTING A DICTIONARY IN A LISTS

In [9]:
```
travellog1=[
    {"Country":"France","City_Visited":["Parris","Djion","Lilee"]},{"Country":"Germany"
```

In [10]:
```
travellog1
```

Out[10]:
```
[{'Country': 'France', 'City_Visited': ['Parris', 'Djion', 'Lilee']},
 {'Country': 'Germany', 'City_visited': ['Berlin', 'Hamburg']}]
```

In [11]:
```
type(travellog1)
```

Out[11]:
```
list
```

CODING-EXERCISE:DICTIONARY IN LIST

In [12]:
```
travel_log = [
{
  "country": "France",
  "visits": 12,
  "cities": ["Paris", "Lille", "Dijon"]
},
{
  "country": "Germany",
  "visits": 5,
  "cities": ["Berlin", "Hamburg", "Stuttgart"]
},
]
#DO NOT change the code above 👆

#TODO: Write the function that will allow new countries
#to be added to the travel_log.
def add_new_country(name, visit_count, cities_visited):
    new_country = {}
    new_country["country"] = name
    new_country["visits"] = visit_count
    new_country["cities"] = cities_visited
    travel_log.append(new_country)

#Do not change the code below 👇
add_new_country("Russia", 2, ["Moscow", "Saint Petersburg"])
print(travel_log)
```

```
[{'country': 'France', 'visits': 12, 'cities': ['Paris', 'Lille', 'Dijon']}, {'country':
'Germany', 'visits': 5, 'cities': ['Berlin', 'Hamburg', 'Stuttgart']}, {'country': 'Russ
ia', 'visits': 2, 'cities': ['Moscow', 'Saint Petersburg']}]
```

DAY-9:CODING CHALLANGE-->SECRET AUCTION PROGRAM

In [16]:
```
bids = {}
bidding_finished = False

def find_highest_bidder(bidding_record):
    highest_bid = 0
    winner = ""
    # bidding_record = {"Angela": 123, "James": 321}
    for bidder in bidding_record:
        bid_amount = bidding_record[bidder]
        if bid_amount > highest_bid:
            highest_bid = bid_amount
            winner = bidder
            print(f"The winner is {winner} with a bid of ${highest_bid}")

while not bidding_finished:
    name = input("What is your name?: ")
```

```
    price = int(input("What is your bid?: $"))
    bids[name] = price
    should_continue = input("Are there any other bidders? Type 'yes or 'no'.\n")
    if should_continue == "no":
        bidding_finished = True
        find_highest_bidder(bids)
```

```
The winner is srenath kuamr with a bid of $500
```

DAY-10

FUNCTIONS WITH OUTPUTS

In [19]:
```
def function(num1,num2):
    return num1*num2
```

In [21]:
```
# To convert a string to titled name
def format_name(fname,lname):
    formatted_fname=fname.title()
    formatted_lname=lname.title()
    print(formatted_fname," ",formatted_lname)
format_name("SRenath","KUMar")
```

```
Srenath   Kumar
```

In [24]:
```
# To convert a string to titled name and with return functions
def format_name(fname,lname):
    if fname=="" and lname=="":
        return "Please Provide a name"
    formatted_fname=fname.title()
    formatted_lname=lname.title()
    print(formatted_fname," ",formatted_lname)
format_name(input("Enter the first name"),input("Enter the last name"))
```

Out[24]:
```
'Please Provide a name'
```

CODING EXERCISE-->DAYS IN A MONTH

In [25]:
```
def is_leap(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True
            else:
                return False
        else:
                return True
    else:
        return False

def days_in_month(year, month):
```

```
    month_days = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    if month > 12 or month < 1:
        return "Invalid month entered."
    if month == 2 and is_leap(year):
        return 29
    return month_days[month - 1]



#Do NOT change any of the code below 👇
year = int(input("Enter a year: "))
month = int(input("Enter a month: "))
days = days_in_month(year, month)
print(days)
```

```
31
```

DOCSTRINGS

In [26]:

```
#Dcostrings means that taking notes of the code which is more than one line
#Eg:
"""dsadsadasdsdsdsd
sdadsadadsadsadsadsad
sdasdasdkfjdlfjioep"""
#It is done with three semicoluns
```

Out[26]:

```
'dsadsadasdsdsdsd\nsdadsadadsadsadsadsad\nsdasdasdkfjdlfjioep'
```

DAY-10-->BUILDING A CALCULATOR

In [ ]:

```
def add(n1, n2):
    return n1 + n2

def subtract(n1, n2):
    return n1 - n2

def multiply(n1, n2):
    return n1 * n2

def divide(n1, n2):
    return n1 / n2

operations = {
  "+": add,
  "-": subtract,
  "*": multiply,
  "/": divide
}

def calculator():
    num1 = float(input("What's the first number?: "))
    for symbol in operations:
        print(symbol)
        should_continue = True
    while should_continue:
        operation_symbol = input("Pick an operation: ")
        num2 = float(input("What's the next number?: "))
        calculation_function = operations[operation_symbol]
```

```
        answer = calculation_function(num1, num2)
        print(f"{num1} {operation_symbol} {num2} = {answer}")
        if input(f"Type 'y' to continue calculating with {answer}, or type 'n' to start
            num1 = answer
        else:
            should_continue = False
            calculator()

calculator()
```

Output: What's the first number?: 5 + - * / Pick an operation: + What's the next number?: 5 5.0 + 5.0 = 10.0 Type 'y' to continue calculating with 10.0, or type 'n' to start a new calculation: y Pick an operation: - What's the next number?: 4 10.0 - 4.0 = 6.0 Type 'y' to continue calculating with 6.0, or type 'n' to start a new calculation: n What's the first number?: 2 + - * / Pick an operation: + What's the next number?: 2 2.0 + 2.0 = 4.0

DAY-11

BLACKJACK CAPSTONE PROJECT

In [ ]:

```
import random


def deal_card():
    """Returns a random card from the deck."""
    cards = [11, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10]
    card = random.choice(cards)
    return card

#Hint 6: Create a function called calculate_score() that takes a List of cards as input
#and returns the score.
#Look up the sum() function to help you do this.
def calculate_score(cards):
    """Take a list of cards and return the score calculated from the cards"""

    #Hint 7: Inside calculate_score() check for a blackjack (a hand with only 2 cards:
    if sum(cards) == 21 and len(cards) == 2:
        return 0
    #Hint 8: Inside calculate_score() check for an 11 (ace). If the score is already ov
    if 11 in cards and sum(cards) > 21:
        cards.remove(11)
        cards.append(1)
    return sum(cards)

#Hint 13: Create a function called compare() and pass in the user_score and computer_sc
def compare(user_score, computer_score):
  #Bug fix. If you and the computer are both over, you lose.
  if user_score > 21 and computer_score > 21:
    return "You went over. You lose 😭"


  if user_score == computer_score:
    return "Draw 🙃"
  elif computer_score == 0:
    return "Lose, opponent has Blackjack 😱"
  elif user_score == 0:
    return "Win with a Blackjack 😎"
  elif user_score > 21:
    return "You went over. You lose 😭"
```

```
  elif computer_score > 21:
    return "Opponent went over. You win 😁"
  elif user_score > computer_score:
    return "You win 😃"
  else:
    return "You lose 😱"

def play_game():



  #Hint 5: Deal the user and computer 2 cards each using deal_card()
    user_cards = []
    computer_cards = []
    is_game_over = False

    for _ in range(2):
        user_cards.append(deal_card())
        computer_cards.append(deal_card())

  #Hint 11: The score will need to be rechecked with every new card drawn and the check

    while not is_game_over:
        #Hint 9: Call calculate_score(). If the computer or the user has a blackjack (0
        user_score = calculate_score(user_cards)
        computer_score = calculate_score(computer_cards)
        print(f"   Your cards: {user_cards}, current score: {user_score}")
        print(f"   Computer's first card: {computer_cards[0]}")

        if user_score == 0 or computer_score == 0 or user_score > 21:
           is_game_over = True
        else:
           #Hint 10: If the game has not ended, ask the user if they want to draw anoth
           user_should_deal = input("Type 'y' to get another card, type 'n' to pass: ")
            if user_should_deal == "y":
                user_cards.append(deal_card())
            else:
                is_game_over = True

  #Hint 12: Once the user is done, it's time to let the computer play. The computer sho

    while computer_score != 0 and computer_score < 17:
         computer_cards.append(deal_card())
           computer_score = calculate_score(computer_cards)

    print(f"   Your final hand: {user_cards}, final score: {user_score}")
    print(f"   Computer's final hand: {computer_cards}, final score: {computer_score}")
    print(compare(user_score, computer_score))

#Hint 14: Ask the user if they want to restart the game. If they answer yes, clear the
while input("Do you want to play a game of Blackjack? Type 'y' or 'n': ") == "y":

  play_game()
```

DAY-12

LOCAL SCOPE

In [5]:
```
variable=10  #Global Variable

def hello():
    variable=2 #Local Variable
    print(variable)

    # When the local variable is given the variable takes only local variable
hello()
```

```
2
```

GLOBAL SCOPE

In [7]:
```
variable=10  #Global Variable

def hello():

    print(variable)

    # When the local variable is not given the variable takes global variable
hello()
```

```
10
```

There is no block Scope in python

In [11]:
```
game_level=3

def create_enemy():
    enemies=["Skeleton","Zombie","Alien"]
    if game_level <5:
        #In python we can create a variable inside a function or in if else condition t
        new_enemy=enemies[0]

    print(new_enemy)


create_enemy()
```

```
Skeleton
```

Modfying the global Scopes

In [ ]:
```
# The method down below is the best to handle with global variable and this type of cod
```

In [13]:
```
enemies=1

def increase_enemies():
    print("Printing enemies inside ",enemies)
    return enemies + 1
enemies=increase_enemies()
print("Printing enemies outside ",enemies)
```

```
Printing enemies inside  1
```

```
Printing enemies outside  2
```

In [16]:
```
enemies=1

def increase_enemies():
    global enemies
    enemies=3
    print("Printing enemies inside ",enemies)
increase_enemies()
print("Printing enemies outside ",enemies)
```

```
Printing enemies inside  3
Printing enemies outside  3
```

In [17]:
```
#The code above is extremenly dangerous as it just changes the global value on getting
```

Python Constants and Global Scope

In [21]:
```
# In programming if you are planning to not change the value of global variable use the

PI=3.14

def calculate():
    answer=PI*4
    print(answer)

calculate()
```

```
12.56
```

DAY-12 Project-->Guess the number

In [2]:
```
from random import randint


EASY_LEVEL_TURNS = 10
HARD_LEVEL_TURNS = 5

#Function to check user's guess against actual answer.
def check_answer(guess, answer, turns):
    """checks answer against guess. Returns the number of turns remaining."""
    if guess > answer:
        print("Too high.")
        return turns - 1
    elif guess < answer:
        print("Too low.")
        return turns - 1
    else:
        print(f"You got it! The answer was {answer}.")

#Make function to set difficulty.
def set_difficulty():
    level = input("Choose a difficulty. Type 'easy' or 'hard': ")
    if level == "easy":
        return EASY_LEVEL_TURNS
```

```
    else:
        return HARD_LEVEL_TURNS

def game():
    #Choosing a random number between 1 and 100.
    print("Welcome to the Number Guessing Game!")
    print("I'm thinking of a number between 1 and 100.")
    answer = randint(1, 100)
    print(f"Pssst, the correct answer is {answer}")

    turns = set_difficulty()
    #Repeat the guessing functionality if they get it wrong.
    guess = 0
    while guess != answer:
        print(f"You have {turns} attempts remaining to guess the number.")

        #Let the user guess a number.
        guess = int(input("Make a guess: "))

        #Track the number of turns and reduce by 1 if they get it wrong.
        turns = check_answer(guess, answer, turns)
        if turns == 0:
            print("You've run out of guesses, you lose.")
            return
        elif guess != answer:
            print("Guess again.")


game()
```

```
Welcome to the Number Guessing Game!
I'm thinking of a number between 1 and 100.
Pssst, the correct answer is 25

You have 10 attempts remaining to guess the number.

Too low.
Guess again.
You have 9 attempts remaining to guess the number.

Too low.
Guess again.
You have 8 attempts remaining to guess the number.

Too low.
Guess again.
You have 7 attempts remaining to guess the number.

Too high.
Guess again.
You have 6 attempts remaining to guess the number.

Too low.
Guess again.
You have 5 attempts remaining to guess the number.

Too low.
Guess again.
You have 4 attempts remaining to guess the number.

Too low.
Guess again.
You have 3 attempts remaining to guess the number.
```

```
Too low.
Guess again.
You have 2 attempts remaining to guess the number.

Too high.
Guess again.
You have 1 attempts remaining to guess the number.

Too high.
You've run out of guesses, you lose.
```

DAY-13

DEBUGGING

-->If you get an error try to split up and understand -->If the problems and errors arecluttered try to organize it into neat mannerTips of Debugging: 1)Describe the Problem 2)Reproduce the bug 3)Play Computer 4)Fix the errors 5)Print is yout friend 6)Use a Debugger 7)Take a break 8)Ask your friend 9)Run often 10)Ask StackOverflow

DAY-14

In [2]:

```
#art.py

logo = """
    __  ___         __
   / / / (_)___ _/ /_  ___  _____
  / /_/ / / __ `/ __ \/ _ \/ ___/
 / __  / / /_/ / / / /  __/ /
/_/ /_/_/\__, /_/ /_/\___/_/
   / /  /____/_      _____  _____
  / /   / __ \ | /| / / _ \/ ___/
 / /___/ /_/ / |/ |/ /  __/ /
/_____/\____/|__/|__/\___/_/
"""

vs = """
 _    __
| |  / /____
| | / / ___/
| |/ (__  )
|___/____(_)
"""
```

In [3]:

```
data = [
    {
        'name': 'Instagram',
        'follower_count': 346,
        'description': 'Social media platform',
        'country': 'United States'
    },
    {
        'name': 'Cristiano Ronaldo',
        'follower_count': 215,
        'description': 'Footballer',
        'country': 'Portugal'
    },
    {
```

```
        'name': 'Ariana Grande',
        'follower_count': 183,
        'description': 'Musician and actress',
        'country': 'United States'
    },
    {
        'name': 'Dwayne Johnson',
        'follower_count': 181,
        'description': 'Actor and professional wrestler',
        'country': 'United States'
    },
    {
        'name': 'Selena Gomez',
        'follower_count': 174,
        'description': 'Musician and actress',
        'country': 'United States'
    },
    {
        'name': 'Kylie Jenner',
        'follower_count': 172,
        'description': 'Reality TV personality and businesswoman and Self-Made Billiona
        'country': 'United States'
    },
    {
        'name': 'Kim Kardashian',
        'follower_count': 167,
        'description': 'Reality TV personality and businesswoman',
        'country': 'United States'
    },
    {
        'name': 'Lionel Messi',
        'follower_count': 149,
        'description': 'Footballer',
        'country': 'Argentina'
    },
    {
        'name': 'Beyoncé',
        'follower_count': 145,
        'description': 'Musician',
        'country': 'United States'
    },
    {
        'name': 'Neymar',
        'follower_count': 138,
        'description': 'Footballer',
        'country': 'Brasil'
    },
    {
        'name': 'National Geographic',
        'follower_count': 135,
        'description': 'Magazine',
        'country': 'United States'
    },
    {
        'name': 'Justin Bieber',
        'follower_count': 133,
        'description': 'Musician',
        'country': 'Canada'
    },
    {
        'name': 'Taylor Swift',
```

```
        'follower_count': 131,
        'description': 'Musician',
        'country': 'United States'
    },
    {
        'name': 'Kendall Jenner',
        'follower_count': 127,
        'description': 'Reality TV personality and Model',
        'country': 'United States'
    },
    {
        'name': 'Jennifer Lopez',
        'follower_count': 119,
        'description': 'Musician and actress',
        'country': 'United States'
    },
    {
        'name': 'Nicki Minaj',
        'follower_count': 113,
        'description': 'Musician',
        'country': 'Trinidad and Tobago'
    },
    {
        'name': 'Nike',
        'follower_count': 109,
        'description': 'Sportswear multinational',
        'country': 'United States'
    },
    {
        'name': 'Khloé Kardashian',
        'follower_count': 108,
        'description': 'Reality TV personality and businesswoman',
        'country': 'United States'
    },
    {
        'name': 'Miley Cyrus',
        'follower_count': 107,
        'description': 'Musician and actress',
        'country': 'United States'
    },
    {
        'name': 'Katy Perry',
        'follower_count': 94,
        'description': 'Musician',
        'country': 'United States'
    },
    {
        'name': 'Kourtney Kardashian',
        'follower_count': 90,
        'description': 'Reality TV personality',
        'country': 'United States'
    },
    {
        'name': 'Kevin Hart',
        'follower_count': 89,
        'description': 'Comedian and actor',
        'country': 'United States'
    },
    {
        'name': 'Ellen DeGeneres',
        'follower_count': 87,
```

```
        'description': 'Comedian',
        'country': 'United States'
    },
    {
        'name': 'Real Madrid CF',
        'follower_count': 86,
        'description': 'Football club',
        'country': 'Spain'
    },
    {
        'name': 'FC Barcelona',
        'follower_count': 85,
        'description': 'Football club',
        'country': 'Spain'
    },
    {
        'name': 'Rihanna',
        'follower_count': 81,
        'description': 'Musician and businesswoman',
        'country': 'Barbados'
    },
    {
        'name': 'Demi Lovato',
        'follower_count': 80,
        'description': 'Musician and actress',
        'country': 'United States'
    },
    {
        'name': "Victoria's Secret",
        'follower_count': 69,
        'description': 'Lingerie brand',
        'country': 'United States'
    },
    {
        'name': 'Zendaya',
        'follower_count': 68,
        'description': 'Actress and musician',
        'country': 'United States'
    },
    {
        'name': 'Shakira',
        'follower_count': 66,
        'description': 'Musician',
        'country': 'Colombia'
    },
    {
        'name': 'Drake',
        'follower_count': 65,
        'description': 'Musician',
        'country': 'Canada'
    },
    {
        'name': 'Chris Brown',
        'follower_count': 64,
        'description': 'Musician',
        'country': 'United States'
    },
    {
        'name': 'LeBron James',
        'follower_count': 63,
        'description': 'Basketball player',
```

```
        'country': 'United States'
    },
    {
        'name': 'Vin Diesel',
        'follower_count': 62,
        'description': 'Actor',
        'country': 'United States'
    },
    {
        'name': 'Cardi B',
        'follower_count': 67,
        'description': 'Musician',
        'country': 'United States'
    },
    {
        'name': 'David Beckham',
        'follower_count': 82,
        'description': 'Footballer',
        'country': 'United Kingdom'
    },
    {
        'name': 'Billie Eilish',
        'follower_count': 61,
        'description': 'Musician',
        'country': 'United States'
    },
    {
        'name': 'Justin Timberlake',
        'follower_count': 59,
        'description': 'Musician and actor',
        'country': 'United States'
    },
    {
        'name': 'UEFA Champions League',
        'follower_count': 58,
        'description': 'Club football competition',
        'country': 'Europe'
    },
    {
        'name': 'NASA',
        'follower_count': 56,
        'description': 'Space agency',
        'country': 'United States'
    },
    {
        'name': 'Emma Watson',
        'follower_count': 56,
        'description': 'Actress',
        'country': 'United Kingdom'
    },
    {
        'name': 'Shawn Mendes',
        'follower_count': 57,
        'description': 'Musician',
        'country': 'Canada'
    },
    {
        'name': 'Virat Kohli',
        'follower_count': 55,
        'description': 'Cricketer',
        'country': 'India'
```

```
    },
    {
        'name': 'Gigi Hadid',
        'follower_count': 54,
        'description': 'Model',
        'country': 'United States'
    },
    {
        'name': 'Priyanka Chopra Jonas',
        'follower_count': 53,
        'description': 'Actress and musician',
        'country': 'India'
    },
    {
        'name': '9GAG',
        'follower_count': 52,
        'description': 'Social media platform',
        'country': 'China'
    },
    {
        'name': 'Ronaldinho',
        'follower_count': 51,
        'description': 'Footballer',
        'country': 'Brasil'
    },
    {
        'name': 'Maluma',
        'follower_count': 50,
        'description': 'Musician',
        'country': 'Colombia'
    },
    {
        'name': 'Camila Cabello',
        'follower_count': 49,
        'description': 'Musician',
        'country': 'Cuba'
    },
    {
        'name': 'NBA',
        'follower_count': 47,
        'description': 'Club Basketball Competition',
        'country': 'United States'
    }
]
```

main.py from game_data import data import random from art import logo, vs from replit import clear def get_random_account(): """Get data from random account""" return random.choice(data) def format_data(account): """Format account into printable format: name, description and country""" name = account["name"] description = account["description"] country = account["country"] # print(f'{name}: {account["follower_count"]}') return f"{name}, a {description}, from {country}" def check_answer(guess, a_followers, b_followers): """Checks followers against user's guess and returns True if they got it right. Or False if they got it wrong.""" if a_followers > b_followers: return guess == "a" else: return guess == "b" def game(): print(logo) score = 0 game_should_continue = True account_a = get_random_account() account_b = get_random_account() while game_should_continue: account_a = account_b account_b = get_random_account() while account_a == account_b: account_b = get_random_account() print(f"Compare A: {format_data(account_a)}.") print(vs) print(f"Against B: {format_data(account_b)}.") guess = input("Who has more followers? Type 'A' or 'B': ").lower() a_follower_count = account_a["follower_count"] b_follower_count = account_b["follower_count"] is_correct = check_answer(guess, a_follower_count, b_follower_count) clear() print(logo) if is_correct: score += 1 print(f"You're right! Current score: {score}.") else: game_should_continue = False print(f"Sorry, that's wrong. Final score: {score}") game()

DAY-15

Project-->Coffee Machine Program

In [ ]:

```
MENU = {
    "espresso": {
        "ingredients": {
            "water": 50,
            "coffee": 18,
        },
        "cost": 1.5,
    },
    "latte": {
        "ingredients": {
            "water": 200,
            "milk": 150,
            "coffee": 24,
        },
        "cost": 2.5,
    },
    "cappuccino": {
        "ingredients": {
            "water": 250,
            "milk": 100,
            "coffee": 24,
        },
        "cost": 3.0,
    }
}

profit = 0
resources = {
    "water": 300,
    "milk": 200,
    "coffee": 100,
}


def is_resource_sufficient(order_ingredients):
    """Returns True when order can be made, False if ingredients are insufficient."""
    for item in order_ingredients:
        if order_ingredients[item] > resources[item]:
            print(f"Sorry there is not enough {item}.")
            return False
    return True


def process_coins():
    """Returns the total calculated from coins inserted."""
    print("Please insert coins.")
    total = int(input("how many quarters?: ")) * 0.25
    total += int(input("how many dimes?: ")) * 0.1
    total += int(input("how many nickles?: ")) * 0.05
    total += int(input("how many pennies?: ")) * 0.01
    return total


def is_transaction_successful(money_received, drink_cost):
```

```
    """Return True when the payment is accepted, or False if money is insufficient."""
    if money_received >= drink_cost:
        change = round(money_received - drink_cost, 2)
        print(f"Here is ${change} in change.")
        global profit
        profit += drink_cost
        return True
    else:
        print("Sorry that's not enough money. Money refunded.")
        return False


def make_coffee(drink_name, order_ingredients):
    """Deduct the required ingredients from the resources."""
    for item in order_ingredients:
        resources[item] -= order_ingredients[item]
    print(f"Here is your {drink_name} ☕. Enjoy!")


is_on = True

while is_on:
    choice = input("What would you like? (espresso/latte/cappuccino): ")
    if choice == "off":
        is_on = False
    elif choice == "report":
        print(f"Water: {resources['water']}ml")
        print(f"Milk: {resources['milk']}ml")
        print(f"Coffee: {resources['coffee']}g")
        print(f"Money: ${profit}")
    else:
        drink = MENU[choice]
        if is_resource_sufficient(drink["ingredients"]):
            payment = process_coins()
            if is_transaction_successful(payment, drink["cost"]):
                make_coffee(choice, drink["ingredients"])
```

```
Water: 300ml
Milk: 200ml
Coffee: 100g
Money: $0

Please insert coins.

Here is $19.31 in change.
Here is your espresso ☕. Enjoy!

Please insert coins.

Sorry that's not enough money. Money refunded.

Water: 250ml
Milk: 200ml
Coffee: 82g
Money: $1.5

Please insert coins.

Here is $33.9 in change.
Here is your cappuccino ☕. Enjoy!

Water: 0ml
Milk: 100ml
```

```
Coffee: 58g
Money: $4.5
```

DAY-16

The above style of programming is called Procedural Programming

In [1]:
```
# In order to break the task into pieces we use oops concept
```

In [2]:
```
# In a class we have attributes and objects
```

In [3]:
```
#Attributes in a class is like a variables
#Objects in a class is like a function
```

In [4]:
```
#Eg:Consider there is a waiter() which is considered as class
#Here,
#Attributes:
#           is_holding_plate=True
#           table_alloted=[3,4,5]

#Objects:
#        def take_order(table,order):
#                    #takes order to chef
#        def take_payments(amount):
#                    #adding money to restaurent
```

In [6]:
```
#Here a class is considered as a blueprint
#We can take as many waiters we want(It is like xerox copy)
# Betty=waiter()
# Arjun=waiter()
#Betty and arjun are the xerox copy of the class which are objects{like a function}
#Waiter is the original copy
```

In [7]:
```
#Eg:
#Car BLueprint is a class named [CarBlueprint()]
#Xerox of Blueprint is a object
 # car1=CarBlueprint()    -->Copy-1 -->Object-1
 # car2=CarBlueprint()    -->Copy-2 -->Object-2
 # car3=CarBlueprint()    -->Copy-3 -->Object-3
```

In [8]:
```
#Car attributes of the class:
 #speed=0
 #fuel=32

#Getting the value from the class:
 #Object.attribute
```

```
 #car1.speed
 #car2.fuel
```

In [11]:

```
#Objects methods are the sub function Eg:Car1 is a function and in that move() is anoth
#Car Object Methods of the class:
 #def move():
    #speed=60
 #def stop():
    #speed=0

#Getting the value from the class:
 #Object.method
 #car1.move()
 #car2.stop()
```

In [1]:

```
# Example of a inbuilt class turtle
from turtle import Turtle,Screen

#Creating an object with class
timmy=Turtle()
print(timmy)

#methods in the object with class
timmy.shape("turtle")
timmy.color("red")
timmy.forward(100)
timmy.left(55)
timmy.forward(110)
timmy.left(55)
timmy.forward(100)
timmy.left(55)
timmy.forward(100)
timmy.left(55)
timmy.forward(110)
timmy.left(55)
timmy.forward(110)

#Creating an object with the class
my_screen=Screen()

#getting attribute value from the class
print(my_screen.canvheight)
my_screen.exitonclick()
```

```
<turtle.Turtle object at 0x0000029297507880>
300
```

How to Add Python Packages and use PyPi

PyPi is a online resource which stores all the packages

we use pip install to download the module

In [4]:

```
pip install prettytable
```

```
Requirement already satisfied: prettytable in c:\users\srena\anaconda3\lib\site-packages
(2.1.0)
```

```
Requirement already satisfied: wcwidth in c:\users\srena\anaconda3\lib\site-packages (fr
om prettytable) (0.2.5)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
from prettytable import PrettyTable
```

In [6]:
```
#Creating a object with class
table= PrettyTable()
```

In [7]:
```
print(table)
```

```
++
||
++
++
```

In [8]:
```
#Adding a column in the table
#Object with methods
table.add_column("City name",["Adelaide","Brisbane","Darwin","Hobart","Sydney","Melbour
```

In [9]:
```
print(table)
```

```
+-----------+
| City name |
+-----------+
|  Adelaide |
|  Brisbane |
|   Darwin  |
|   Hobart  |
|   Sydney  |
| Melbourne |
|   Perth   |
+-----------+
```

In [10]:
```
table.add_column("Area", [1295, 5905, 112, 1357, 2058, 1566, 5386])
table.add_column("Population", [1158259, 1857594, 120900, 205556, 4336374, 3806092, 155
table.add_column("Annual Rainfall",[600.5, 1146.4, 1714.7, 619.5, 1214.8, 646.9, 869.4]
```

In [12]:
```
print(table)
```

```
+-----------+------+------------+-----------------+
| City name | Area | Population | Annual Rainfall |
+-----------+------+------------+-----------------+
|  Adelaide | 1295 |  1158259   |      600.5      |
|  Brisbane | 5905 |  1857594   |      1146.4     |
|   Darwin  | 112  |   120900   |      1714.7     |
|   Hobart  | 1357 |   205556   |      619.5      |
|   Sydney  | 2058 |  4336374   |      1214.8     |
| Melbourne | 1566 |  3806092   |      646.9      |
|   Perth   | 5386 |  1554769   |      869.4      |
+-----------+------+------------+-----------------+
```

In [14]:
```
#Objects with attributes
table.align="l"
```

In [15]:
```
print(table)
```

```
+-----------+------+------------+-----------------+
| City name | Area | Population | Annual Rainfall |
+-----------+------+------------+-----------------+
| Adelaide  | 1295 | 1158259    | 600.5           |
| Brisbane  | 5905 | 1857594    | 1146.4          |
| Darwin    | 112  | 120900     | 1714.7          |
| Hobart    | 1357 | 205556     | 619.5           |
| Sydney    | 2058 | 4336374    | 1214.8          |
| Melbourne | 1566 | 3806092    | 646.9           |
| Perth     | 5386 | 1554769    | 869.4           |
+-----------+------+------------+-----------------+
```

Coffee-Machine with OOPS Concept

In [16]:
```
#coffee_maker.py

class CoffeeMaker:
    """Models the machine that makes the coffee"""
    def __init__(self):
        self.resources = {
            "water": 300,
            "milk": 200,
            "coffee": 100,
        }

    def report(self):
        """Prints a report of all resources."""
        print(f"Water: {self.resources['water']}ml")
        print(f"Milk: {self.resources['milk']}ml")
        print(f"Coffee: {self.resources['coffee']}g")

    def is_resource_sufficient(self, drink):
        """Returns True when order can be made, False if ingredients are insufficient."
        can_make = True
        for item in drink.ingredients:
            if drink.ingredients[item] > self.resources[item]:
                print(f"Sorry there is not enough {item}.")
                can_make = False
        return can_make

    def make_coffee(self, order):
        """Deducts the required ingredients from the resources."""
        for item in order.ingredients:
            self.resources[item] -= order.ingredients[item]
        print(f"Here is your {order.name} ☕. Enjoy!")
```

In [17]:
```
# menu.py

class MenuItem:
    """Models each Menu Item."""
    def __init__(self, name, water, milk, coffee, cost):
        self.name = name
        self.cost = cost
        self.ingredients = {
            "water": water,
```

```
            "milk": milk,
            "coffee": coffee
        }


class Menu:
    """Models the Menu with drinks."""
    def __init__(self):
        self.menu = [
            MenuItem(name="latte", water=200, milk=150, coffee=24, cost=2.5),
            MenuItem(name="espresso", water=50, milk=0, coffee=18, cost=1.5),
            MenuItem(name="cappuccino", water=250, milk=50, coffee=24, cost=3),
        ]

    def get_items(self):
        """Returns all the names of the available menu items"""
        options = ""
        for item in self.menu:
            options += f"{item.name}/"
        return options

    def find_drink(self, order_name):
        """Searches the menu for a particular drink by name. Returns that item if it ex
        for item in self.menu:
            if item.name == order_name:
                return item
        print("Sorry that item is not available.")
```

In [18]:

```
# money_machine.py

class MoneyMachine:

    CURRENCY = "$"

    COIN_VALUES = {
        "quarters": 0.25,
        "dimes": 0.10,
        "nickles": 0.05,
        "pennies": 0.01
    }

    def __init__(self):
        self.profit = 0
        self.money_received = 0

    def report(self):
        """Prints the current profit"""
        print(f"Money: {self.CURRENCY}{self.profit}")

    def process_coins(self):
        """Returns the total calculated from coins inserted."""
        print("Please insert coins.")
        for coin in self.COIN_VALUES:
            self.money_received += int(input(f"How many {coin}?: ")) * self.COIN_VALUES
        return self.money_received

    def make_payment(self, cost):
        """Returns True when payment is accepted, or False if insufficient."""
        self.process_coins()
```

```
        if self.money_received >= cost:
            change = round(self.money_received - cost, 2)
            print(f"Here is {self.CURRENCY}{change} in change.")
            self.profit += cost
            self.money_received = 0
            return True
        else:
            print("Sorry that's not enough money. Money refunded.")
            self.money_received = 0
            return False
```

In [20]:

```
# main.py

#from menu import Menu
#from coffee_maker import CoffeeMaker
#from money_machine import MoneyMachine

money_machine=MoneyMachine()
coffee_maker=CoffeeMaker()
menu = Menu()
while True:
    choice=input("What would you like? (latte/espresso/cappuccino/):")

    if choice == "report":
        coffee_maker.report()
    elif choice=="off":
        print("Turning Off....")
        break
    else:
        drink = menu.find_drink(choice)
        if coffee_maker.is_resource_sufficient(drink) is True:
            if money_machine.make_payment(drink.cost):
                coffee_maker.make_coffee(drink)
```

```
Please insert coins.

Here is $29.31 in change.
Here is your latte ☕. Enjoy!

Sorry there is not enough water.

Please insert coins.

Here is $19.21 in change.
Here is your espresso ☕. Enjoy!

Water: 50ml
Milk: 50ml
Coffee: 58g

Turning Off....
```

DAY-17

In [3]:

```
# Creating a simple class
```

```
#Here we are creating a simple class and creating an object user1 and user2
#In the object user1 and user2 we create an attribute of id name and assigning values
class user:
    pass

user1=user()
user1.id="001"
user1.name="Srenath Kumar"

user2=user()
user2.id="002"
user2.name="Arjun"


print(user1.id)
```

```
001
```

In [13]:

```
#In the above code it is very difficult to add each details of 1000 members
#In order to solve this problem we use a constructor {self constructor} called init

#In this init the self is considered as the object we are assigned for the class

class User:
    def __init__(self,id,name):
        self.id=id
        self.name=name
        self.init_marks= 0     #-->Creating a default value in the init


#Remeber to add parameters to the user as given below
user1=User("001","Srenath")
user2=User("002","Kumar")

#On assigning the User class to object user1, the self is considered as user1 on that s
#Eg :self = user1 when user1 is initialized
# self=user2 when user2 is initialized

#Getting the values from the class assigned
print(user1.id)
print(user2.name)
#Displaying the default value from the User class
print(user1.init_marks)
```

```
001
Kumar
0
```

In [16]:

```
#Creting a class like instagram
#Creating a class user with attributes of id name followers and following
class User:
    def __init__(self,id,name):
        self.id=id
        self.name=name
        self.followers=0
        self.following=0

    #Crating a object method in a class
    #This method gets the object and user for increasing the followers and following by
    def follow(self,user):
```

```
        user.followers += 1
        self.following += 1

user1=User("001","Jack")
user2=User("002","Vin")

#This below statement is user1 is following user 2 so
# user2 followers is 1
# user1 following is 1
user1.follow(user2)

print(user1.name)

print(user1.followers)
print(user1.following)
print(user2.followers)
print(user2.following)
```

```
Jack
0
1
1
0
```

PROJECT --> Creating a Quiz-Game

In [2]:

```
# data.py

question_data = [
    {
        "category": "Science: Computers",
        "type": "boolean",
        "difficulty": "medium",
        "question": "The HTML5 standard was published in 2014.",
        "correct_answer": "True",
        "incorrect_answers": [
            "False"
        ]
    },
    {
        "category": "Science: Computers",
        "type": "boolean",
        "difficulty": "medium",
        "question": "The first computer bug was formed by faulty wires.",
        "correct_answer": "False",
        "incorrect_answers": [
            "True"
        ]
    },
    {
        "category": "Science: Computers",
        "type": "boolean",
        "difficulty": "medium",
        "question": "FLAC stands for 'Free Lossless Audio Condenser'.",
        "correct_answer": "False",
        "incorrect_answers": [
            "True"
        ]
    },
    {
```

```
        "category": "Science: Computers",
        "type": "boolean",
        "difficulty": "medium",
        "question": "All program codes have to be compiled into an executable file in o
        "correct_answer": "False",
        "incorrect_answers": [
            "True"
        ]
    },
    {
        "category": "Science: Computers",
        "type": "boolean",
        "difficulty": "easy",
        "question": "Linus Torvalds created Linux and Git.",
        "correct_answer": "True",
        "incorrect_answers": [
            "False"
        ]
    },
    {
        "category": "Science: Computers",
        "type": "boolean",
        "difficulty": "easy",
        "question": "The programming language 'Python' is based off a modified version
        "correct_answer": "False",
        "incorrect_answers": [
            "True"
        ]
    },
    {
        "category": "Science: Computers",
        "type": "boolean",
        "difficulty": "medium",
        "question": "AMD created the first consumer 64-bit processor.",
        "correct_answer": "True",
        "incorrect_answers": [
            "False"
        ]
    },
    {
        "category": "Science: Computers",
        "type": "boolean",
        "difficulty": "easy",
        "question": "'HTML' stands for Hypertext Markup Language.",
        "correct_answer": "True",
        "incorrect_answers": [
            "False"
        ]
    },
    {
        "category": "Science: Computers",
        "type": "boolean",
        "difficulty": "easy",
        "question": "In most programming languages, the operator ++ is equivalent to th
        "correct_answer": "True",
        "incorrect_answers": [
            "False"
        ]
    },
    {
        "category": "Science: Computers",
```

```
        "type": "boolean",
        "difficulty": "hard",
        "question": "The IBM PC used an Intel 8008 microprocessor clocked at 4.77 MHz a
        "correct_answer": "False",
        "incorrect_answers": [
            "True"
        ]
    }
]
```

In [3]:

```
#question_model.py

class Question:

    def __init__(self, q_text, q_answer):
        self.text = q_text
        self.answer = q_answer
```

In [4]:

```
# quiz_brain.py

class QuizBrain:
    #Creating a init for getting the details
    def __init__(self, q_list):
        self.question_number = 0
        self.score = 0
        self.question_list = q_list


    def still_has_questions(self):
        #When the question number in text is less than in the data list
        return self.question_number < len(self.question_list)

    def next_question(self):
        current_question = self.question_list[self.question_number]
        self.question_number += 1
        user_answer = input(f"Q.{self.question_number}: {current_question.text} (True/F
        self.check_answer(user_answer, current_question.answer)

    def check_answer(self, user_answer, correct_answer):
        if user_answer.lower() == correct_answer.lower():
            self.score += 1
            print("You got it right!")
        else:
            print("That's wrong.")
        print(f"The correct answer was: {correct_answer}.")
        print(f"Your current score is: {self.score}/{self.question_number}")
        print("\n")
```

In [5]:

```
#main.py

#from question_model import Question
#from data import question_data
#from quiz_brain import QuizBrain

question_bank = []
for question in question_data:
    question_text = question["question"]
```

```
    question_answer = question["correct_answer"]
    new_question = Question(question_text, question_answer)
    question_bank.append(new_question)

quiz = QuizBrain(question_bank)

while quiz.still_has_questions():
    quiz.next_question()

print("You've completed the quiz")
print(f"Your final score was: {quiz.score}/{quiz.question_number}")
```

```
You got it right!
The correct answer was: True.
Your current score is: 1/1


That's wrong.
The correct answer was: False.
Your current score is: 1/2


You got it right!
The correct answer was: False.
Your current score is: 2/3


That's wrong.
The correct answer was: False.
Your current score is: 2/4


That's wrong.
The correct answer was: True.
Your current score is: 2/5


That's wrong.
The correct answer was: False.
Your current score is: 2/6


That's wrong.
The correct answer was: True.
Your current score is: 2/7


That's wrong.
The correct answer was: True.
Your current score is: 2/8


You got it right!
The correct answer was: True.
Your current score is: 3/9
```

```
You got it right!
The correct answer was: False.
Your current score is: 4/10


You've completed the quiz
Your final score was: 4/10
```

DAY-18

Turtle Challenge 1 - Draw a Square

In [2]:
```
import turtle as t

timmy_the_turtle = t.Turtle()

for _ in range(4):
    timmy_the_turtle.forward(100)
    timmy_the_turtle.left(90)
```

In [2]:
```
#Method two of importing

from turtle import *
#This helps to get all the functions and classes from the turtle module

forward(100)
```

In [1]:
```
# alias variable for the module

import turtle as t
#Where turtle is taken as t
```

Turtle Challenge 2 - Draw a Dashed Line

In [1]:
```
import turtle as t
from turtle import Screen

tim=t.Turtle()
#This _ below in the for loop is the line to draw the board
for _ in range(15):
    tim.forward(10)
    #penup is used for making the foward line invisible
    tim.penup()
    tim.forward(10)
    tim.pendown()
screen=Screen()
screen.exitonclick()
```

Turtle Challenge 3 - Drawing Different Shapes

In [19]:
```
import turtle as t
from turtle import Screen
import random as r

#triangle_angle=360/3 (3 sides) ......


tim45=t.Turtle()
colours=["blue","green","red","yellow","coral"]
def drawing(sides):
    for _ in range(sides):

        angle=360/sides
        tim45.forward(100)
        tim45.right(angle)

for i in range(3,11):
    tim45.color(r.choice(colours))
    drawing(i)



screen=Screen()
screen.exitonclick()
```

Turtle Challenge 4 - Generate a Random Walk

In [ ]:
```
import turtle as t
from turtle import Screen
import random as r

timmy95324=t.Turtle()

turnings=[0,98,100,270]
colours = ["blue","green","red","yellow","coral"]

for _ in range(0,100):
    timmy95324.pensize(10)
    timmy95324.color(r.choice(colours))
    timmy95324.forward(100)
    #set heading is used to turn the turtle at given angle of direction
    timmy95324.setheading(r.choice(turnings))

screen=Screen()
screen.exitonclick()
```

Turtle Challange 4.1-Generate Random RGB Colours

In [ ]:
```
import turtle as t
from turtle import Screen
import random as r1

timmy65=t.Turtle()
t.colormode(255)
#Keeping the mode of the rgb at 255
```

```
turnings=[0,98,100,270]

def colors():
    #This randint selects random numbers from 0,255
    r=r1.randint(0,255)
    g=r1.randint(0,255)
    b=r1.randint(0,255)
    #Combining r,g,b to rgb with tuples
    random_color=(r,g,b)
    #And returning the color
    return random_color

for _ in range(0,100):
    timmy65.pensize(10)
    timmy65.color(colors())
    timmy65.forward(100)
    timmy65.setheading(r.choice(turnings))

screen=Screen()
screen.exitonclick()
```

Turtle Challenge 5 - Draw a Spirograph

In [29]:

```
import turtle as t
from turtle import Screen

#Remember to add colormode in the class if you are using rgb combinations
t.colormode(255)

def colors():
    #This randint selects random numbers from 0,255
    r=r1.randint(0,255)
    g=r1.randint(0,255)
    b=r1.randint(0,255)
    #Combining r,g,b to rgb with tuples
    random_color=(r,g,b)
    #And returning the color
    return random_color

#Crating an object with class turtle
turtle77=t.Turtle()
#Increasing the speed of drawing
turtle77.speed("fastest")
def spirograph(size):
    for _ in range(75):
        turtle77.color(colors())
        #Adding a circle
        turtle77.circle(100)
        #Moving the pointer to left
        turtle77.left(size)

spirograph(5)

screen=Screen()
screen.exitonclick()
```

Project-1 --> The Hirst Painting Project Part 1 - How to

Extract RGB Values from Images

In [49]:
```
import colorgram

rgb_colors=[]
# Extract 6 colors from an image.
#This extract helps to convert the image into differenct colors available 30 is the fir
colors = colorgram.extract('download.jpg', 30)
#Converting rgb data type to a tuple of rgb and added to a rgb_colors list
for color in colors:
    r = color.rgb.r
    g = color.rgb.g
    b = color.rgb.b
    c=(r,g,b)
    rgb_colors.append(c)


print(rgb_colors)
```

```
[(248, 243, 228), (251, 237, 247), (233, 251, 242), (233, 241, 249), (208, 157, 107), (1
97, 142, 168), (231, 215, 92), (37, 109, 159), (125, 166, 196), (119, 185, 144), (38, 13
1, 78), (150, 79, 57), (47, 175, 110), (132, 70, 91), (180, 176, 29), (54, 19, 32), (23
1, 166, 197), (45, 24, 16), (227, 82, 56), (199, 83, 106), (135, 29, 50), (233, 222, 4),
(10, 101, 55), (177, 184, 223), (38, 166, 194), (23, 24, 35), (144, 218, 175), (141, 30,
20), (74, 123, 204), (238, 168, 156)]
```

In [ ]:
```
color_list=[(248, 243, 228), (251, 237, 247), (233, 251, 242), (233, 241, 249), (208, 1
```

Project-1 --> The Hirst Painting Project Part 2 - Drawing the Dots

In [54]:
```
import turtle as turtle_module
import random

#Setting the color mode
turtle_module.colormode(255)
tim = turtle_module.Turtle()
tim.speed("fastest")
#penup is used to remove forward line
tim.penup()
#Hinding the icon
tim.hideturtle()
color_list=[(248, 243, 228), (251, 237, 247), (233, 251, 242), (233, 241, 249), (208, 1


tim.setheading(225)
tim.forward(300)
tim.setheading(0)
number_of_dots = 100

for dot_count in range(1, number_of_dots + 1):
    tim.dot(20, random.choice(color_list))
    tim.forward(50)

    if dot_count % 10 == 0:
        tim.setheading(90)
```

```
        tim.forward(50)
        tim.setheading(180)
        tim.forward(500)
        tim.setheading(0)



screen = turtle_module.Screen()
screen.exitonclick()
```

DAY-19

Python Higher Order Functions & Event Listeners

In [6]:

```
#Here we use a function called listen which gets theh input of the keyboard and does ac
#When the space button is pressed the turtle moves 10 steps on every click
import turtle as t
from turtle import Screen

def forward():
    timis1.forward(10)

timis1=t.Turtle()
screen=Screen()
screen.listen()
#Adding function into another function
#Remeber that on adding a function into a function we should not add paranthesis
screen.onkey(key="space",fun=forward)
screen.exitonclick()
```

Challenge: Make an Etch-A-Sketch App

In [ ]:

```
import turtle as t
from turtle import Screen
#Function forward to move the turtle forward
def forward():
    timess.forward(10)
#function reverse to move the turtle reverse
def reverse():
    timess.backward(10)
#function clockwise to move the turtle right size
def clockwise():
    timess.right(25)
#function counter_clockwise to move the turtle left size
def couter_clockwise():
    timess.left(25)
#function clear to clear the screen
def clear():
    timess.clear()

timess=t.Turtle()
screen=Screen()
#object method listen to get the inputs of the keyboard
screen.listen()
```

```
screen.onkey(key="w",fun=forward)
screen.onkey(key="s",fun=reverse)
screen.onkey(key="d",fun=clockwise)
screen.onkey(key="a",fun=clockwise)
screen.onkey(key="c",fun=clear)

screen.exitonclick()
```

Building the turtle race

In [9]:

```
from turtle import Turtle, Screen
import random

is_race_on = False
screen = Screen()
screen.setup(width=500, height=400)
user_bet = screen.textinput(title="Make your bet", prompt="Which turtle will win the ra
colors = ["red", "orange", "yellow", "green", "blue", "purple"]
y_positions = [-70, -40, -10, 20, 50, 80]
all_turtles = []

#Create 6 turtles
for turtle_index in range(0, 6):
    new_turtle = Turtle(shape="turtle")
    new_turtle.penup()
    new_turtle.color(colors[turtle_index])
    new_turtle.goto(x=-230, y=y_positions[turtle_index])
    all_turtles.append(new_turtle)

if user_bet:
    is_race_on = True

while is_race_on:
    for turtle in all_turtles:
        #230 is 250 - half the width of the turtle.
        if turtle.xcor() > 230:
            is_race_on = False
            winning_color = turtle.pencolor()
            if winning_color == user_bet:
                print(f"You've won! The {winning_color} turtle is the winner!")
            else:
                print(f"You've lost! The {winning_color} turtle is the winner!")

        #Make each turtle move a random amount.
        rand_distance = random.randint(0, 10)
        turtle.forward(rand_distance)

screen.exitonclick()
```

```
You've lost! The orange turtle is the winner!
You've lost! The green turtle is the winner!
```

DAY-20

Build the Snake Game Part 1: Animation & Coordinates

In [4]:
```
#Step-1: Creating a Black Background with 600x600 screen

import turtle as t
from turtle import Screen

times=t.Turtle()
screen=Screen()
screen.setup(width=600,height=600)
screen.bgcolor("black")
screen.exitonclick()
```

In [23]:
```
# Step-2 Creating three turtles with size of 20x20 squares like and changing the color

import turtle as t
from turtle import Screen

#Positions alotted for each snake tails
starting_position=[(0,0),(-20,0),(-40,0)]
#looping starting_poisition at each iterations
for i in starting_position:
    #Creating a turtle class
    times1=t.Turtle()
    times1.shape("square")
    times1.color("white")
    times1.goto(i)




screen=Screen()
screen.setup(width=600,height=600)
screen.bgcolor("black")
screen.exitonclick()
```

In [ ]:
```
# Step-3 Creting a class and moving the snake to the given direction through keyboard
from turtle import Turtle
from turtle import Screen
import time

#Giving the start game positions for the three squares
STARTING_POSITIONS = [(0, 0), (-20, 0), (-40, 0)]
MOVE_DISTANCE = 20
#Angle for turning
UP = 90
DOWN = 270
LEFT = 180
RIGHT = 0


class Snake:

    def __init__(self):
        self.segments = []
        self.create_snake()
        self.head = self.segments[0]

    def create_snake(self):
        for position in STARTING_POSITIONS:
```

```
            #Crreating a object
            new_segment = Turtle("square")
            new_segment.color("white")
            #Removing the line
            new_segment.penup()
            #Going the the given position
            new_segment.goto(position)
            #Adding new position of the first square(head) to the segment which helps t
            self.segments.append(new_segment)

    def move(self):
        #In this code the concept is first the second square is moved to 1st cube posit
        for seg_num in range(len(self.segments) - 1, 0, -1):
            new_x = self.segments[seg_num - 1].xcor()
            new_y = self.segments[seg_num - 1].ycor()
            self.segments[seg_num].goto(new_x, new_y)
        self.head.forward(MOVE_DISTANCE)

    def up(self):
        #This if condition helps to restrict the to go down when the snake goes up
        if self.head.heading() != DOWN:
            self.head.setheading(UP)

    def down(self):
        if self.head.heading() != UP:
            self.head.setheading(DOWN)

    def left(self):
        if self.head.heading() != RIGHT:
            self.head.setheading(LEFT)

    def right(self):
        if self.head.heading() != LEFT:
            self.head.setheading(RIGHT)



screen = Screen()
screen.setup(width=600, height=600)
screen.bgcolor("black")
screen.title("My Snake Game")
screen.tracer(0)

snake = Snake()


screen.listen()
screen.onkey(snake.up, "Up")
screen.onkey(snake.down, "Down")
screen.onkey(snake.left, "Left")
screen.onkey(snake.right, "Right")

game_is_on = True
while game_is_on:
    screen.update()
    time.sleep(0.1)

    snake.move()


screen.exitonclick()
```

DAY-21

Inheritence

In [ ]:
```
--> Meaning of Inheritence is a class which inherits another class is inheritence
EG:Lets have we have a class animal,fish ,lion
    The common things in all the animals are eyes and breathing
    So the common things will be in the animal class and this details will be inherited
```

In [16]:
```
#Remember to give self in all the methods inside the class

class Animals():
    def __init__(self):
        self.eyes=2

    def breathe(self):
        print("Inhales and Exhales")

class Fish(Animals):          #The fish class gets inherited to the animal class
    def __init__(self):
        super().__init__()

    def breathe(self):
        super().breathe()
        print("Breathing wiht Gills")

    def temperature(self):
        print("45' ")
```

In [18]:
```
fish=Fish()
print(fish.eyes)
fish.breathe()
fish.temperature()
```

```
2
Inhales and Exhales
Breathing wiht Gills
45'
```

SNAKE-GAME FINAL CODE

In [1]:
```
from turtle import Turtle
STARTING_POSITIONS = [(0, 0), (-20, 0), (-40, 0)]
MOVE_DISTANCE = 20
UP = 90
DOWN = 270
LEFT = 180
RIGHT = 0


class Snake:
```

```
    def __init__(self):
        self.segments = []
        self.create_snake()
        self.head = self.segments[0]

    #Creating a snake wih first three squares
    def create_snake(self):
        for position in STARTING_POSITIONS:
            self.add_segment(position)
    #Adding the squares to the snake
    def add_segment(self, position):
        new_segment = Turtle("square")
        new_segment.color("white")
        new_segment.penup()
        new_segment.goto(position)
        self.segments.append(new_segment)

    def extend(self):
        self.add_segment(self.segments[-1].position())

    def move(self):
        for seg_num in range(len(self.segments) - 1, 0, -1):
            new_x = self.segments[seg_num - 1].xcor()
            new_y = self.segments[seg_num - 1].ycor()
            self.segments[seg_num].goto(new_x, new_y)
        self.head.forward(MOVE_DISTANCE)

    def up(self):
        if self.head.heading() != DOWN:
            self.head.setheading(UP)

    def down(self):
        if self.head.heading() != UP:
            self.head.setheading(DOWN)

    def left(self):
        if self.head.heading() != RIGHT:
            self.head.setheading(LEFT)

    def right(self):
        if self.head.heading() != LEFT:
            self.head.setheading(RIGHT)
```

In [2]:

```
from turtle import Turtle
import random


class Food(Turtle):

    def __init__(self):
        super().__init__()
        self.shape("circle")
        self.penup()
        self.shapesize(stretch_len=0.5, stretch_wid=0.5)
        self.color("blue")
        self.speed("fastest")
        self.refresh()
```

```
    def refresh(self):
        random_x = random.randint(-280, 280)
        random_y = random.randint(-280, 280)
        self.goto(random_x, random_y)
```

In [3]:

```
from turtle import Turtle
ALIGNMENT = "center"
FONT = ("Courier", 24, "normal")


class Scoreboard(Turtle):

    def __init__(self):
        super().__init__()
        self.score = 0
        self.color("white")
        self.penup()
        self.goto(0, 270)
        self.hideturtle()
        self.update_scoreboard()

    def update_scoreboard(self):
        self.write(f"Score: {self.score}", align=ALIGNMENT, font=FONT)

    def game_over(self):
        self.goto(0, 0)
        self.write("GAME OVER", align=ALIGNMENT, font=FONT)

    def increase_score(self):
        self.score += 1
        self.clear()
        self.update_scoreboard()
```

In [4]:

```
from turtle import Screen
#from snake import Snake
#from food import Food
#from scoreboard import Scoreboard
import time

screen = Screen()
screen.setup(width=600, height=600)
screen.bgcolor("black")
screen.title("My Snake Game")
screen.tracer(0)

snake = Snake()
food = Food()
scoreboard = Scoreboard()

screen.listen()
screen.onkey(snake.up, "Up")
screen.onkey(snake.down, "Down")
screen.onkey(snake.left, "Left")
screen.onkey(snake.right, "Right")

game_is_on = True
while game_is_on:
    screen.update()
```

```
    time.sleep(0.1)
    snake.move()

    #Detect collision with food.
    if snake.head.distance(food) < 15:
        food.refresh()
        snake.extend()
        scoreboard.increase_score()

    #Detect collision with wall.
    if snake.head.xcor() > 280 or snake.head.xcor() < -280 or snake.head.ycor() > 280 o
        game_is_on = False
        scoreboard.game_over()

    #Detect collision with tail.
    for segment in snake.segments:
        if segment == snake.head:
            pass
        elif snake.head.distance(segment) < 10:
            game_is_on = False
            scoreboard.game_over()




screen.exitonclick()
```

DAY-22

Creating a Ping-Pong Game

In [ ]:

```
#Step:1 Setting up a new background with 600x800 screen
import turtle as t
from turtle import Screen

pong=t.Turtle()
screen=Screen()
screen.bgcolor("black")
screen.setup(height=600,width=800)
screen.exitonclick()
```

In [ ]:

```
#Step:2 Creting a class paddle and creating two paddles with the class that get respons

import turtle as t
from turtle import Screen,Turtle


class paddle(Turtle):
    def __init__(self,position):
        super().__init__()
        self.color("white")
        self.shape("square")
        self.turtlesize(stretch_wid=5,stretch_len=1)
        self.penup()
```

```
        self.goto(position)

    def go_up(self):
        new_y=self.ycor()+20
        self.goto(self.xcor(),new_y)

    def go_down(self):
        new_y=self.ycor()-20
        self.goto(self.xcor(),new_y)


r_paddle=paddle((350,0))
l_paddle=paddle((-350,0))

screen=Screen()
screen.bgcolor("black")
screen.setup(height=600,width=800)


screen.tracer(0)
screen.listen()
screen.onkey(r_paddle.go_up,"Up")
screen.onkey(r_paddle.go_down,"Down")

game_is_on=True
while game_is_on:
    screen.update()



screen.exitonclick()
```

In [ ]:

```
# Step:3 Creating a ball and make it move


import turtle as t
from turtle import Screen,Turtle
from time import sleep


class paddle(Turtle):
    def __init__(self,position):
        super().__init__()
        self.color("white")
        self.shape("square")
        self.turtlesize(stretch_wid=5,stretch_len=1)
        self.penup()
        self.goto(position)

    def go_up(self):
        new_y=self.ycor()+20
        self.goto(self.xcor(),new_y)

    def go_down(self):
        new_y=self.ycor()-20
        self.goto(self.xcor(),new_y)

class ball(Turtle):
    def __init__(self):
```

```
        super().__init__()
        self.color("white")
        self.shape("circle")
        self.penup()

    def move(self):
        x_coor=self.xcor()+10
        y_coor=self.ycor()+10
        self.goto(x_coor,y_coor)



r_paddle=paddle((350,0))
l_paddle=paddle((-350,0))
ball=ball()
screen=Screen()
screen.bgcolor("black")
screen.setup(height=600,width=800)


screen.tracer(0)
screen.listen()
screen.onkey(r_paddle.go_up,"Up")
screen.onkey(r_paddle.go_down,"Down")

game_is_on=True
while game_is_on:
    sleep(0.1)
    screen.update()
    ball.move()



screen.exitonclick()
```

In [ ]:

```
# Step:4 Detect collision and bounce in the wall


import turtle as t
from turtle import Screen,Turtle
from time import sleep


class paddle(Turtle):
    def __init__(self,position):
        super().__init__()
        self.color("white")
        self.shape("square")
        self.turtlesize(stretch_wid=5,stretch_len=1)
        self.penup()
        self.goto(position)

    def go_up(self):
        new_y=self.ycor()+20
        self.goto(self.xcor(),new_y)

    def go_down(self):
        new_y=self.ycor()-20
        self.goto(self.xcor(),new_y)
```

```
class ball(Turtle):
    def __init__(self):
        super().__init__()
        self.color("white")
        self.shape("circle")
        self.penup()
        self.x_move=10
        self.y_move=10

    def move(self):
        x_coor=self.xcor()+self.x_move
        y_coor=self.ycor()+self.y_move
        self.goto(x_coor,y_coor)
    def bounce(self):
        self.y_move *= -1



r_paddle=paddle((350,0))
l_paddle=paddle((-350,0))
ball=ball()
screen=Screen()
screen.bgcolor("black")
screen.setup(height=600,width=800)


screen.tracer(0)
screen.listen()
screen.onkey(r_paddle.go_up,"Up")
screen.onkey(r_paddle.go_down,"Down")

game_is_on=True
while game_is_on:
    sleep(0.1)
    screen.update()
    ball.move()
    if ball.ycor() > 280 or ball.ycor() < -280:
        ball.bounce()



screen.exitonclick()
```

In [ ]:

```
#Step-5 Detect collison with paddle


import turtle as t
from turtle import Screen,Turtle
from time import sleep


class paddle(Turtle):
    def __init__(self,position):
        super().__init__()
        self.color("white")
        self.shape("square")
        self.turtlesize(stretch_wid=5,stretch_len=1)
        self.penup()
```

```
        self.goto(position)

    def go_up(self):
        new_y=self.ycor()+20
        self.goto(self.xcor(),new_y)

    def go_down(self):
        new_y=self.ycor()-20
        self.goto(self.xcor(),new_y)

class ball(Turtle):
    def __init__(self):
        super().__init__()
        self.color("white")
        self.shape("circle")
        self.penup()
        self.x_move=10
        self.y_move=10

    def move(self):
        x_coor=self.xcor()+self.x_move
        y_coor=self.ycor()+self.y_move
        self.goto(x_coor,y_coor)
    def y_bounce(self):
        self.y_move *= -1

    def x_bounce(self):
        self.x_move *= -1




r_paddle=paddle((350,0))
l_paddle=paddle((-350,0))
ball=ball()
screen=Screen()
screen.bgcolor("black")
screen.setup(height=600,width=800)


screen.tracer(0)
screen.listen()
screen.onkey(r_paddle.go_up,"Up")
screen.onkey(r_paddle.go_down,"Down")
screen.onkey(l_paddle.go_up,"w")
screen.onkey(l_paddle.go_down,"s")

game_is_on=True
while game_is_on:
    sleep(0.1)
    screen.update()
    ball.move()
    if ball.ycor() > 280 or ball.ycor() < -280:
        ball.y_bounce()
    if ball.distance(r_paddle) < 50 and ball.xcor() > 320 or ball.distance(l_paddle) <
        ball.x_bounce()
```

```
screen.exitonclick()
```

In [ ]:

```
# Step-6: Final code for the ping pong game with score board update


import turtle as t
from turtle import Screen,Turtle
from time import sleep


class paddle(Turtle):
    def __init__(self,position):
        super().__init__()
        self.color("white")
        self.shape("square")
        self.turtlesize(stretch_wid=5,stretch_len=1)
        self.penup()
        self.goto(position)

    def go_up(self):
        new_y=self.ycor()+20
        self.goto(self.xcor(),new_y)

    def go_down(self):
        new_y=self.ycor()-20
        self.goto(self.xcor(),new_y)

class ball(Turtle):
    def __init__(self):
        super().__init__()
        self.color("white")
        self.shape("circle")
        self.penup()
        self.x_move=10
        self.y_move=10


    def move(self):
        x_coor=self.xcor()+self.x_move
        y_coor=self.ycor()+self.y_move
        self.goto(x_coor,y_coor)
    def y_bounce(self):
        self.y_move *= -1

    def x_bounce(self):
        self.x_move *= -1
    def refresh(self):
        self.goto(0,0)
        self.x_bounce()

class score_board(Turtle):
    def __init__(self):
        super().__init__()
        self.color("white")
        self.hideturtle()
        self.penup()
        self.lscore=0
        self.rscore=0
```

```
        self.update_scoreboard()
    def update_scoreboard(self):
        self.clear()
        self.goto(-100,200)
        self.write(self.lscore,align="center",font=("Courier",80,"normal"))
        self.goto(100,200)
        self.write(self.rscore,align="center",font=("Courier",80,"normal"))
    def l_point(self):
        score.lscore += 1
        self.update_scoreboard()
    def r_point(self):
        score.rscore += 1
        self.update_scoreboard()



r_paddle=paddle((350,0))
l_paddle=paddle((-350,0))
ball=ball()
screen=Screen()
score=score_board()
screen.bgcolor("black")
screen.setup(height=600,width=800)


screen.tracer(0)
screen.listen()
screen.onkey(r_paddle.go_up,"Up")
screen.onkey(r_paddle.go_down,"Down")
screen.onkey(l_paddle.go_up,"w")
screen.onkey(l_paddle.go_down,"s")

game_is_on=True
while game_is_on:
    sleep(0.1)
    screen.update()
    ball.move()
    if ball.ycor() > 280 or ball.ycor() < -280:
        ball.y_bounce()
    if ball.distance(r_paddle) < 50 and ball.xcor() > 320 or ball.distance(l_paddle) <
        ball.x_bounce()
    if ball.xcor() > 380:
        ball.refresh()
        score.l_point()
    if ball.xcor()< -380:
        ball.refresh()
        score.r_point()









screen.exitonclick()
```

DAY-23

In [ ]:
```
# Create a turtle player that starts at the bottom of the screen and listen for the "Up
import time
from turtle import Screen
from turtle import Turtle

STARTING_POSITION = (0, -280)
MOVE_DISTANCE = 10
FINISH_LINE_Y = 280


class Player(Turtle):
    def __init__(self):
        super().__init__()
        self.shape("turtle")
        self.penup()
        self.left(90)
        self.goto(STARTING_POSITION)

    def go_up(self):
        new_y=self.ycor()+MOVE_DISTANCE
        self.goto(self.xcor(),new_y)

    def go_down(self):
        new_y=self.ycor()-MOVE_DISTANCE
        self.goto(self.xcor(),new_y)

turtle1=Player()
screen = Screen()
screen.setup(width=600, height=600)
screen.tracer(0)
screen.listen()
screen.onkey(turtle1.go_up,"Up")
screen.onkey(turtle1.go_down,"Down")


game_is_on = True
while game_is_on:

    time.sleep(0.1)
    screen.update()
```

In [ ]:
```
#Above task continuination
# Create cars that are 20px high by 40px wide that are randomly generated along the y-a
#left edge of the screen. No cars should be generated in the top and bottom 50px of the
#as a safe zone for our little turtle). Hint: generate a new car only every 6th time th
import random
import time
from turtle import Screen
from turtle import Turtle

STARTING_POSITION = (0, -280)
MOVE_DISTANCE = 10
FINISH_LINE_Y = 280


class Player(Turtle):
    def __init__(self):
        super().__init__()
```

```
        self.shape("turtle")
        self.penup()
        self.left(90)
        self.goto(STARTING_POSITION)

    def go_up(self):
        new_y=self.ycor()+MOVE_DISTANCE
        self.goto(self.xcor(),new_y)

    def go_down(self):
        new_y=self.ycor()-MOVE_DISTANCE
        self.goto(self.xcor(),new_y)

COLORS = ["red", "orange", "yellow", "green", "blue", "purple"]
STARTING_MOVE_DISTANCE = 5
MOVE_INCREMENT = 10


class CarManager():
    def __init__(self):
        self.all_cars=[]
    def create_cars(self):
        random_chance=random.randint(1,6)
        if random_chance==1:
            new_car=Turtle("square")

            new_car.turtlesize(stretch_wid=1,stretch_len=2)
            new_car.penup()
            new_car.color(random.choice(COLORS))

            random_y=random.randint(-250,250)
            new_car.goto(360,random_y)
            self.all_cars.append(new_car)

    def move(self):
        for car in self.all_cars:
            car.backward(STARTING_MOVE_DISTANCE)


car_manager=CarManager()
turtle1=Player()
screen = Screen()
screen.setup(width=600, height=600)
screen.tracer(0)
screen.listen()
screen.onkey(turtle1.go_up,"Up")
screen.onkey(turtle1.go_down,"Down")


game_is_on = True
while game_is_on:

    time.sleep(0.1)
    screen.update()
    car_manager.create_cars()
    car_manager.move()
    for car in car_manager.all_cars:
        if car.distance(turtle1) < 20:
            game_is_on=False
```

```
screen.exitonclick()
```

Turtle Car Crossing Game Final Code

In [78]:

```
# Detect when the turtle player has reached the top edge of the screen (i.e., reached t
#return the turtle to the starting position and increase the speed of the cars. Hint: t
#using the MOVE_INCREMENT to increase the car speed

#Create a scoreboard that keeps track of which level the user is on. Every time the tur
#crossing, the level should increase. When the turtle hits a car, GAME OVER should be d
#If you get stuck,

import random
import time
from turtle import Screen,Turtle

STARTING_POSITION = (0, -280)
MOVE_DISTANCE = 10
FINISH_LINE_Y = 280


class Player(Turtle):
    def __init__(self):
        super().__init__()
        self.shape("turtle")
        self.penup()
        self.left(90)
        self.goto(STARTING_POSITION)

    def go_up(self):
        new_y=self.ycor()+MOVE_DISTANCE
        self.goto(self.xcor(),new_y)

    def go_down(self):
        new_y=self.ycor()-MOVE_DISTANCE
        self.goto(self.xcor(),new_y)
    def change_pos(self):
        self.goto(STARTING_POSITION)


COLORS = ["red", "orange", "yellow", "green", "blue", "purple"]
STARTING_MOVE_DISTANCE = 5
MOVE_INCREMENT = 10


class CarManager:
    def __init__(self):
        self.all_cars=[]

        self.car_speed=STARTING_MOVE_DISTANCE
    def create_cars(self):
        random_chance=random.randint(1,6)
        if random_chance==1:
            new_car=Turtle("square")

            new_car.turtlesize(stretch_wid=1,stretch_len=2)
            new_car.penup()
```

```
            new_car.color(random.choice(COLORS))


            random_y=random.randint(-250,250)
            new_car.goto(360,random_y)
            self.all_cars.append(new_car)

    def move(self):
        for car in self.all_cars:
            car.backward(self.car_speed)
    def level_up(self):
        self.car_speed += MOVE_INCREMENT

FONT = ("Courier", 24, "normal")

class Scoreboard(Turtle):
    def __init__(self):
        super().__init__()
        self.level=1
        self.color("Black")
        self.penup()
        self.hideturtle()
        self.update_score_board()


    def update_score_board(self):
        self.clear()
        self.goto(-230,250)
        self.write(f"Level:{self.level}",align="center",font=FONT)

    def score_increment(self):
        self.level += 1
        self.update_score_board()
    def game_over(self):
        self.clear()
        self.goto(0,0)
        self.write("GAME OVER",align="center",font=FONT)



screen = Screen()
turtle1=Player()
car_manager=CarManager()


score=Scoreboard()
screen.setup(width=600, height=600)
screen.tracer(0)
screen.listen()
screen.onkey(turtle1.go_up,"Up")
screen.onkey(turtle1.go_down,"Down")


game_is_on = True
while game_is_on:

    time.sleep(0.1)
    screen.update()
    car_manager.create_cars()
    car_manager.move()
    for car in car_manager.all_cars:
```

```
        if car.distance(turtle1) < 20:
            score.game_over()
            game_is_on=False
    if turtle1.ycor() >FINISH_LINE_Y:
        turtle1.change_pos()
        score.score_increment()
        car_manager.level_up()

screen.exitonclick()
```

DAY-24

How to Open, Read, and Write to Files using the "with" Keyword

In [94]:
```
#Reading the content from the file
#If we use this method we have to close the file
file=open("file.txt","r")
content=file.read()
print(content)
file.close()
```

```
Hello this is srenath Kumar
I am 18 years old
```

In [95]:
```
# with (with method) we no need to close the file as it automatically closes after comp

with open("file.txt","r") as file:
    content=file.read()
    print(content)
```

```
Hello this is srenath Kumar
I am 18 years old
```

In [96]:
```
# write mode with {with method} write will destroy all the existing data and writes it

with open("file.txt","w") as file:
    file.write("New text writen successfully")
```

In [97]:
```
with open("file.txt","r") as file:
    content=file.read()
    print(content)
```

```
New text writen successfully
```

In [98]:
```
# append mode will add the details below the existing datas
with open("file.txt","a") as file:
    file.write("Second Text added")
```

In [99]:
```
with open("file.txt","r") as file:
    content=file.read()
    print(content)
```

```
New text writen successfullySecond Text added
```

Writting the scores of the snake game which was created previously

In [100…

```
from turtle import Turtle
STARTING_POSITIONS = [(0, 0), (-20, 0), (-40, 0)]
MOVE_DISTANCE = 20
UP = 90
DOWN = 270
LEFT = 180
RIGHT = 0


class Snake:

    def __init__(self):
        self.segments = []
        self.create_snake()
        self.head = self.segments[0]

    def create_snake(self):
        for position in STARTING_POSITIONS:
            self.add_segment(position)

    def add_segment(self, position):
        new_segment = Turtle("square")
        new_segment.color("white")
        new_segment.penup()
        new_segment.goto(position)
        self.segments.append(new_segment)

    def extend(self):
        self.add_segment(self.segments[-1].position())

    def move(self):
        for seg_num in range(len(self.segments) - 1, 0, -1):
            new_x = self.segments[seg_num - 1].xcor()
            new_y = self.segments[seg_num - 1].ycor()
            self.segments[seg_num].goto(new_x, new_y)
        self.head.forward(MOVE_DISTANCE)

    def up(self):
        if self.head.heading() != DOWN:
            self.head.setheading(UP)

    def down(self):
        if self.head.heading() != UP:
            self.head.setheading(DOWN)

    def left(self):
        if self.head.heading() != RIGHT:
            self.head.setheading(LEFT)

    def right(self):
        if self.head.heading() != LEFT:
            self.head.setheading(RIGHT)
```

In [101…

```
from turtle import Turtle
ALIGNMENT = "center"
FONT = ("Courier", 24, "normal")


class Scoreboard(Turtle):

    def __init__(self):
        super().__init__()
        self.score = 0
        with open("data.txt") as data:
            self.high_score = int(data.read())
        self.color("white")
        self.penup()
        self.goto(0, 270)
        self.hideturtle()
        self.update_scoreboard()

    def update_scoreboard(self):
        self.clear()
        self.write(f"Score: {self.score} High Score: {self.high_score}", align=ALIGNMEN

    def reset(self):
        if self.score > self.high_score:
            self.high_score = self.score
            with open("data.txt", mode="w") as data:
                data.write(f"{self.high_score}")
        self.score = 0
        self.update_scoreboard()

    def increase_score(self):
        self.score += 1
        self.update_scoreboard()
```

In [102…

```
from turtle import Turtle
import random


class Food(Turtle):

    def __init__(self):
        super().__init__()
        self.shape("circle")
        self.penup()
        self.shapesize(stretch_len=0.5, stretch_wid=0.5)
        self.color("blue")
        self.speed("fastest")
        self.refresh()

    def refresh(self):
        random_x = random.randint(-280, 280)
        random_y = random.randint(-280, 280)
        self.goto(random_x, random_y)
```

In [109…

```
from turtle import Screen
#from snake import Snake
#from food import Food
#from scoreboard import Scoreboard
```

```
import time

screen = Screen()
screen.setup(width=600, height=600)
screen.bgcolor("black")
screen.title("My Snake Game")
screen.tracer(0)

snake = Snake()
food = Food()
scoreboard = Scoreboard()

screen.listen()
screen.onkey(snake.up, "Up")
screen.onkey(snake.down, "Down")
screen.onkey(snake.left, "Left")
screen.onkey(snake.right, "Right")

game_is_on = True
while game_is_on:
    screen.update()
    time.sleep(0.1)
    snake.move()

    #Detect collision with food.
    if snake.head.distance(food) < 15:
        food.refresh()
        snake.extend()
        scoreboard.increase_score()

    #Detect collision with wall.
    if snake.head.xcor() > 280 or snake.head.xcor() < -280 or snake.head.ycor() > 280 o
        game_is_on = False
        scoreboard.game_over()

    #Detect collision with tail.
    for segment in snake.segments:
        if segment == snake.head:
            pass
        elif snake.head.distance(segment) < 10:
            game_is_on = False
            scoreboard.game_over()




screen.exitonclick()
```

Mail Merge Challenge

In [1]:

```
PLACEHOLDER = "[name]"


with open("./Input/Names/invited_names.txt") as names_file:
    names = names_file.readlines()

with open("./Input/Letters/starting_letter.txt") as letter_file:
    letter_contents = letter_file.read()
```

```
    for name in names:
        stripped_name = name.strip()
        new_letter = letter_contents.replace(PLACEHOLDER, stripped_name)
        with open(f"./Output/ReadyToSend/letter_for_{stripped_name}.txt", mode="w") as
            completed_letter.write(new_letter)
```

NOTE: The code is not about make it shorterning it is about the readabality

The code you wrote should be understood by others

DAY-25

Working with CSV Data and the Pandas Library

In [ ]:
```
#Pandas Documentation
# https://pandas.pydata.org/docs/index.html
```

In [20]:
```
# Opening csv file with out pandas
import csv

with open("weather_data.csv") as file:
    data=csv.reader(file)
    temperatures=[]
    for i in data:
        print(i)
```

```
['day', 'temp', 'condition']
['Monday', '12', 'Sunny']
['Tuesday', '14', 'Rain']
['Wednesday', '15', 'Rain']
['Thursday', '14', 'Cloudy']
['Friday', '21', 'Sunny']
['Saturday', '22', 'Sunny']
['Sunday', '24', 'Sunny']
```

In [3]:
```
#Opening csv file with pandas
import pandas as pd

df=pd.read_csv("weather_data.csv")
values=df["temp"]
```

In [4]:
```
values
```

Out[4]:
```
0    12
1    14
2    15
3    14
```

```
4    21
5    22
6    24
Name: temp, dtype: int64
```

In [5]:
```
# In Datas there are two types of data Structures : Series(1-Dimensional) and Data Fram
```

In [6]:
```
#Eg for series
type(values)
```

Out[6]:
```
pandas.core.series.Series
```

In [8]:
```
#Eg for dataframe
type(df)
```

Out[8]:
```
pandas.core.frame.DataFrame
```

In [10]:
```
#converting dataframe to dict

df_dict=df.to_dict()
```

In [11]:
```
df_dict
```

Out[11]:
```
{'day': {0: 'Monday',
  1: 'Tuesday',
  2: 'Wednesday',
  3: 'Thursday',
  4: 'Friday',
  5: 'Saturday',
  6: 'Sunday'},
 'temp': {0: 12, 1: 14, 2: 15, 3: 14, 4: 21, 5: 22, 6: 24},
 'condition': {0: 'Sunny',
  1: 'Rain',
  2: 'Rain',
  3: 'Cloudy',
  4: 'Sunny',
  5: 'Sunny',
  6: 'Sunny'}}
```

In [14]:
```
#Convetting dataframe to lists

data_list=df["temp"].to_list()
```

In [16]:
```
#calculating average tempertature of the list
print(sum(data_list)/len(data_list))
```

```
17.428571428571427
```

In [17]:
```
#finding the max temperature in the data series
df["temp"].max()
```

Out[17]:
```
24
```

In [19]:
```
#This below condition is used to get the datas of the row
print(df[df.day=="Monday"])
```

```
      day  temp condition
0  Monday    12     Sunny
```

In [23]:
```
#Finding the max temperature row in the data

print(df[df.temp==df["temp"].max()])
```

```
      day  temp condition
6  Sunday    24     Sunny
```

In [24]:
```
# Finding a max temperature and condition of the row
print(df[df.temp==df["temp"].max()]["condition"])
```

```
6    Sunny
Name: condition, dtype: object
```

In [34]:
```
#converting mondays temperature from celsius to fahrenheit
cel_temp=df[df.day=="Monday"]["temp"][0]
```

In [36]:
```
fah_temp=(cel_temp*(9/5))+32
```

In [37]:
```
cel_temp
```

Out[37]:
```
12
```

In [38]:
```
fah_temp
```

Out[38]:
```
53.6
```

In [40]:
```
# Creating a dataframe from scratch

df={
    "Name":["Srenath","Kumar","Arjun"],
    "Score":[21,21,21]
}
```

In [41]:
```
df
```

Out[41]:
```
{'Name': ['Srenath', 'Kumar', 'Arjun'], 'Score': [21, 21, 21]}
```

In [44]:
```
dataframe=pd.DataFrame(df)
dataframe
```

Out[44]:

	Name	Score

	Name	Score
0	Srenath	21
1	Kumar	21
2	Arjun	21

In [46]:
```
#converting the data frame to csv

dataframe.to_csv("new_data_file.csv")
```

The Great Squirrel Census Data Analysis (with Pandas!)

In [4]:
```
df=pd.read_csv("2018_Central_Park_Squirrel_Census_-_Squirrel_Data.csv")
```

In [5]:
```
df.head()
```

Out[5]:

	X	Y	Unique Squirrel ID	Hectare	Shift	Date	Hectare Squirrel Number	Age	Primary Fur Color	Highlight Fur Color	..
0	-73.956134	40.794082	37F-PM-1014-03	37F	PM	10142018	3	NaN	NaN	NaN	..
1	-73.957044	40.794851	37E-PM-1006-03	37E	PM	10062018	3	Adult	Gray	Cinnamon	..
2	-73.976831	40.766718	2E-AM-1010-03	02E	AM	10102018	3	Adult	Cinnamon	NaN	..
3	-73.975725	40.769703	5D-PM-1018-05	05D	PM	10182018	5	Juvenile	Gray	NaN	..
4	-73.959313	40.797533	39B-AM-1018-01	39B	AM	10182018	1	NaN	NaN	NaN	..

5 rows × 31 columns

In [15]:
```
cinnamon_color_squril=len(df[df["Highlight Fur Color"]=="Cinnamon"])
```

In [16]:
```
cinnamon_color_squril
```

Out[16]: 767

In [17]:
```
gray_color_squril=len(df[df["Highlight Fur Color"]=="Gray"])
```

In [18]:
```
gray_color_squril
```

Out[18]: 170

In [19]:
```
black_color_squril=len(df[df["Highlight Fur Color"]=="Black"])
```

In [20]:
```
black_color_squril
```

Out[20]: 34

In [30]:
```
dicta={
    "Fur Color":["Black","Cinnamon","Gray"],
    "Color Count":[black_color_squril,cinnamon_color_squril,gray_color_squril]
}
```

In [31]:
```
dataframe3=pd.DataFrame(dicta)
dataframe3
```

Out[31]:

	Fur Color	Color Count
0	Black	34
1	Cinnamon	767
2	Gray	170

In [32]:
```
dataframe3.to_csv("dataframe3.csv")
```

In [33]:
```
dataframe3.head()
```

Out[33]:

	Fur Color	Color Count
0	Black	34
1	Cinnamon	767
2	Gray	170

In [ ]:
```
# Creating a US Game with csv file
import turtle
import pandas

screen = turtle.Screen()
screen.title("U.S. States Game")
image = "blank_states_img.gif"
screen.addshape(image)
turtle.shape(image)

data = pandas.read_csv("50_states.csv")
```

```
all_states = data.state.to_list()
guessed_states = []

while len(guessed_states) < 50:
    answer_state = screen.textinput(title=f"{len(guessed_states)}/50 States Correct",
                                    prompt="What's another state's name?").title()
    if answer_state == "Exit":
        missing_states = []
        for state in all_states:
            if state not in guessed_states:
                missing_states.append(state)
        new_data = pandas.DataFrame(missing_states)
        new_data.to_csv("states_to_learn.csv")
        break
    if answer_state in all_states:
        guessed_states.append(answer_state)
        t = turtle.Turtle()
        t.hideturtle()
        t.penup()
        state_data = data[data.state == answer_state]
        t.goto(int(state_data.x), int(state_data.y))
        t.write(answer_state)
```

DAY-26

How to Create Lists using List Comprehension

This list comprehension helps to shorten the code

i,e Adding all the for loop ,while if etc to al single list (inside the brackets)

In [2]:
```
numbers=[1,2,3]
new_numbers=[i+1 for i in numbers]
```

In [3]:
```
new_numbers
```

Out[3]: [2, 3, 4]

In [6]:
```
name="Srenath"
new_sequence=[i for i in name]
```

In [7]:
```
new_sequence
```

Out[7]: ['S', 'r', 'e', 'n', 'a', 't', 'h']

In [8]:
```
# Challange:Creating a new list from the range and double it

new_list_double=[i*2 for i in range(1,5)]
```

In [9]:
```
new_list_double
```

Out[9]: [2, 4, 6, 8]

In [15]:
```
names=["Angela","Harini","Srenath"]
list2=[i for i in names if len(i) <10]
```

In [16]:
```
list2
```

Out[16]: ['Angela', 'Harini', 'Srenath']

In [18]:
```
#Challange: Converting the names to capital words with the lenght of names is lesser th
names1=["Angela","Harini","Srenath","Arjun","Kal","ca"]
list3=[i.upper() for i in names1 if len(i) <5]
```

In [19]:
```
list3
```

Out[19]: ['KAL', 'CA']

[Interactive Coding Exercise] Squaring Numbers

In [21]:
```
# You are going to write a List Comprehension to create a new list called `squared_numb
#This new list should contain every number in the list `numbers` but each number should

numbers = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

squared_numbers=[i**2 for i in numbers]
squared_numbers
```

Out[21]: [1, 1, 4, 9, 25, 64, 169, 441, 1156, 3025]

[Interactive Coding Exercise] Filtering Even Numbers

In [22]:
```
# You are going to write a List Comprehension to create a new list called `result`.
#This new list should only contain the even numbers from the list `numbers`
numbers = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

result=[i for i in numbers if i%2==0]
result
```

Out[22]: [2, 8, 34]

[Interactive Coding Exercise] Data Overlap

In [ ]:
```
#Take a look inside **file1.txt** and **file2.txt**. They each contain a bunch of numbe
#You are going to create a list called result which contains the numbers that are commo

with open("file1.txt","r") as file1:
    content1=file1.readlines()
```

```
with open("file2.txt","r") as file2:
    content2=file2.readlines()

result=[int(i) for i in content1  if i in content2]
print(result)

Output:
    [3, 6, 5, 33, 12, 7, 42, 13]
```

Apply List Comprehension to the U.S. States Game

In [ ]:

```
#Change the us state game to list comprehension

import turtle
import pandas

screen = turtle.Screen()
screen.title("U.S. States Game")
image = "blank_states_img.gif"
screen.addshape(image)
turtle.shape(image)

data = pandas.read_csv("50_states.csv")
all_states = data.state.to_list()
guessed_states = []

while len(guessed_states) < 50:
    answer_state = screen.textinput(title=f"{len(guessed_states)}/50 States Correct",
                                    prompt="What's another state's name?").title()
    if answer_state == "Exit":
        missing_states = [state for state in all_states if state not in guessed_states]
        new_data = pandas.DataFrame(missing_states)
        new_data.to_csv("states_to_learn.csv")
        break
    if answer_state in all_states:
        guessed_states.append(answer_state)
        t = turtle.Turtle()
        t.hideturtle()
        t.penup()
        state_data = data[data.state == answer_state]
        t.goto(int(state_data.x), int(state_data.y))
        t.write(answer_state)
```

How to Create dictionary using dictionary Comprehension

In [7]:

```
import random
names=["Srenath","Kumar","Arjun","Harini"]
#Format:variable={keys:values condition}
student_score={student:random.randint(1,100) for student in names}
student_score
```

Out[7]: {'Srenath': 71, 'Kumar': 39, 'Arjun': 78, 'Harini': 65}

In [9]:

```
# Format  variable={new_key:new_value for (key:value) in loop_variable }
passed_students={student:score for (student,score) in student_score.items() if score >6
```

In [10]:
```
passed_students
```

Out[10]:
```
{'Srenath': 71, 'Arjun': 78, 'Harini': 65}
```

[Interactive Coding Exercise] Dictionary Comprehension 1

In [11]:
```
sentence = "What is the Airspeed Velocity of an Unladen Swallow?"
# Don't change code above 👆

# Write your code below:


result={word:len(word) for word in sentence.split() }

print(result)
```

```
{'What': 4, 'is': 2, 'the': 3, 'Airspeed': 8, 'Velocity': 8, 'of': 2, 'an': 2, 'Unlade
n': 7, 'Swallow?': 8}
```

[Interactive Coding Exercise] Dictionary Comprehension 2

In [12]:
```
weather_c = {
    "Monday": 12,
    "Tuesday": 14,
    "Wednesday": 15,
    "Thursday": 14,
    "Friday": 21,
    "Saturday": 22,
    "Sunday": 24,
}
# 🚨 Don't change code above 👆


# Write your code 👇 below:
weather_f={day:(temp_c*9/5)+32 for (day,temp_c) in weather_c.items()}



print(weather_f)
```

```
{'Monday': 53.6, 'Tuesday': 57.2, 'Wednesday': 59.0, 'Thursday': 57.2, 'Friday': 69.8,
'Saturday': 71.6, 'Sunday': 75.2}
```

How to Iterate over a Pandas DataFrame

In [16]:
```
student_dict={
    "student":["Angela","Srenath","Kumar"],
    "scores":[89,76,98]
}

import pandas as pd

df=pd.DataFrame(student_dict)
```

In [17]:
```
df
```

Out[17]:

	student	scores
0	Angela	89
1	Srenath	76
2	Kumar	98

In [20]:
```
#We use iterrows in for loop to get the students name from the data frame
for (index,row) in df.iterrows():
    print(row.student)
```

```
Angela
Srenath
Kumar
```

In [21]:
```
#WE can loop through with all the conditions
for (index,row) in df.iterrows():
    if row.student== "Srenath":
        print("Found")
```

```
Found
```

Project:Introducing the NATO Alphabet Project

In [26]:
```
import pandas

data = pandas.read_csv("nato_phonetic_alphabet.csv")
#TODO 1. Create a dictionary in this format:
phonetic_dict = {row.letter: row.code for (index, row) in data.iterrows()}
print(phonetic_dict)

#TODO 2. Create a list of the phonetic code words from a word that the user inputs.

word = input("Enter a word: ").upper()
output_list = [phonetic_dict[letter] for letter in word]
print(output_list)
```

```
{'A': 'Alfa', 'B': 'Bravo', 'C': 'Charlie', 'D': 'Delta', 'E': 'Echo', 'F': 'Foxtrot',
'G': 'Golf', 'H': 'Hotel', 'I': 'India', 'J': 'Juliet', 'K': 'Kilo', 'L': 'Lima', 'M':
'Mike', 'N': 'November', 'O': 'Oscar', 'P': 'Papa', 'Q': 'Quebec', 'R': 'Romeo', 'S': 'S
ierra', 'T': 'Tango', 'U': 'Uniform', 'V': 'Victor', 'W': 'Whiskey', 'X': 'X-ray', 'Y':
'Yankee', 'Z': 'Zulu'}

['Sierra', 'Romeo', 'Echo', 'November', 'Alfa', 'Tango', 'Hotel']
```

DAY-27

Tkinter Module

In [8]:
```
import tkinter
#This Tk() is a method of class tkinter and window is a object
#This window is a screen
window=tkinter.Tk()
#Adding title in the tkinter
window.title("My First GUI Program")
```

```
#chanfing the size
window.minsize(width=500,height=300)
#Creating a label
label_1=tkinter.Label(text="This is Test",font=("Arial",24,"bold"))
#Adding label to the window
#This pack functio helps to change the position on specifing it
label_1.pack(side="left")
#In onder to run the window continously we use mainloop
window.mainloop()
```

Default Arguments

In [7]:
```
def functions(a=1,b=2,c=3):
    print(a,b,c)

functions()
```

```
1 2 3
```

In [8]:
```
functions(4)
```

```
4 2 3
```

*args : Many Positional Arguments

In [9]:
```
#This args is used to get unlimited arguments in the function
#Here i
def funct(*args):
    for n in args:
        print(n)
funct(2,4,3,6,4)
```

```
2
4
3
6
4
```

In [6]:
```
#Challange is to add the unlimited input of the add function

def add(*args):
    print(args[0])
    sum=0
    for n in args:
        sum += n
    print(sum)
add(3,4,3)
```

```
3
10
```

In [7]:
```
# We can also the name of args to any names

def add(*nums):
    print(nums[0])
    sum=0
    for n in nums:
```

```
        sum += n
    print(sum)
add(3,4,3)
```

```
3
10
```

**kwargs: Many Keyword Arguments

In [12]:
```
def calculate(**kwargs):
    print(kwargs)
```

In [13]:
```
calculate(add=3,multiply=4)
```

```
{'add': 3, 'multiply': 4}
```

In [14]:
```
#This **kwargs helps to get the unlimited value and converts it into dictionary
```

Example for variable *args and **kwargs

In [15]:
```
#Variable will be in given type for example here a=4 which is integer so the type of a
#The type of args is tuple
#The type **kwargs is dictionary
def all_aboard(a, *args, **kw):
    print(a, args, kw)


all_aboard(4, 7, 3, 0, x=10, y=64)
```

```
4 (7, 3, 0) {'x': 10, 'y': 64}
```

WE use , import tkinter as * which helps to get all the class and functions from the module

In [18]:
```
import tkinter

window=tkinter.Tk()
window.minsize(width=500,height=300)


label_1=tkinter.Label(text="This is Test",font=("Arial",24,"bold"))
label_1.pack()
#Function when button clicked
def buttons():
    print("I got clicked")
#Creating a button  and command is the responding function when clicked
button_1=tkinter.Button(text="Click Me",command=buttons) #Remember to not add () in the
button_1.pack()
#In onder to run the window continously we use mainloop
window.mainloop()
```

```
I got clicked
I got clicked
```

In [22]:
```
#Challange: When the button is clicked it should change th label to I got clicked
```

```
import tkinter

window=tkinter.Tk()
window.minsize(width=500,height=300)


label_1=tkinter.Label(text="This is Test",font=("Arial",24,"bold"))
label_1.pack()
#Function when button clicked
def button_clicked():
    #This config method is used to chnage existing text to given text
    label_1.config(text="I got clicked")

#Creating a button  and command is the responding function when clicked
button_1=tkinter.Button(text="Click Me",command=button_clicked) #Remember to not add ()
button_1.pack()
#In onder to run the window continously we use mainloop
window.mainloop()
```

Entry() method in tkinter

In [38]:

```
import tkinter

window=tkinter.Tk()
window.minsize(width=500,height=300)
label_1=tkinter.Label(text="This is text")
label_1.pack()
#This width changes the side of the input bar
input1=tkinter.Entry(width=30)
input1.pack()

def inputted():
    #This get function helps to return the inputted string
    print(input1.get())
button1=tkinter.Button(text="Inputting",command=inputted)
button1.pack()


window.mainloop()
```

```
Srenath Kumar
Sreja
```

In [40]:

```
#Challange:When the button is clicked i need to print the inputted text

import tkinter

window=tkinter.Tk()
window.minsize(width=500,height=300)

label1=tkinter.Label(text="This is the test")
label1.pack()

input1=tkinter.Entry()
input1.pack()

def inputted():
    label1.config(text=input1.get())
button1=tkinter.Button(text="Click Here",command=inputted)
```

```
button1.pack()

window.mainloop()
```

tkinter-widget

In [42]:

```
from tkinter import *

#Creating a new window and configurations
window = Tk()
window.title("Widget Examples")
window.minsize(width=500, height=500)

#Labels
label = Label(text="This is old text")
label.config(text="This is new text")
label.pack()

#Buttons
def action():
    print("Do something")

#calls action() when pressed
button = Button(text="Click Me", command=action)
button.pack()

#Entries
entry = Entry(width=30)
#Add some text to begin with
#This inset function helps to add some text before we input
#It is like we give hints to it
entry.insert(END, string="Some text to begin with.")
#Gets text in entry
print(entry.get())
entry.pack()

#Text
text = Text(height=5, width=30)
#Puts cursor in textbox.
text.focus()
#Adds some text to begin with.
text.insert(END, "Example of multi-line text entry.")
#Get's current value in textbox at line 1, character 0
print(text.get("1.0", END))
text.pack()

#Spinbox
def spinbox_used():
    #gets the current value in spinbox.
    print(spinbox.get())
spinbox = Spinbox(from_=0, to=10, width=5, command=spinbox_used)
spinbox.pack()

#Scale
#Called with current scale value.
def scale_used(value):
    print(value)
scale = Scale(from_=0, to=100, command=scale_used)
scale.pack()
```

```
#Checkbutton
def checkbutton_used():
    #Prints 1 if On button checked, otherwise 0.
    print(checked_state.get())
#variable to hold on to checked state, 0 is off, 1 is on.
checked_state = IntVar()
checkbutton = Checkbutton(text="Is On?", variable=checked_state, command=checkbutton_us
checked_state.get()
checkbutton.pack()

#Radiobutton
def radio_used():
    print(radio_state.get())
#Variable to hold on to which radio button value is checked.
#Int Var is a method to track the check box
radio_state = IntVar()
radiobutton1 = Radiobutton(text="Option1", value=1, variable=radio_state, command=radio
radiobutton2 = Radiobutton(text="Option2", value=2, variable=radio_state, command=radio
radiobutton1.pack()
radiobutton2.pack()


#Listbox
def listbox_used(event):
    # Gets current selection from listbox
    print(listbox.get(listbox.curselection()))

listbox = Listbox(height=4)
fruits = ["Apple", "Pear", "Orange", "Banana"]
for item in fruits:
    listbox.insert(fruits.index(item), item)
listbox.bind("<<ListboxSelect>>", listbox_used)
listbox.pack()
window.mainloop()
```

```
Some text to begin with.
Example of multi-line text entry.

1
1
3
1
1
0
2
Orange
```

Tkinter Layout Managers: pack(), place() and grid()

Place()

In [47]:

```
#place but this is difficult to add coordinates in the wigets
import tkinter

window=tkinter.Tk()
window.minsize(width=500,height=300)
label1=tkinter.Label(text="This is a test")
#This place is used to place the labels or button or any other wiget to the given posit
label1.place(x=30,y=100)
window.mainloop()
```

Grid()

In [53]:
```
import tkinter

window=tkinter.Tk()
window.minsize(width=500,height=300)
label1=tkinter.Label(text="This is a test")
#The grid takes rows and columsn input for positioning the wigets
label1.grid(column=0,row=0)

button=tkinter.Button(text="Click me")
button.grid(column=1,row=1)
window.mainloop()
```

In [58]:
```
# Challange is to position the wigets according to given grid image

import tkinter

window=tkinter.Tk()
window.minsize(width=500,height=300)
#This padx and pady helps to give a space in corners
window.config(padx=20,pady=20)
label1=tkinter.Label(text="This is a text")
label1.grid(column=0,row=0)

new_button=tkinter.Button(text="This is new button")
new_button.grid(column=2,row=0)
#This helps to increase the button size
new_button.config(padx=20,pady=20)

button=tkinter.Button(text="This is a button")
button.grid(column=1,row=1)

input1=tkinter.Entry()
input1.grid(column=3,row=2)
window.mainloop()
```

GUI-Mile to Kilometers Converter Project

In [67]:
```
import tkinter

window=tkinter.Tk()
window.title("Mile to Km Converter")
window.config(padx=10,pady=10)
label1=tkinter.Label(text="Miles")
label1.grid(column=2,row=0)
label2=tkinter.Label(text="Km")
label2.grid(column=2,row=1)
label3=tkinter.Label(text="is equal to")
label3.grid(column=0,row=1)
label4=tkinter.Label(text="")
label4.grid(column=1,row=1)
input1=tkinter.Entry()
input1.grid(column=1,row=0)

def response():
```

```
    miles=input1.get()
    km=float(miles)*1.60934
    km_round=round(km,2)
    label4.config(text=km_round)
button=tkinter.Button(text="Calculate",command=response)
button.grid(column=1,row=2)



window.mainloop()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: