**Code:-**

**while** True:

**for** event **in** pygame.event.get():

**if** event.type == pygame.QUIT:

running = **False**

* print(i) tracks mouse or keyboard etc.
* QUIT means clicking the “close” button.
* **pygame.quit()** must come last.

**Fact:-**

* Typing **“ctrl-C”** in console stops the program.
* **pygame.quit()** mustn’t be inside any loop.
* Three colour combination = 1+1+1 = 3 bytes
* Never forget **pygame.display.update()**

**Code:-**

screen = pygame.display.set\_mode((x,y))

pygame.display.set\_caption(“something”)

screen.fill(colour)

**Facts:-**

* Error code - screen.fill(255,255,255)
* Correct code – screen.fill((255,255,255))

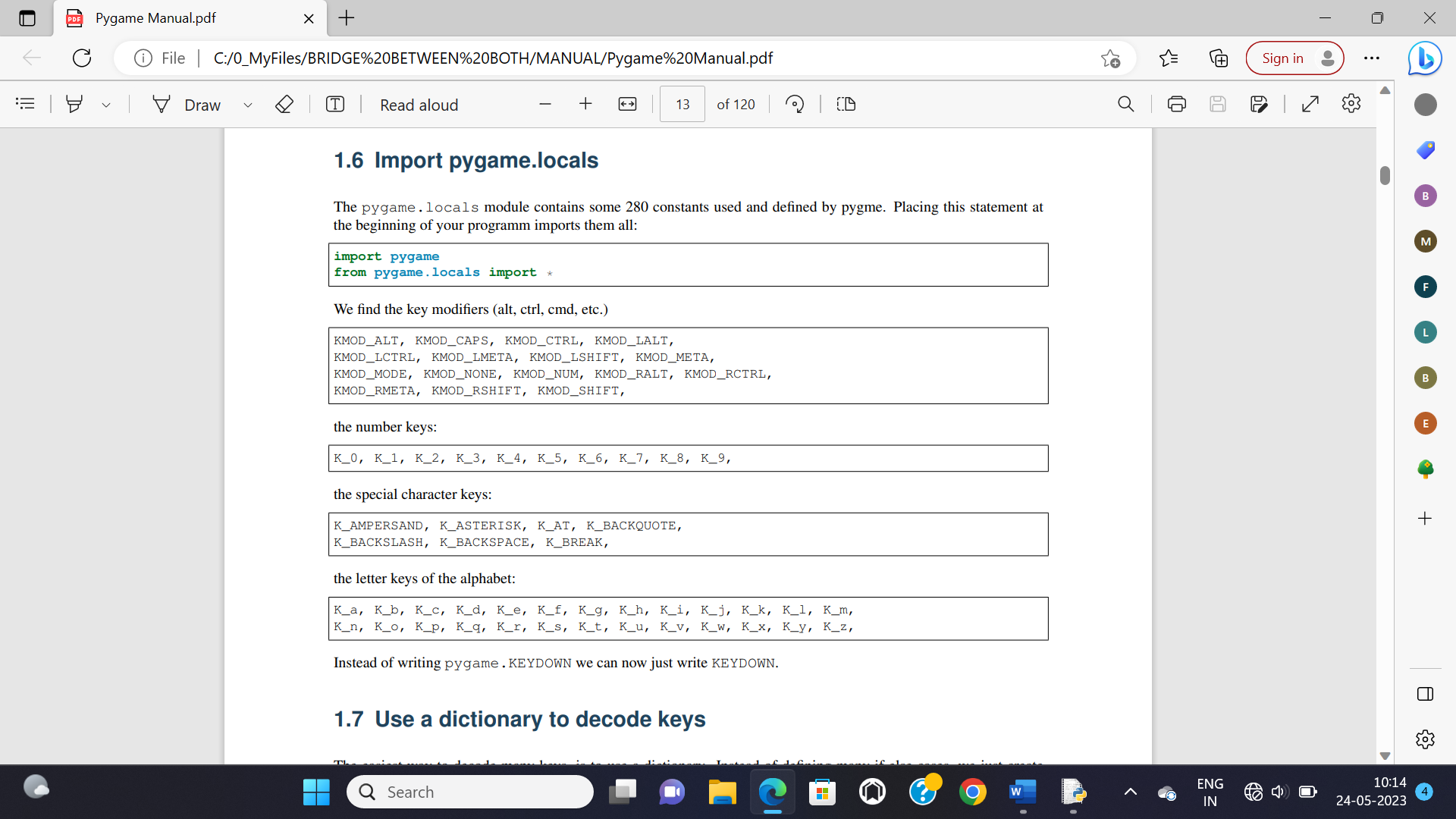
**More Code:-**

**if** event.type == pygame.KEYDOWN:

**if** event.key == pygame.K\_r:

#something

**Some Keys For Events:-**



**Drawing rectangle:-**

pygame.draw.rect(screen, colour, pygame.Rect(x\_position, y\_position, length, breadth), thickness)

* Skipping thickness will fill any shape with the choosen colour.

**Drawing shapes:-**

* **pygame.Rect** isn’t required to write if you do screen fill with any colour before.

screen.fill(COLOUR)

pygame.draw.rect(screen, GREEN, (100, 60, 120, 100))

pygame.draw.ellipse(screen, RED, (50, 20, 160, 100))

pygame.draw.circle(screen, BLUE, (100, 60), (120, 100))

pygame.draw.cirlce(screen, WHITE, (80,70), 50)

* In circle here, 50 is radius size.

**All event types:-**

1. QUIT
2. KEYDOWN
3. KEYUP
4. MOUSEBUTTONDOWN
5. MOUSEBUTTONUP
6. MOUSEMOTION

**Fact:-**

* (x,y) calculation and calculation and insertion may generate an error.
* (size) is least probable to raise an error.
* Many errors are resolved changing the way to represent coordinates.

**Important Reverse Engineering Observation:-**

* The **screen.fill(COLOUR)** thing must be the first line inside while event loop.
* To enable objects to move, write object mechanics as 3rd last thing inside while event loop.
* 2nd last thing inside **while** loop is **screen.blit code**.
* Last thing inside event while loop is **pygame.display.update()**.
* Only thing outside the while loop is **pygame.quit()**.
* Writing **pygame.display.update()** once immediately before starting **event while loop**, and another one **at last inside** event while loop is enough.

**Rectangle Operations:-**

* **r0.clip(r1)** returns rectangle in intersection region of rectangle **r0** and **r1**.
* **r0.union(r1)** returns rectangle in union region of rectangle **r0** and **r1**.
* **rect.collidepoint(pos)** returns **True** if point **pos** collides with object **rect**.
* Same way, **colliderect** is used to see if there is collision between rectangles.

**Rotation & Scaling:-**

img = pygame.transform.rotozoom(img,angle,scale)

**Flipping Image:-**

img = pygame.transform.flip(img,**True**,**False**)

* Paranthesis argument 1: choosen image
* Paranthesis argument 2: Flip horizontally?
* Paranthesis argument 3: Flip vertically?

**Detect Image Boundaries:-**

img = pygame.transform.laplacian(img)

**Working With Texts:-**

fonter = pygame.font.SysFont(None,24)

* Assigning a font with a particular size.

fonts = pygame.font.get\_fonts()

* Returns all installed fonts to the console.

img = fonter.render(“hello”,True,BLUE)

**Musics & Sounds:-**

**from** pygame **import** mixer

mixer.music.load(**“music.mp3”**)

mixer.music.set\_volume(0.7)

mixer.music.play()

mixer.music.pause()

mixer.music.unpause()

mixer.music.stop()

pygame.mixer.Sound(**“crash.wav”**)

* Put **-1** into paranthesis for playing in loop.

**Time Manipulation:-**

pygame.time.get\_tick()

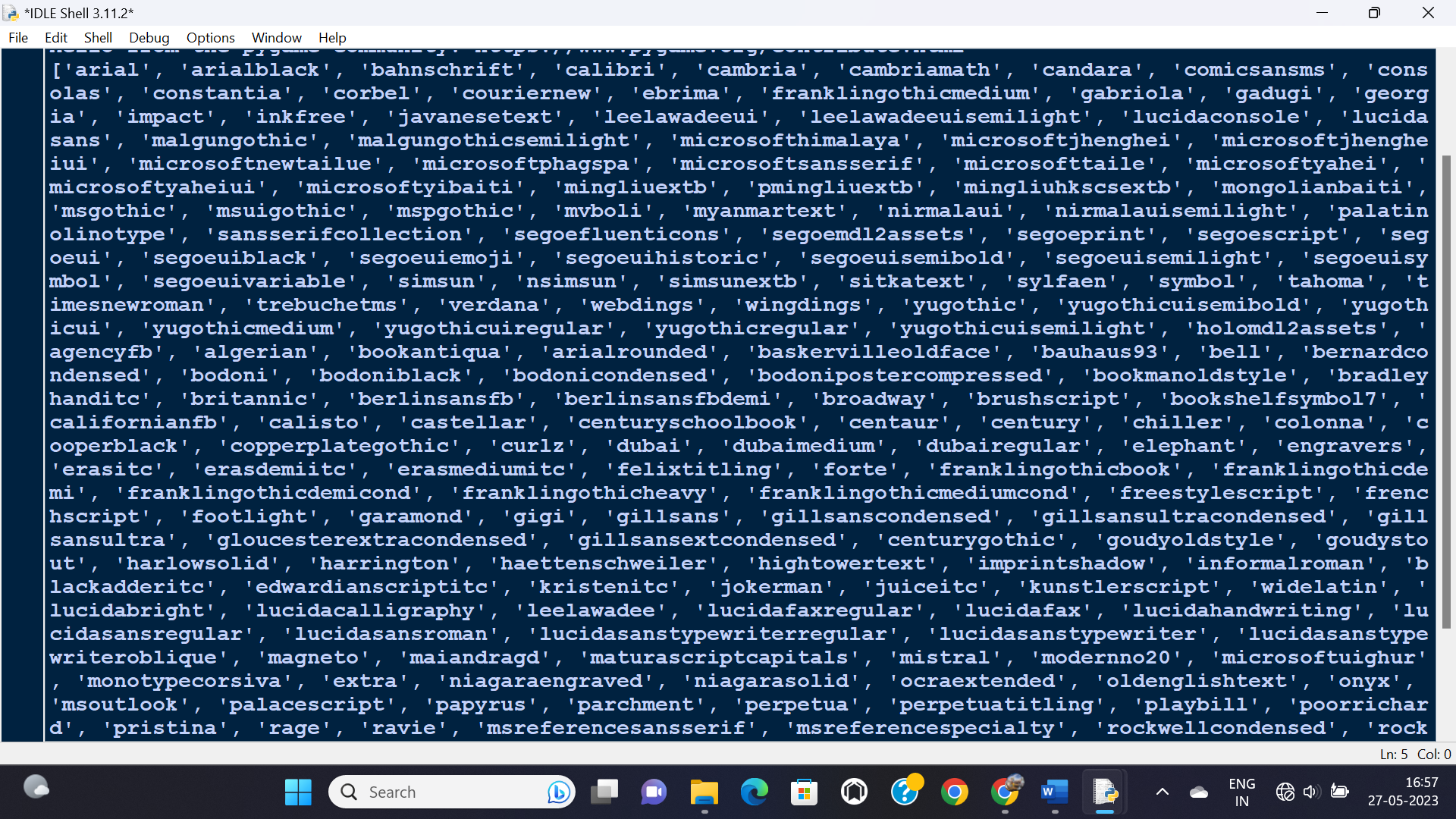
pygame.time.wait()

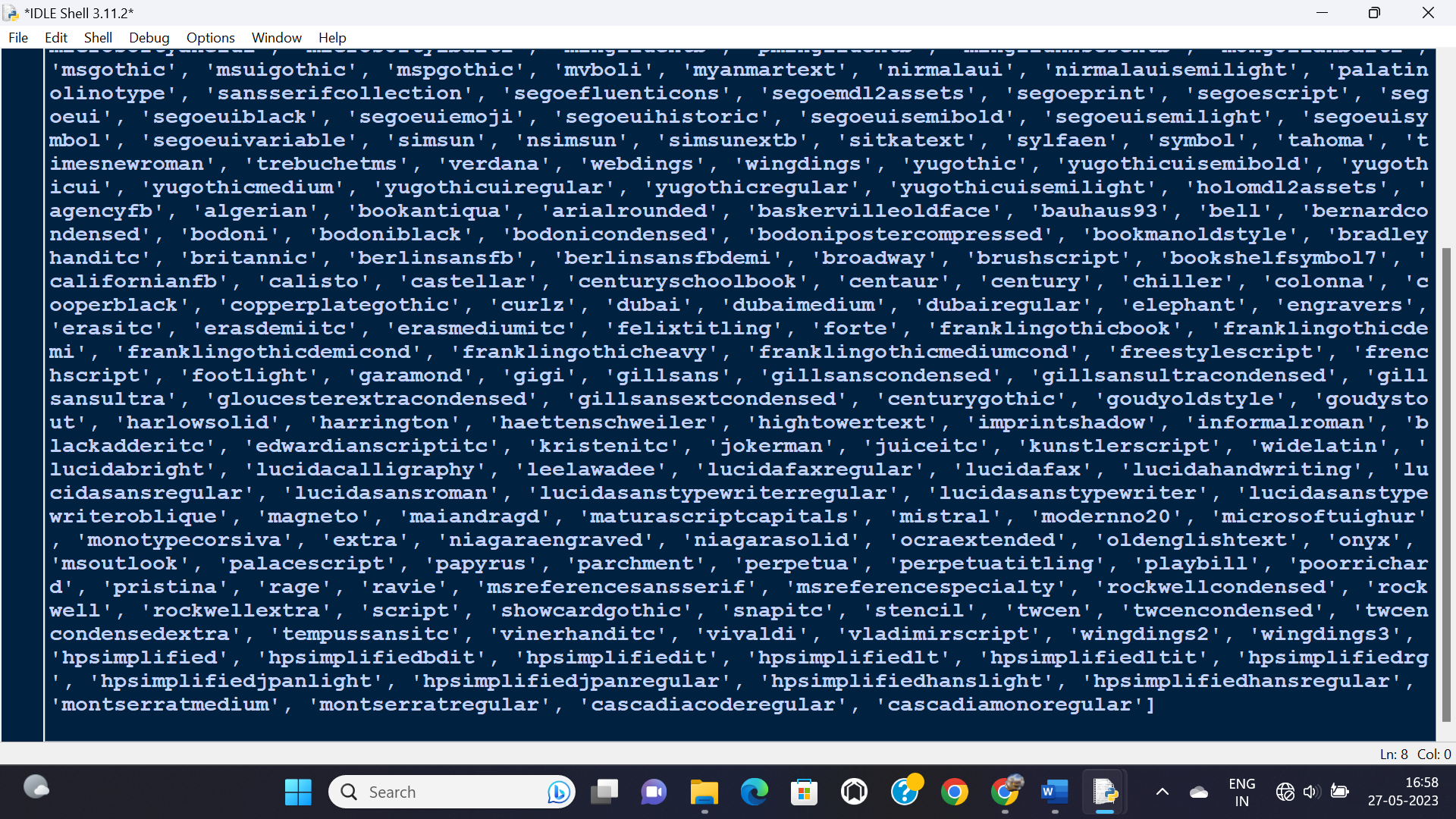
pygame.time.delay()

pygame.time.Clock.tick()

* Updates the clock.

**Available Fonts:-**





**Boosting Up Pygame Performance:-**

* Use **.convert()** or **.convert\_alpha()** for all images in pygame.
* Upgrade to pygame 2.0.
* Remove unrequired images/sprites/objects to prevent memory leakage.
* Use **pygame.mixer.pre\_init(44100,16,2,4096)** before **pygame.init()**.
* Don’t render/redraw something offscreen in while loop specially.
* Try keeping frame rates around 30.
* Use **while(1)** instead **while True**.
* Open pygame in fullscreen mode.
* Allow limited events with **pygame.event.set\_allowed([QUIT,KEYDOWN,KEYUP])**
* Don’t call **flip()** or **update()** again and again, rather use it for only sprites which do change.
* Load modules when required, rather than loading all at the start of program.
* Prefer using built-in functions.
* Optimize your code (search “Python Profiling”).