Random module

The random() method returns a random floating number between 0 and 1.

In [5]:
```
import random
print(random.random())
```

```
0.4288890546751146
```

seed() Function

The seed() method is used to initialize the random number generator.

The random number generator needs a number to start with (a seed value), to be able to generate a random number.

It will generate the same value for same number in seed().

In [4]:
```
import random
random.seed(10)
print(random.random())
```

```
0.5714025946899135
```

Note: If you use the same seed value twice you will get the same random number twice. See example below

In [61]:
```
import random

random.seed(10)
print(random.random())

random.seed(20)
print(random.random())
```

```
0.5714025946899135
0.9056396761745207
```

In [67]:
```
import random
random.seed()   # if argument not passed then random number everytime
print(random.random())
```

```
0.021068803718420015
```

In [68]:
```
import random
random.seed()   # if argument not passed then random number everytime
print(random.random())
```

```
0.41464068520374786
```

choice() Function

The choice() method returns a randomly selected element from the specified sequence.

The sequence can be a string, a range, a list, a tuple or any other kind of sequence.

In [74]:
```
import random
x = "WELCOME"
```

```
print(random.choice(x))
```

```
M
```

In [76]:
```
import random
mylist = ["apple", "banana", "cherry"]
print(random.choice(mylist))
```

```
banana
```

In [77]:
```
import random
x=[1,2,3,4,5,6]
print(random.choice(x))
```

```
2
```

In [78]:
```
import random
x=["apple"]
print(random.choice(x))
```

```
apple
```

In [79]:
```
import random
x="apple"
print(random.choice(x))
```

```
p
```

randint() Function

The randint() method returns an integer number selected element from the specified range.

In [81]:
```
import random
print(random.randint(3, 9)) #End Point is included
```

```
9
```

randrange() Function

The randrange() method returns a randomly selected element from the specified range.

In [83]:
```
import random
print(random.randrange(3, 9, 2)) #End Point is excluded
```

```
7
```

In [99]:
```
import random
print(random.randrange(3, 9)) #End Point is excluded
```

```
5
```

Use randint() when you want to generate a random number from an inclusive range.

Use randrange() when you want to generate a random number within a range by specifying the increment.

It produces a random number from an exclusive range.

shuffle() Function

The shuffle() method takes a sequence, like a list, and reorganize the order of the items.

In [109…
```python
import random
def myfunction():
    return 0.1
mylist=["apple","banana","cherry"]
random.shuffle(mylist,myfunction) #myfunction or value bet 0 to 1 then single value not
print(mylist)
```

```
['banana', 'cherry', 'apple']
```

In [112…
```python
import random
def myfunction():
    return 0.1
mylist=["apple","banana","cherry"]
random.shuffle(mylist)  #remove myfunction then shuffle will be happening
print(mylist)
```

```
['cherry', 'apple', 'banana']
```

In [128…
```python
import random
mylist=["apple","banana","cherry"]
random.seed(10)
random.shuffle(mylist)  # shuffle will happen only once
print(mylist)
```

```
['banana', 'apple', 'cherry']
```

In [129…
```python
help(random.shuffle)
```

```
Help on method shuffle in module random:

shuffle(x, random=None) method of random.Random instance
    Shuffle list x in place, and return None.

    Optional argument random is a 0-argument function returning a
    random float in [0.0, 1.0); if it is the default None, the
    standard random.random will be used.
```

Mystical Octosphere Game

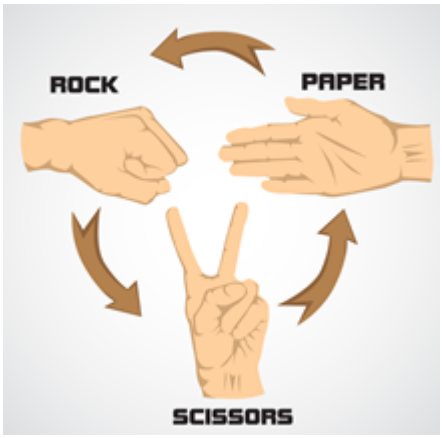
This game is based on a common toy. It is a round black ball with a clear plastic window. The ball is filled with murky blue liquid and you use it as a fortune teller. You ask a yes-or-no question and shake the ball. There is a white many-sided die inside with answers, and when you stop shaking, one of the sides floats up and is readable against the window.

In [130…
```python
import random

def number_to_fortune(number):

    if number == 0:
        return "Yes, for sure!"
    elif number == 1:
        return "Probably yes."
    elif number == 2:
        return "Seems like yes..."
    elif number == 3:
        return "Definitely not!"
    elif number == 4:
        return "Probably not."
```

```
    elif number == 5:
        return "I really doubt it..."
    elif number == 6:
        return "Not sure, check back later!"
    elif number == 7:
        return "I really can't tell."
    else:
        return "Something was wrong with my input."


def mystical_octosphere(question):

    print("Your question was... ",question)

    answer_number = random.randrange(0, 8)

    answer_fortune = number_to_fortune(answer_number)

    print("The mystical octosphere says... " , answer_fortune)

question = input('Enter your question: ')
mystical_octosphere(question)
```

```
Enter your question: Will india win the match?
Your question was...  Will india win the match?
The mystical octosphere says...  Not sure, check back later!
```

Rock Paper Scissors Game



In [136…

```
import random

def num_to_name(num):
    if num==0:
        return "rock"
    elif num==1:
        return "paper"
    elif num==2:
        return "scissors"

def name_to_num(name):
    if name.lower()=="rock":
        return 0
    elif name.lower()=="paper":
        return 1
    elif name.lower()=="scissors":
```

```
        return 2

def play():
    score=0
    cscore=0

    round=int(input("Enter How Many Round you want to Play:"))
    print()
    for i in range(round):
        x=input("Enter Your Choice: ")
        user=name_to_num(x)
        comp=random.randint(0,2)
        y=num_to_name(comp)
        print("Choice of Computer:",y)

        if user==comp:
            print("It's Tie")

        elif user==0 and comp==1:
            print("Computer Wins")
            cscore+=1

        elif user==0 and comp==2:
            print("User Wins")
            score+=1

        elif user==1 and comp==0:
            print("User Wins")
            score+=1

        elif user==1 and comp==2:
            print("Computer Wins")
            cscore+=1

        elif user==2 and comp==0:
            print("Computer Wins")
            cscore+=1

        elif user==2 and comp==1:
            print("User Wins")
            score+=1
        print()
    print("Player Final Score:",score)
    print("Computer Final Score:",cscore)

play()
```

```
Enter How Many Round you want to Play:3

Enter Your Choice: rock
Choice of Computer: rock
It's Tie

Enter Your Choice: paper
Choice of Computer: scissors
Computer Wins

Enter Your Choice: scissors
Choice of Computer: paper
User Wins
```

```
Player Final Score: 1
Computer Final Score: 1
```

Logic-2

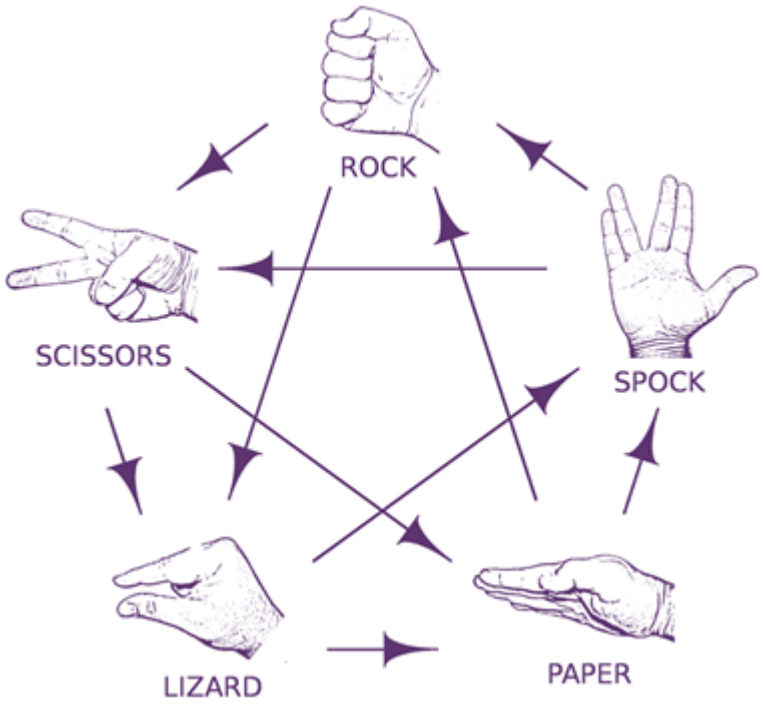
In [140…

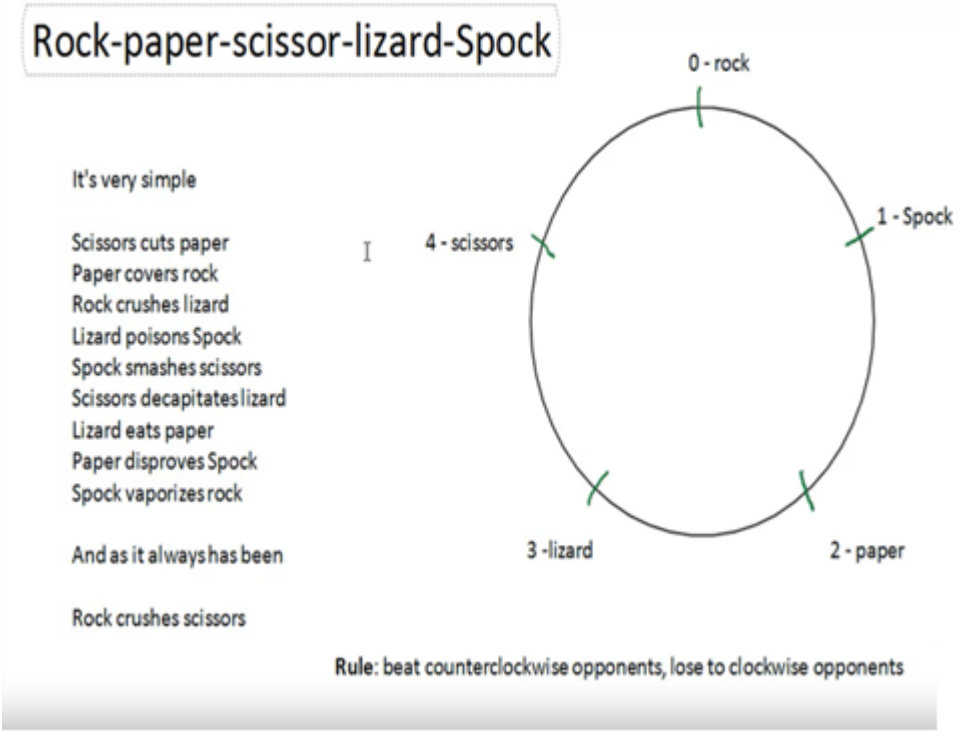
```python
import random

def number_to_name(number):
    if number==0:
        return 'rock'
    elif number==1:
        return 'paper'
    elif number==2:
        return 'scissor'

def name_to_number(user_input):
    if user_input=='rock':
        return 0
    elif user_input=='paper':
        return 1
    elif user_input=='scissor':
        return 2

def rps(player_choice):    # rock ,paper,scissor
    global score,cscore
    print()
    print('Player chooses',player_choice)
    player_number=name_to_number(player_choice)
    computer_number=random.randrange(0,3)
    computer_choice=number_to_name(computer_number)
    print('computer chooses',computer_choice)

    difference=(player_number-computer_number)%3

    if difference==1:
        print("Player wins!")
        score+=1

    elif difference==2:
        print("Computer wins!")
        cscore+=1
    else:
        print("It's a Tie..!!")

n=int(input('How many times do you want to play?'))
score=0
cscore=0
for i in range(n):
    player_choice=input("enter your choice: ")
    rps(player_choice)
print("Your score is ",score)
print("Computer Score is:",cscore)
```

```
How many times do you want to play?3
enter your choice: rock

Player chooses rock
computer chooses paper
Computer wins!
enter your choice: paper
```

```
Player chooses paper
computer chooses rock
Player wins!
enter your choice: scissor

Player chooses scissor
computer chooses scissor
It's a Tie..!!
Your score is  1
Computer Score is: 1
```

Rock Paper Scissors Lizard Spock Game





In [142…

```
import random

# The key idea of this program is to equate the strings
# "rock", "paper", "scissors", "lizard", "Spock" to numbers
# as follows:
#
# 0 - rock
# 1 - Spock
# 2 - paper
# 3 - lizard
# 4 - scissors

#assign name to given number
def num_to_name(num):
    if num==0:
        return "rock"
    elif num==1:
        return "spock"
    elif num==2:
        return "paper"
    elif num==3:
        return "lizard"
    elif num==4:
        return "scissors"

#assign number to given name
def name_to_num(name):
    if name.lower()=="rock":
        return 0
    elif name.lower()=="spock":
        return 1
    elif name.lower()=="paper":
        return 2
    elif name.lower()=="lizard":
        return 3
    elif name.lower()=="scissors":
        return 4

def play():
    user_score=0
    comp_score=0
    round=int(input("Enter How Many Round you want to Play:"))

    for i in range(round):
        print()
        x=input("Enter Your Choice: ")
        # convert the player's choice to user using the function name_to_number()
        user=name_to_num(x)

        # compute random guess for comp using random.randint()
        comp=random.randint(0,4)

        # convert comp to y using the function number_to_name()
        y=num_to_name(comp)

        # print out the message for computer's choice
        print("Choice of Computer:",y)

        difference=(user-comp)%5         #Modular Arithmetic
```

```
        if difference==1 or difference==2:
            print("User Wins")
            user_score+=1

        elif difference==3 or difference==4:
            print("Computer Wins")
            comp_score+=1

        else:
            print("It's a Tie")

    print()
    print("Final User Score:",user_score)
    print("Final Computer Score:",comp_score)

    if(user_score>comp_score):
        print("User Wins")
    elif(user_score<comp_score):
        print("Computer Wins")
    else:
        print("It's a Tie!!!!!")
play()
```

```
Enter How Many Round you want to Play:3

Enter Your Choice: rock
Choice of Computer: paper
Computer Wins

Enter Your Choice: paper
Choice of Computer: paper
It's a Tie

Enter Your Choice: scissors
Choice of Computer: lizard
User Wins

Final User Score: 1
Final Computer Score: 1
It's a Tie!!!!!
```

Number Guessing Game

In [143…

```
import random

def number_guessing_game():
    secret_number = random.randint(1, 100)
    attempts = 0

    print("Welcome to the Number Guessing Game!")
    print("I'm thinking of a number between 1 and 100.")

    while True:
        guess = int(input("Take a guess: "))
        attempts += 1

        if guess < secret_number:
            print("Too low! Try a higher number.")
```

```
        elif guess > secret_number:
            print("Too high! Try a lower number.")
        else:
            print(f"Congratulations! You guessed the number in {attempts} attempts.")
            break

number_guessing_game()
```

```
Welcome to the Number Guessing Game!
I'm thinking of a number between 1 and 100.
Take a guess: 50
Too high! Try a lower number.
Take a guess: 25
Too high! Try a lower number.
Take a guess: 15
Too low! Try a higher number.
Take a guess: 20
Too low! Try a higher number.
Take a guess: 22
Too low! Try a higher number.
Take a guess: 24
Too high! Try a lower number.
Take a guess: 23
Congratulations! You guessed the number in 7 attempts.
```

Write a Program to shuffle a given deck of cards randomly.

In [146…

```
import random

def shuffle_deck(deck):
    random.shuffle(deck)
    return deck

suits=['Hearts','Diamonds','Clubs','Spades']
ranks=['Ace','2','3','4','5','6','7','8','9','Jack','Queen','King']

deck=[(rank,suit) for suit in suits for rank in ranks]
print(deck)
print()

shuffled_deck=shuffle_deck(deck)
print(shuffled_deck)
print()

print("shuffled Deck:")
for card in shuffled_deck:
    print(card[0],"of",card[1])
```

```
[('Ace', 'Hearts'), ('2', 'Hearts'), ('3', 'Hearts'), ('4', 'Hearts'), ('5', 'Hearts'),
('6', 'Hearts'), ('7', 'Hearts'), ('8', 'Hearts'), ('9', 'Hearts'), ('Jack', 'Hearts'),
('Queen', 'Hearts'), ('King', 'Hearts'), ('Ace', 'Diamonds'), ('2', 'Diamonds'), ('3',
'Diamonds'), ('4', 'Diamonds'), ('5', 'Diamonds'), ('6', 'Diamonds'), ('7', 'Diamonds'),
('8', 'Diamonds'), ('9', 'Diamonds'), ('Jack', 'Diamonds'), ('Queen', 'Diamonds'), ('Kin
g', 'Diamonds'), ('Ace', 'Clubs'), ('2', 'Clubs'), ('3', 'Clubs'), ('4', 'Clubs'), ('5',
'Clubs'), ('6', 'Clubs'), ('7', 'Clubs'), ('8', 'Clubs'), ('9', 'Clubs'), ('Jack', 'Club
s'), ('Queen', 'Clubs'), ('King', 'Clubs'), ('Ace', 'Spades'), ('2', 'Spades'), ('3', 'S
pades'), ('4', 'Spades'), ('5', 'Spades'), ('6', 'Spades'), ('7', 'Spades'), ('8', 'Spad
es'), ('9', 'Spades'), ('Jack', 'Spades'), ('Queen', 'Spades'), ('King', 'Spades')]

[('Ace', 'Diamonds'), ('Jack', 'Clubs'), ('Jack', 'Hearts'), ('Ace', 'Spades'), ('Jack',
'Diamonds'), ('9', 'Diamonds'), ('7', 'Diamonds'), ('9', 'Clubs'), ('5', 'Diamonds'),
('6', 'Hearts'), ('King', 'Clubs'), ('5', 'Spades'), ('7', 'Hearts'), ('7', 'Spades'),
('8', 'Hearts'), ('8', 'Diamonds'), ('4', 'Diamonds'), ('6', 'Spades'), ('7', 'Clubs'),
```

```
('Jack', 'Spades'), ('King', 'Diamonds'), ('King', 'Hearts'), ('4', 'Clubs'), ('3', 'Hea
rts'), ('6', 'Diamonds'), ('3', 'Diamonds'), ('Ace', 'Hearts'), ('2', 'Hearts'), ('4',
'Spades'), ('2', 'Diamonds'), ('4', 'Hearts'), ('5', 'Clubs'), ('9', 'Spades'), ('8', 'S
pades'), ('Ace', 'Clubs'), ('King', 'Spades'), ('Queen', 'Diamonds'), ('2', 'Spades'),
('Queen', 'Clubs'), ('3', 'Clubs'), ('9', 'Hearts'), ('2', 'Clubs'), ('6', 'Clubs'), ('Q
ueen', 'Spades'), ('Queen', 'Hearts'), ('5', 'Hearts'), ('8', 'Clubs'), ('3', 'Spades')]

shuffled Deck:
Ace of Diamonds
Jack of Clubs
Jack of Hearts
Ace of Spades
Jack of Diamonds
9 of Diamonds
7 of Diamonds
9 of Clubs
5 of Diamonds
6 of Hearts
King of Clubs
5 of Spades
7 of Hearts
7 of Spades
8 of Hearts
8 of Diamonds
4 of Diamonds
6 of Spades
7 of Clubs
Jack of Spades
King of Diamonds
King of Hearts
4 of Clubs
3 of Hearts
6 of Diamonds
3 of Diamonds
Ace of Hearts
2 of Hearts
4 of Spades
2 of Diamonds
4 of Hearts
5 of Clubs
9 of Spades
8 of Spades
Ace of Clubs
King of Spades
Queen of Diamonds
2 of Spades
Queen of Clubs
3 of Clubs
9 of Hearts
2 of Clubs
6 of Clubs
Queen of Spades
Queen of Hearts
5 of Hearts
8 of Clubs
3 of Spades
```

Write a program to generate a Random Sentence using articles,nouns,verbs and adverbs.

In [149…

```python
import random

def generate_sequence():
    articles=['the','a','an']
    nouns=['cat','dog','car','apple','book']
```

```
    verbs=['runs','jumps','eats','reads','drives']
    adverbs=['quickly','slowly','happilly','loudly','carefully']

    article=random.choice(articles)
    noun=random.choice(nouns)
    verb=random.choice(verbs)
    adverb=random.choice(adverbs)

    sentence=f"{article.capitalize()} {noun} {verb} {adverb}."
    return sentence

sentence=generate_sequence()
print(sentence)
```

```
The cat eats loudly.
```

Why SimpleGUICS2Pygame?

SimpleGUICS2Pygame is a special library created using pygame to make coding games easier to learn for beginners.

Using SimpleGUICS2Pygame, one can learn all the basics of creating games like motion, acceleration, friction, sounds, images, etc.

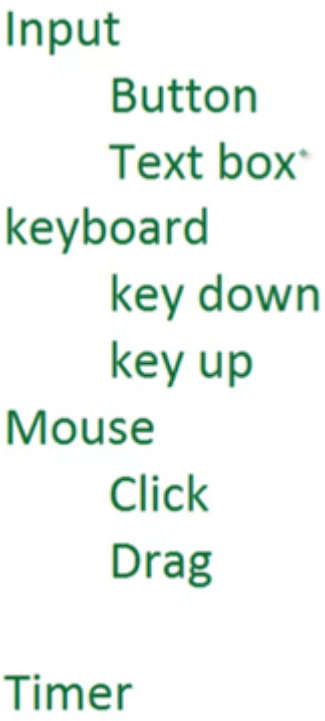
This can enable those who want to take up game development professionally in future, to apply the same knowledge in other gaming libraries.

Game Template

Event Driven Programming Model

Program waits till an event occurs. Till then it doesn't do anything.

We create handlers for every event. E.g., when one button is pushed, its handler is called and the program is executed.



The Event Queue

What happens if you press a key and click the mouse at exactly the same time?

One of the event handlers executes and the other waits in the event queue until the first handler finishes.

You can’t control the order that the system inserts events into the event queue and only one event handler executes at a time.

Program Structure

Program Structure

Globals (state)
Helper functions
Classes (later)
Define event handlers
create a frame
Register event handlers
Start frame & timers

Command to Install SimpleGUICS2Pygame

pip install SimpleGUICS2Pygame

In [3]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
```

Timers

▸ Create Timer	`simplegui.create_timer()`
▸ Start Timer	`timer.start()`
▸ Stop Timer	`timer.stop()`
▸ Check if Timer is Running	`timer.is_running()`

Create Timer

Create Timer — `simplegui.create_timer()`

Syntax: `simplegui.create_timer(interval, timer_handler)`

Example:

Code	Output

`import simplegui`
 	
`def timer_handler():`
`    …`
 	
`timer = simplegui.create_timer(500, timer_handler)`
`timer.start()`
	`# starts a timer`

Creates a timer. Once started, it will repeatedly call the given event handler at the specified interval, which is given in milliseconds.

The handler should be defined with no arguments, as in the above example.

Start Timer

Start Timer `timer.start()`

Syntax: `timer.start()`

Example:

Code	Output

`import simplegui`
 	
`def timer_handler():`
`    …`
 	
`timer = simplegui.create_timer(500, timer_handler)`
`timer.start()`
	`# starts a timer`

Starts or restarts the timer.

In [1]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

#globals

#helper functions

#classes

#event handlers
def tick():
  print('Hello')

#create a frame

#register event handlers
timer = simplegui.create_timer(1000, tick)

#start frame and timers
timer.start()
```

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

In [1]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
def timer_handler():
    print("tick")

timer=simplegui.create_timer(500,timer_handler)
timer.start()
```

```
tick
tick
tick
tick
tick
tick
```

```
tick
tick
```

Stop Timer

Stop Timer `timer.stop()`

Syntax: `timer.stop()`

Example:

Code	Output

`import simplegui`
 	
`def timer_handler():`
`  timer.stop()`
 	
`timer = simplegui.create_timer(500, timer_handler)`
`timer.start()`
	`# starts a timer and stops it on its first tick`

Stops the timer. It can be restarted.

In [1]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

#globals
t=0

#helper functions

#classes

#event handlers
def tick():
    global t
    t+=1
    if t == 5:
        timer.stop()
    print('tick')
#create a frame

#register event handlers
timer = simplegui.create_timer(1000, tick)

#start frame and timers
timer.start()
```

```
tick
tick
tick
tick
tick
```

In [1]:
```
#print tick 5 times
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
t=0
def timer_handler():
    global t
    while t==5:
        timer.stop()
    print("tick")
    t+=1
timer=simplegui.create_timer(500,timer_handler)
timer.start()
```

```
tick
tick
tick
tick
tick
```

In [1]:
```
#Print tick only once
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
def timer_handler():
    print("tick")
    timer.stop()
timer=simplegui.create_timer(500,timer_handler)
timer.start()
```

```
tick
```

Check if Timer is Running

Check if Timer is Running `timer.is_running()`

Syntax: `timer.is_running()`

Example:

Code	Output

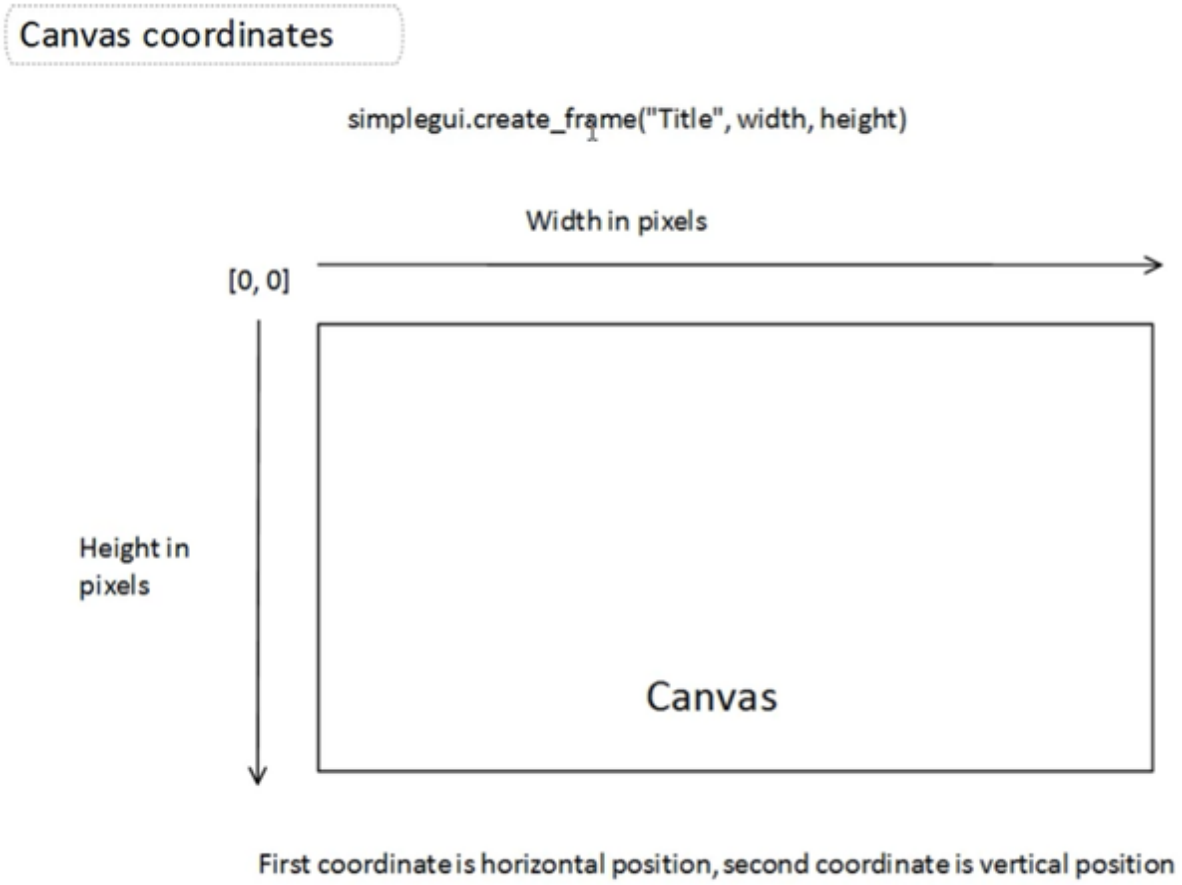
`import simplegui`
 	
`def timer_handler():`
`  pass`
 	
`timer = simplegui.create_timer(100, timer_handler)`
`print timer.is_running()`
`timer.start()`
`print timer.is_running()`
`timer.stop()`
`print timer.is_running()`
	`False`
	`True`
	`False`

Returns whether the timer is running, i.e., it has been started, but not stopped.

In [2]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

def timer_handler():
    pass
timer=simplegui.create_timer(100,timer_handler)
print(timer.is_running()) #it checks whether timer is started
timer.start()
print(timer.is_running()) #it checks after timer is started
timer.stop()
print(timer.is_running()) #it checks after timer is stopped
```

```
False
True
False
```

Frames



▸ Create Frame	`simplegui.create_frame()`
▸ Set Canvas' Background Color	`frame.set_canvas_background()`
▸ Start Frame's Interactivity	`frame.start()`
▸ Get Canvas Text's Width	`frame.get_canvas_textwidth()`

Create Frame

A frame is a window, which is a container for the controls, status information, and canvas. A program can create only one frame.

Creates a new frame for interactive programs. The frame's window has the given title, a string.

The frame consists of two parts: a control panel on the left and a canvas on the right. The control panel's width in pixels can be specified by the number control_width.

The canvas width in pixels is the number canvas_width. The height in pixels of both the control panel and canvas is the number canvas_height.

Create Frame `simplegui.create_frame()`

Syntax: `simplegui.create_frame(title, canvas_width, canvas_height)`

`simplegui.create_frame(title, canvas_width, canvas_height, control_width)`

Examples:

Code	Output

`import simplegui`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
	`# Opens frame`

`import simplegui`
 	
`frame = simplegui.create_frame('Testing', 200, 200, 300)`
	`# Opens frame`

In [4]:
```
#It opens in a new window
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
frame=simplegui.create_frame('Testing',150,150)
frame.start()
```

In [3]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
frame=simplegui.create_frame('Testing1',150,150,200)
frame.start()
```

Set the Draw Handler on Canvas `frame.set_draw_handler()`

Syntax: `frame.set_draw_handler(draw_handler)`

Example:

Code	Output

`import simplegui`
 	
`def draw_handler(canvas):`
`    …`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`frame.set_draw_handler(draw_handler)`
`frame.start()`
	`# Opens frame with active draw handler`

Adds an event handler that is responsible for all drawing.

The handler should be defined with one parameter, as in the above example. This parameter will receive a canvas object.

Set Canvas Background

syntax: frame.set_canvas_background(color)

▼ **Set Canvas' Background Color** `frame.set_canvas_background()`

Syntax: `frame.set_canvas_background(color)`

In [5]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

def draw_handler(canvas):
    frame.set_canvas_background('red')

frame=simplegui.create_frame('Testing',150,150,40)
frame.set_draw_handler(draw_handler)
frame.start()
```

In [6]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

def draw_handler(canvas):
    frame.set_canvas_background('blue')

frame=simplegui.create_frame("Testing",400,400,200)
frame.set_draw_handler(draw_handler)
frame.start()
```

Start the Frame

Start Frame's Interactivity `frame.start()`

Syntax: `frame.start()`

Example:

Code	Output

`import simplegui`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`frame.start()`
	`# Opens frame`

Starts all frame event handlers — drawing and controls.

Draw Text on Canvas

Given a text string, a font size, and a font face, this returns the width of the text in pixels. It does not draw the text. This is useful in computing the position to draw text when you want it centered or right justified in some region.

The supported font faces are the default "serif", "sans-serif", and "monospace".

Syntax: `canvas.draw_text(text, point, font_size, font_color)`

`canvas.draw_text(text, point, font_size, font_color, font_face)`

Example:

Code	Output

`import simplegui`
 	
`def draw_handler(canvas):`
`  canvas.draw_text('A', (20, 20), 12, 'Red')`
`  canvas.draw_text('B', [30, 50], 20, 'Blue')`
`  canvas.draw_text('C', (80, 50), 12, 'Gray', 'serif')`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`frame.set_draw_handler(draw_handler)`
`frame.start()`
	`# Opens frame and draws three letters`

syntax: canvas.draw_text(text,location,size of text,color of text,face of text)

In [1]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
def draw_handler(canvas):
    canvas.draw_text('A',(20,20),12,"Red")
    canvas.draw_text('B',[40,40],20,"Blue")
    canvas.draw_text('C',[80,80],12,"Gray",'serif')
    frame.set_canvas_background("Black")

frame=simplegui.create_frame("Testing",400,400,200)
```

```
frame.set_draw_handler(draw_handler)
frame.start()
```

In [2]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
def draw_handler(canvas):
    canvas.draw_text('A',(100,50),12,'Magenta')
    canvas.draw_text('B',(30,50),20,'Blue')
    canvas.draw_text('C',(80,50),12,'Gray','serif')
    frame.set_canvas_background('yellow')
frame=simplegui.create_frame('Testing',150,150,40)
frame.set_draw_handler(draw_handler)
frame.start()
```

In [3]:
```
#Event Driven Programming
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

#Global
mycolors=['Red','Black','Blue',"Green","Yellow","Orange","Gray"]
t=0

#Event Handlers
def timer_handler():  #Timer Handler Function
    global t
    t+=1
    if t==7:
        t=0

def draw_handler(canvas): # Draw Handler Function
    frame.set_canvas_background(mycolors[t])

#Frame and Timer
timer=simplegui.create_timer(1000,timer_handler)
frame=simplegui.create_frame("Testing",400,400,200)

#Register Event Handlers
frame.set_draw_handler(draw_handler)

#Start
timer.start()
frame.start()
```

▸ **Set the Draw Handler on Canvas**	`frame.set_draw_handler()`
▸ **Draw Text on Canvas**	`canvas.draw_text()`
▸ **Draw Line Segment on Canvas**	`canvas.draw_line()`
▸ **Draw Connected Line Segments on Canvas**	`canvas.draw_polyline()`
▸ **Draw Polygon on Canvas**	`canvas.draw_polygon()`
▸ **Draw Circle on Canvas**	`canvas.draw_circle()`
▸ **Draw Point on Canvas**	`canvas.draw_point()`
▸ **Draw Image on Canvas**	`canvas.draw_image()`

Control Objects

▸ Add Text Label to Frame Control Panel	`frame.add_label()`
▸ Add Button to Frame Control Panel	`frame.add_button()`
▸ Add Text Input Box to Frame Control Panel	`frame.add_input()`
▸ Get the Text of Control Object	`control.get_text()`
▸ Set the Text of Control Object	`control.set_text()`
▸ Set the Keyboard Input Handler	`frame.set_keydown_handler()`, `frame.set_keyup_handler()`
▸ Set the Mouse Input Handler	`frame.set_mouseclick_handler()`, `frame.set_mousedrag_handler()`

Drawing on Canvas

Drawing Lines on Canvas

Draws a line segment between the two points, each of which is a 2-element tuple or list of screen coordinates.

The line's width is given in pixels and must be positive.

Syntax: `canvas.draw_line(point1, point2, line_width, line_color)`

Example:

Code	Output

`import simplegui`
 	
`def draw_handler(canvas):`
`  canvas.draw_line((10, 20), (30, 40), 12, 'Red')`
`  canvas.draw_line([10, 20], [80, 70], 20, 'Blue')`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`frame.set_draw_handler(draw_handler)`
`frame.start()`
	`# Opens frame and draws two lines`

In [4]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

def draw_handler(canvas):
    canvas.draw_line((10,20),(60,70),12,"red")
frame=simplegui.create_frame('test',200,200)
frame.set_draw_handler(draw_handler)
frame.start()
```

In [5]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

def draw_handler(canvas):
    canvas.draw_line((10,20),(30,60),12,"Red")
    canvas.draw_line([50,150],[180,280],15,"Blue")

frame=simplegui.create_frame("Line",400,300)
frame.set_draw_handler(draw_handler)
frame.start()
```

Drawing Connected Lines on Canvas

Draws a sequence of line segments between each adjacent pair of points in the non-empty list.

It is an error for the list of points to be empty.

Each point is a 2-element tuple or list of screen coordinates.

The line's width is given in pixels and must be positive

Syntax: `canvas.draw_polyline(point_list, line_width, line_color)`

Example:

Code	Output

`import simplegui`
 	
`def draw_handler(canvas):`
`  canvas.draw_polyline([(10, 20), (30, 20), (90, 70)], 12, 'Red')`
`  canvas.draw_polyline([[40, 20], [80, 40], [30, 90]], 20, 'Blue')`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`frame.set_draw_handler(draw_handler)`
`frame.start()`
	`# Opens frame and draws two polylines`

In [7]:
```
#Drawing polygon with polyline to connect last and first point by just connecting
#Last and first
#Drawing connected lines on canvas
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
def draw_handler(canvas):
    canvas.draw_polyline([(10, 20), (30, 20), (90, 70),(10,20)], 5, 'Red')
    canvas.draw_polyline([[40, 20], [80, 40], [30, 90],[40,20]], 6, 'Blue')
#It can pass tuple or list
frame = simplegui.create_frame('Testing', 200, 200)
frame.set_draw_handler(draw_handler)
frame.start()
```

In [9]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

def draw_handler(canvas):
    canvas.draw_polyline([(10,20),(30,60),(90,120)],12,"Red")
    canvas.draw_polyline([(150,50),(180,90),(200,150)],20,"Blue")
frame=simplegui.create_frame("Polyline",400,300)
frame.set_draw_handler(draw_handler)
frame.start()
```

Drawing a Polygon on Canvas

Draws a sequence of line segments between each adjacent pair of points in the non-empty list, plus a line segment between the first and last points.

It is an error for the list of points to be empty.

Each point is a 2-element tuple or list of screen coordinates. The line's width is given in pixels, and must be positive.

The fill color defaults to None. If the fill color is specified, then the interior of the polygon is colored.

Syntax:

```
canvas.draw_polygon(point_list, line_width, line_color)
canvas.draw_polygon(point_list, line_width, line_color, fill_color = color)
```

Example:

Code	Output

`import simplegui`
 	
`def draw_handler(canvas):`
`  canvas.draw_polygon([(10, 20), (20, 30), (30, 10)], 12, 'Green')`
`  canvas.draw_polygon([[30, 20], [40, 40], [50, 20], [10, 10]], 12, 'Red')`
`  canvas.draw_polygon([(50, 70), (80, 40), (30, 90)], 5, 'Blue', 'White')`
`  canvas.draw_polygon([[90, 70], [80, 40], [70, 90], [70, 70]], 12, 'Yellow', 'Orange')`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`frame.set_draw_handler(draw_handler)`
`frame.start()`
	`# Opens frame and draws four polygons`

In [19]:

```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
def draw_handler(canvas):
    canvas.draw_polygon([(100,200),(200,30),(300,10)],12,"green")
    canvas.draw_polygon([[30,20],[40,40],[50,20],[10,10]],12,"Red")
    canvas.draw_polygon([(150,70),(80,40),(30,90)],5,"Blue","white")
    canvas.draw_polygon([(290,70),(280,40),(270,90),(70,90)],12,"yellow","blue")

frame=simplegui.create_frame("Polygon",300,300)

frame.set_draw_handler(draw_handler)

frame.start()
```

Drawing a Circle on Canvas

Draws a circle at the given center point having the given radius.

The point is a 2 element tuple or list of screen coordinates.

The line's width is given in pixels and must be positive.

The fill color defaults to None. If the fill color is specified, then the interior of the circle is colored.

Syntax:

```
canvas.draw_circle(center_point, radius, line_width, line_color)
canvas.draw_circle(center_point, radius, line_width, line_color, fill_color = color)
```

Example:

Code	Output

`import simplegui`
 	
`def draw_handler(canvas):`
`  canvas.draw_circle((10, 10), 20, 12, 'Green')`
`  canvas.draw_circle([20, 30], 30, 12, 'Red')`
`  canvas.draw_circle((50, 50), 20, 5, 'Blue', 'White')`
`  canvas.draw_circle([70, 80], 30, 10, 'Yellow', 'Orange')`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`frame.set_draw_handler(draw_handler)`
`frame.start()`
	`# Opens frame and draws four circles`

See also: Colors Supported colors

In [21]:

```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
def draw_handler(canvas):
    canvas.draw_circle((50,50),20,12,"Green")
    canvas.draw_circle([100,100],30,12,"red")
    canvas.draw_circle((150,150),20,5,"Blue","White")
    canvas.draw_circle((200,200),30,10,"yellow","orange")

frame=simplegui.create_frame("Testing",400,400)
frame.set_draw_handler(draw_handler)
```

```
#Start
frame.start()
```

Drawing Point on Canvas

Draws a 1 × 1 rectangle at the given point in the given color. The point is a 2 element tuple or list of screen coordinates.

▾ Draw Point on Canvas `canvas.draw_point()`

Syntax: `canvas.draw_point(point, color)`

Example:

Code	Output

`import simplegui`
 	
`def draw_handler(canvas):`
`  canvas.draw_point((10, 10), 'Green')`
`  canvas.draw_point([20, 30], 'Red')`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`frame.set_draw_handler(draw_handler)`
`frame.start()`
	`# Opens frame and draws two points`

In [1]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

def draw_handler(canvas):
    canvas.draw_point((50,50),"Green")
    canvas.draw_point((100,100),"Red")

frame=simplegui.create_frame("Testing",400,400)
frame.set_draw_handler(draw_handler)

#Start
frame.start()
```

Draw Image on Canvas

Draw an image that was previously loaded. center_source is a pair of coordinates giving the position of the center of the image, while center_dest is a pair of screen coordinates specifying where the center of the image should be drawn on the canvas.

width_height_source is a pair of integers giving the size of the original image, while width_height_dest is a pair of integers giving the size of how the images should be drawn. The image can be rotated clockwise by rotation radians.

You can draw the whole image file or just part of it. The source information center_source and width_height_source) specifies which pixels to display. If it attempts to use any pixels outside of the actual file size, then no image will be drawn.

Specifying a different width or height in the destination than in the source will rescale the image.

Draw Image on Canvas — canvas.draw_image()

Syntax:

```
canvas.draw_image(image, center_source, width_height_source, center_dest, width_height_dest)
canvas.draw_image(image, center_source, width_height_source, center_dest, width_height_dest, rotation)
```

Examples:

Code	Output

`import simplegui`
 	
`def draw_handler(canvas):`
`  canvas.draw_image(image, (1521 / 2, 1818 / 2), (1521, 1818), (50, 50), (100, 100))`
 	
`image = simplegui.load_image('http://commondatastorage.googleapis.com/codeskulptor-assets/gutenberg.jpg')`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`frame.set_draw_handler(draw_handler)`
`frame.start()`
	`# Opens frame and draws a scaled map`

Get Image's Width — image.get_width()

Syntax: `image.get_width()`

Returns the width of the image in pixels. While the image is still loading, it returns zero.

Get Image's Height — image.get_height()

Syntax: `image.get_height()`

Returns the height of the image in pixels. While the image is still loading, it returns zero.

In [2]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

def draw_handler(canvas):
    canvas.draw_image(image,(1280/2,720/2),(1280,720),(200,200),(400,400))

image=simplegui.load_image('https://www.nasa.gov/sites/default/files/thumbnails/image/s

frame=simplegui.create_frame("Testing",400,400)
frame.set_draw_handler(draw_handler)

#Start
frame.start()
```

In [3]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

def draw_handler(canvas):
    canvas.draw_image(image,(x/2,y/2),(x,y),(200,200),(400,400))

image=simplegui._load_local_image('nasa image.jpg')
x=image.get_width()
y=image.get_height()

frame=simplegui.create_frame("Testing",400,400)
frame.set_draw_handler(draw_handler)

#Start
frame.start()
```

In [9]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

def draw_handler(canvas):
    canvas.draw_image(image,(1521/2,1818/2),(1521,1818),(200,200),(400,400))

image=simplegui._load_local_image('gutenberg.jpg')
```

```
frame=simplegui.create_frame("Testing",400,400)
frame.set_draw_handler(draw_handler)

#Start
frame.start()
```

In [10]:
```
image.get_width()
```

Out[10]: 1521

In [11]:
```
image.get_height()
```

Out[11]: 1818

Control Objects

Control objects are placed in the control panel, which is the left hand part of the frame.

They are placed top down in the order of creation.

Status information is at the bottom of the control panel.

▸ Add Button to Frame Control Panel	`frame.add_button()`
▸ Add Text Input Box to Frame Control Panel	`frame.add_input()`
▸ Set the Keyboard Input Handler	`frame.set_keydown_handler()`, `frame.set_keyup_handler()`
▸ Set the Mouse Input Handler	`frame.set_mouseclick_handler()`, `frame.set_mousedrag_handler()`

Add Label

Syntax:

```
frame.add_label(text)
frame.add_label(text, width)
```

Example:

Code	Output

`import simplegui`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`label1 = frame.add_label('My first label')`
`label2 = frame.add_label('My second label', 200)`
`label3 = frame.add_label('My third label', 20)`
	`# Opens frame with three labels`

In [22]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
frame=simplegui.create_frame("Testing",200,200)
label1=frame.add_label("My First Label")
label2=frame.add_label("My Second Label",110)
label1=frame.add_label("My Third Label",100)
frame.start()
```

Adding Button

Adds a button to the frame's control panel with the given text label.

The width of the button defaults to fit the given text, but can be specified in pixels.

If the provided width is less than that of the text, the text overflows the button.

The handler should be defined with no parameters, as in the below example.

Syntax:

```
frame.add_button(text, button_handler)
frame.add_button(text, button_handler, width)
```

Example:

Code	Output

`import simplegui`
 	
`def button_handler():`
`    ...`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`button1 = frame.add_button('Label 1', button_handler)`
`button2 = frame.add_button('Label 2', button_handler, 50)`
	`# Opens frame with two buttons`

In [1]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
def button_handler():
    pass
frame=simplegui.create_frame("Testing",300,300)
button1=frame.add_button("Label 1",button_handler)
button2=frame.add_button("Label 2",button_handler,50)

frame.start()
```

In [5]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
def timer_handler():
    print("demo")

def start():
    timer.start()

def stop():
    timer.stop()

timer=simplegui.create_timer(1000,timer_handler)
frame=simplegui.create_frame("Testing",300,300)
button1=frame.add_button("Start",start)
button2=frame.add_button("Stop",stop,50)

frame.start()
```

```
demo
demo
demo
demo
demo
```

In [12]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
x=20
y=20

def timer_handler():
    global y
    y+=30
    if(y>300):
        y=20

def draw_handler(canvas):
```

```
    global x,y
    canvas.draw_text('Demo',(x,y),25,"Red")

def start():
    timer.start()

def stop():
    timer.stop()

timer=simplegui.create_timer(1000,timer_handler)
frame=simplegui.create_frame("Testing",300,300)
button1=frame.add_button("Start",start)
button2=frame.add_button("Stop",stop,50)
frame.set_draw_handler(draw_handler)
frame.start()
```

In [4]:

```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

#globals
message = "Welcome!"

# Handler for mouse click
def click():
    global message
    message = "Good job!"

# Handler to draw on canvas
def draw(canvas):
    canvas.draw_text(message, [50,112], 36, "Red")

# Create a frame
frame = simplegui.create_frame("Home", 400, 400)

#register event handlers
frame.add_button("Click me", click)
frame.set_draw_handler(draw)

# Start the frame animation
frame.start()
```

Button with Timer

In [3]:

```
# SimpleGUI program template

# Import the module
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

# Define global variables (program state)
counter = 0

# Define "helper" functions
def increment():
    global counter
    counter = counter + 1

# Define event handler functions
def tick():
    increment()
    print(counter)
```

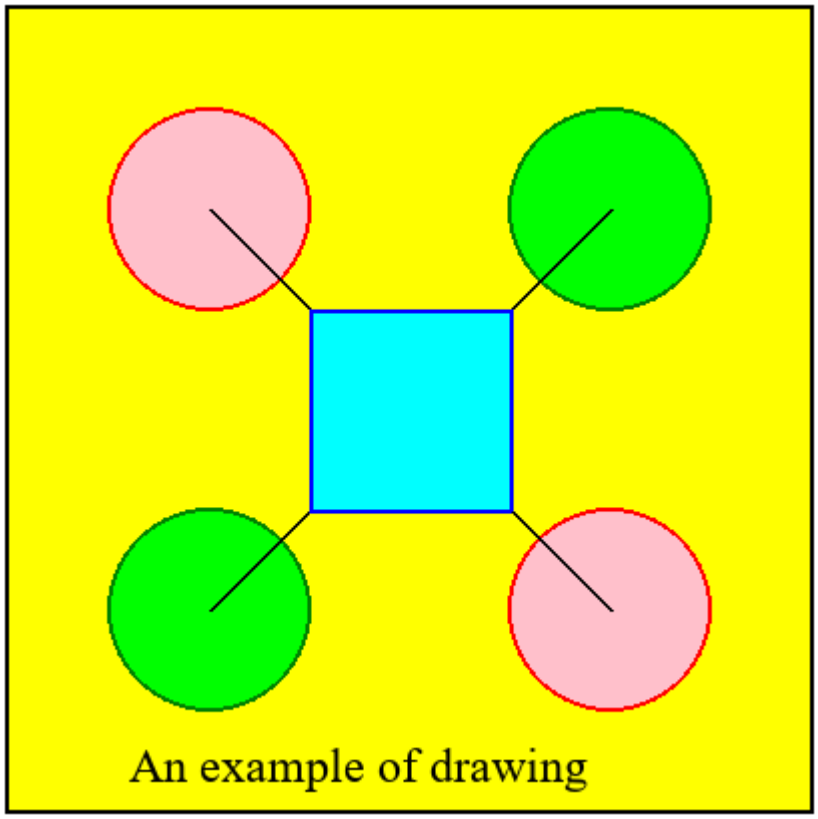
```
def buttonpress():
    global counter
    counter = 0

# Create a frame
frame = simplegui.create_frame("SimpleGUI Test", 100, 100)

# Register event handlers
timer = simplegui.create_timer(1000, tick)
frame.add_button("Click me!", buttonpress)

# Start frame and timers
timer.start()
frame.start()
```

```
1
2
3
1
2
3
```

Start button then colour change stop button then stop

In [13]:
```
#Event Driven Programming
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
#Global
mycolors=['Red','Black','Blue',"Green","Yellow","Orange","Gray"]
t=0

#Event Handlers
def timer_handler():  #Timer Handler Function
    global t
    t+=1
    if t==7:
        t=0

def draw_handler(canvas): # Draw Handler Function
    frame.set_canvas_background(mycolors[t])

def start():
    timer.start()

def stop():
    timer.stop()

#Frame and Timer creation
timer=simplegui.create_timer(1000,timer_handler)
frame=simplegui.create_frame("Testing",400,400,200)

#Register Event Handlers
frame.set_draw_handler(draw_handler)
button1=frame.add_button("Start",start,100)
button2=frame.add_button("Stop",stop,100)

#Start
frame.start()
```

Drawing Given Design



In [1]:
```
# example of drawing operations in simplegui
# standard HMTL color such as "Red" and "Green"
# note later drawing operations overwrite earlier drawing operations

import SimpleGUICS2Pygame.simpleguics2pygame as simplegui


# Handler to draw on canvas
def draw(canvas):
    canvas.draw_circle([100, 100], 50, 2, "Red", "Pink")
    canvas.draw_circle([300, 300], 50, 2, "Red", "Pink")
    canvas.draw_line([100, 100],[300, 300], 2, "Black")
    canvas.draw_circle([100, 300], 50, 2, "Green", "Lime")
    canvas.draw_circle([300, 100], 50, 2, "Green", "Lime")
    canvas.draw_line([100, 300],[300, 100], 2, "Black")
    canvas.draw_polygon([[150, 150], [250, 150], [250, 250], [150, 250]], 2,
          "Blue", "Aqua")
    canvas.draw_text("An example of drawing", [60, 385], 24, "Black")


# Create a frame and assign callbacks to event handlers
frame = simplegui.create_frame("Home", 400, 400)
frame.set_draw_handler(draw)
frame.set_canvas_background("Yellow")


# Start the frame animation
frame.start()
```

Adding Input

Adds a text input field to the control panel with the given text label. The input field has the given width in pixels.

The handler should be defined with one parameter, as in the below example.

This parameter will receive a string of the text input when the user presses the Enter key.

Syntax: `frame.add_input(text, input_handler, width)`

Example:

Code	Output

`import simplegui`
 	
`def input_handler(text_input):`
`    …`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`inp = frame.add_input('My label', input_handler, 50)`
	`# Opens frame with one input box`

In [15]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
def input_handler(text_input):
    pass

frame=simplegui.create_frame("Testing",200,200)
inp=frame.add_input("My Label",input_handler,50)
frame.start()
```

Color Changing by User Input

In [16]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
#Global
color='Red'

#Event Handlers
def draw_handler(canvas): # Draw Handler Function
    global color
    frame.set_canvas_background(color)

def input_handler(text_input):  #Input Handler Function
    global color
    color=text_input

#Frame
frame=simplegui.create_frame("Testing",400,400,200)
inp=frame.add_input("Enter Color",input_handler,50)  #input function

#Register Event Handlers
frame.set_draw_handler(draw_handler)

#Start
frame.start()
```

Calculator GUI

In [17]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

#globals
n1 = 0
n2 = 0
```

```
ans = 0

#event handlers
def input_handler1(num):
    global n1
    n1 = int(num)

def input_handler2(num):
    global n2
    n2 = int(num)

def draw(canvas):
    canvas.draw_text(str(ans), [150,150], 50, 'white')

def add():
    global ans
    ans = n1 + n2

def subtract():
    global ans
    ans = n1 - n2

def multiply():
    global ans
    ans = n1 * n2

def divide():
    global ans
    ans = n1 / n2

#create frame
frame = simplegui.create_frame('Calculator', 300, 300)

#register handlers
frame.set_draw_handler(draw)
frame.add_input('Number 1: ', input_handler1, 100)
frame.add_input('Number 2: ', input_handler2, 100)
frame.add_button('Add', add)
frame.add_button('Subtract', subtract)
frame.add_button('Multiply', multiply)
frame.add_button('Divide', divide)

#start
frame.start()
```

Buttons to Increase/Decrease Ball Radius

In [1]:

```
#import simplegui
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui


# Define globals - Constants are capitalized in Python
HEIGHT = 400
WIDTH = 400
RADIUS_INCREMENT = 5
ball_radius = 20

# Draw handler
```

```
def draw(canvas):
    canvas.draw_circle([WIDTH // 2, HEIGHT // 2], ball_radius, 1,
                       "White", "White")

# Event handler for buttons
def increase_radius():
    global ball_radius
    ball_radius += RADIUS_INCREMENT

def decrease_radius():
    global ball_radius
    if ball_radius > RADIUS_INCREMENT:
        ball_radius -= RADIUS_INCREMENT

# Create frame and assign callbacks to event handlers
frame = simplegui.create_frame("Ball control", WIDTH, HEIGHT)
frame.set_draw_handler(draw)
frame.add_button("Increase radius", increase_radius)
frame.add_button("Decrease radius", decrease_radius)


frame.start()
```

In [2]:

```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
# Define globals
radius=50

# Draw handler
def draw_handler(canvas):
    global radius
    canvas.draw_circle((200,200),radius,10,"blue","pink")

# Event handler for buttons
def inc():
    global radius
    radius+=5
    if(radius>=100):
        radius=5

def dec():
    global radius
    radius-=5
    if(radius<=5):
        radius=5

# Create frame and assign callbacks to event handlers
frame=simplegui.create_frame("Testing",400,400)
button1=frame.add_button("Inc",inc,100)
button2=frame.add_button("Dec",dec,100)
frame.set_draw_handler(draw_handler)

# Start the frame animation
frame.start()
```

Number Guessing Game

In [3]:
```
import random
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
Number=0
chances=7
x=""
win=False

def draw_handler(canvas): # Draw Handler Function
#     global n
    global x
    frame.set_canvas_background("white")
    canvas.draw_text("Chances Left:",(30,30),40,'red')
    canvas.draw_text(str(chances),(250,30),40,'red')
    canvas.draw_text(x,(60,60),20,'red')

def guess(num):
    global chances,Number,x,win

    if win==False:
        if chances!=0:
            if(int(num)>Number):
                x="Number is Lower"
            elif(int(num)<Number):
                x="Number is Higher"
            else:
                x="You Win"
                win=True
            chances=chances-1

        if(chances==0):
            x="You Lose,The Number was "+str(Number)

def range100():
    global chances,Number,x
    Number=random.randint(1,100)
    x=" "
    chances=7

def range1000():
    global chances,Number,x
    Number=random.randint(1,1000)
    chances=12
    x=" "

frame=simplegui.create_frame("Testing",400,400)
inp1=frame.add_input("Guess the No",guess,50)  #input function
button1=frame.add_button("Range 0-100",range100,100)
button2=frame.add_button("Range 0-1000",range1000,100)
frame.set_draw_handler(draw_handler)

#Start
frame.start()
```

Circle Automatic Increase Program

In [4]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
```

```
width=200
height=200
radius=1

#Timer Handler

def draw_handler(canvas):
    global radius
    canvas.draw_circle((200,200),radius,5,"blue")

def start():
    timer.start()
    global radius
    radius+=1
    if(radius==200):
        radius=1
def stop():
    timer.stop()


frame=simplegui.create_frame("Testing",400,400)
timer=simplegui.create_timer(100,start)

frame.set_draw_handler(draw_handler)
button1=frame.add_button("Start",start,100)
button2=frame.add_button("Stop",stop,100)

#Start
frame.start()
```

Screensaver Program

In [5]:

```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
import random
msg="Hello Python"
x=random.randint(1,350)
y=random.randint(1,350)
def draw_handler(canvas):
    global msg,x,y
    canvas.draw_text(msg,(x,y),30,"Lightgreen")

def input_handler(inp1):  #Input Handler Function
    global msg
    msg=inp1

def start():
    timer.start()
    global x,y
    x=random.randint(1,350)
    y=random.randint(1,350)

def stop():
     timer.stop()

frame=simplegui.create_frame("Testing",400,400)
timer=simplegui.create_timer(1000,start)

inp1=frame.add_input("Message",input_handler,50)  #input function
```

```
button1=frame.add_button("Start",start,100)
button2=frame.add_button("Stop",stop,100)
frame.set_draw_handler(draw_handler)

#Start
# timer.start()
frame.start()
```

Reaction Timer Program

In [6]:

```
# Timers
# Reaction Time

# This program tests your reaction in milliseconds using a
#       timer. Notice the uses of start() and stop() methods.

import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
import random

# Global Variables

canvas_width = 300
canvas_height = 300
reaction_time = 0

# Used to see if the circle should be drawn
started = False
count = 0
message = ' '

# Event Handlers
def check_start():
    global started, count
    count += 1
    # This makes sure that the circle won't be drawn for at
    #   least one second after starting the frame or clicking
    #   the restart button.
    if count > 10:
        # This increases the likelihood of starting and ensures
        #       that the game will start within a time limit.
        if count / 500.0 > random.random():
            started = True
            reaction_timer.start()
            circle_timer.stop()

def increment():
    global reaction_time
    reaction_time += 1

def stop_button():
    global message
    if started:
        reaction_timer.stop()
        message = "Your reaction time was " + str(reaction_time) + " ms"

def draw(canvas):
    if started:
        canvas.draw_circle([canvas_width / 2, canvas_height / 2], 60, 2, "Red", "Red")
```

```
        canvas.draw_text(message, (0, canvas_height), 20, "White")

def restart():
    global started, count, reaction_time
    started = False
    count = 0
    reaction_time = 0
    circle_timer.start()
    message = ' '


# Frame
frame = simplegui.create_frame("Reaction Time", canvas_width, canvas_height)

# Register Event Handlers
frame.set_draw_handler(draw)

# By the way, these are labels. You can check the docs for
#       more info.
label = frame.add_label("Get ready...")
frame.add_label("Press 'stop' when the red circle appears.")

frame.add_button("Stop", stop_button, 75)
frame.add_button("Restart", restart, 75)

# Note that the timers do not have the same name or purpose
circle_timer = simplegui.create_timer(100, check_start)
reaction_timer = simplegui.create_timer(1, increment)

# Start Frame and circle_timer
circle_timer.start()
frame.start()
```

In [18]:

```
#Reaction time test
#can also randomly do all import random
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
rt=0
message=''

def draw_handler(canvas):
    canvas.draw_text(message,(100,150),50,'Magenta')

def stop_button():
    global message
    reaction_display_timer.stop()
    message="Your reaction time " + str(rt)

def restart_button():
    global rt,message
    rt=0
    message=''
    reaction_display_timer.start()

def reaction_display_timer_handler():
    global rt,message
    rt=rt+1
    message=str(rt)

frame=simplegui.create_frame('Testing',800,500)
```

```
frame.set_draw_handler(draw_handler)

reaction_display_timer=simplegui.create_timer(1,reaction_display_timer_handler)

button1=frame.add_button('stop',stop_button,150)
button2=frame.add_button('restart',restart_button,150)

reaction_display_timer.start()

frame.start()
```

Keyboard Event Handlers

These add keyboard event handlers waiting for keydown, and keyup events, respectively.

When any key is pressed, the keydown handler is called once. When any key is released, the keyup handler is called once.

The handler for each should be defined with one parameter, as in the above example.

This parameter will receive an integer representing a keyboard character.

Syntax:

```
frame.set_keydown_handler(key_handler)
frame.set_keyup_handler(key_handler)
```

Example:

Code	Output

`import simplegui`
 	
`def key_handler(key):`
`    …`
 	
`frame = simplegui.create_frame('Testing', 100, 100)`
`frame.set_keydown_handler(key_handler)`
`frame.start()`
	`# Opens frame with active keydown handler`

In [1]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

def key_handler(key):
    pass

frame=simplegui.create_frame("Keyboard Event Handler",400,400)
frame.set_keydown_handler(key_handler)
frame.start()
```

Keyboard Echo

In [3]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
current_key=""

def keydown(key):
    global current_key
    current_key=chr(key)

def keyup(key):
    global current_key
    current_key=""
```

```
def draw(canvas):
    canvas.draw_text(current_key,[10,25],20,"red")

frame=simplegui.create_frame("Keyboard Event Handler",300,300)
frame.set_keydown_handler(keydown)
frame.set_keyup_handler(keyup)
frame.set_draw_handler(draw)

frame.start()
```

In [6]:
```
import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
current_key=" "

def keydown(key):
    global current_key
    current_key=chr(key)

def keyup(key):
    global current_key
    current_key=" "

def draw(canvas):
    if current_key in "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789":
        canvas.draw_text(current_key,[10,25],20,"red")


frame=simplegui.create_frame("Keyboard Event Handler",300,300)
frame.set_keydown_handler(keydown)
frame.set_keyup_handler(keyup)
frame.set_draw_handler(draw)

frame.start()
```

Shape Shifting

In [1]:
```
# Keyboard Input
# Shape Selection
# This is a simple program that allows you to change the
#       size, color, fill, and shape of an image on the screen.

import SimpleGUICS2Pygame.simpleguics2pygame as simplegui

# Global Variables
canvas_width = 200
canvas_height = 200
size = 100
colors = ["DeepPink", "Red", "DarkOrange", "Yellow", "Lime", "Green", "Blue", "Aqua", "
fill = False
shapes = ["Square", "Circle", "Triangle"]
color_index = len(colors) // 2
shape_index = len(shapes) // 2

# Event Handlers
def draw(canvas):
    if shapes[shape_index] == "Square":
        x = canvas_width / 2 - size / 2
        y = canvas_height / 2 - size / 2
```

```
        if fill:
            canvas.draw_polygon([(x, y), (x + size, y), (x + size, y + size),
                                 (x, y + size)], 5,
                                colors[color_index], colors[color_index])
        else:
            canvas.draw_polygon([(x, y), (x + size, y), (x + size, y + size),
                                 (x, y + size)], 5,
                                colors[color_index])
    elif shapes[shape_index] == "Circle":
        if fill:
            canvas.draw_circle([canvas_width / 2, canvas_height / 2],
                               size / 2, 5,
                               colors[color_index], colors[color_index])
        else:
            canvas.draw_circle([canvas_width / 2, canvas_height / 2],
                               size / 2, 5,
                               colors[color_index])
    elif shapes[shape_index] == "Triangle":
        x = canvas_width / 2 - size / 2
        y = canvas_height / 2 - size / 2
        if fill:
            canvas.draw_polygon([(x, y + size), (x + size, y + size),
                                 (x + size / 2, y)], 5,
                                colors[color_index], colors[color_index])
        else:
            canvas.draw_polygon([(x, y + size), (x + size, y + size),
                                 (x + size / 2, y)], 5,
                                colors[color_index])

def key_handler(key):
    global fill, size, color_index, shape_index
    if key == simplegui.KEY_MAP['f']:
        fill = not fill
    elif key == simplegui.KEY_MAP['s']:
        if shape_index > 0:
            shape_index -= 1
    elif key == simplegui.KEY_MAP['d']:
        if shape_index < len(shapes) - 1:
            shape_index += 1
    elif key == simplegui.KEY_MAP['z']:
        if size > 10:
            size -= 10
    elif key == simplegui.KEY_MAP['x']:
        if size < 200:
            size += 10
    elif key == simplegui.KEY_MAP['c']:
        if color_index > 0:
            color_index -= 1
    elif key == simplegui.KEY_MAP['v']:
        if color_index < len(colors) - 1:
            color_index += 1

# Frame and Timer

frame = simplegui.create_frame("Shape Selection",
                               canvas_width, canvas_height)
# This statement is necessary to declare a function to be the
#       keydown handler. Try changing the line that is commented
#       out to see the difference between a keyup handler and a
#       keydown handler.
```

```
frame.set_keydown_handler(key_handler)
#frame.set_keyup_handler(key_handler)
frame.add_label("Shape: s and d")
frame.add_label("Fill: f")
frame.add_label("Size: z and x")
frame.add_label("Color: c and v")

# Register Event Handlers
frame.set_draw_handler(draw)

# Start
frame.start()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: