

BTEC Level 3 Computing

R o n s

T e c h

H u b

Unit 1 - Principles of Computer Science

Event Driven Programming

Event Driven Programming

What is Event-Driven Programming?

Programs that respond to events (like clicking a button).

Wait for something to happen, then react to it.

Like a waiter responding to customer requests.

Examples: Mobile apps, video games, desktop applications.

Main Structures

R O N S
T e c h
H u b

Main Loop.

Callback Function.

Sub-Routines.



Structure: Main Loop



R O N S

Tech

hub



Constantly checks for new events.



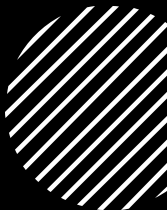
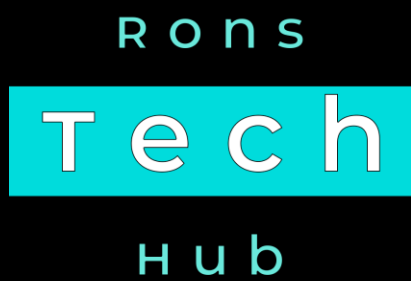
Like a security guard watching for activity.

Structure: Main Loop Example

```
while program_is_running:  
    check_for_events()  
    handle_events()
```




Structure: Callback Functions



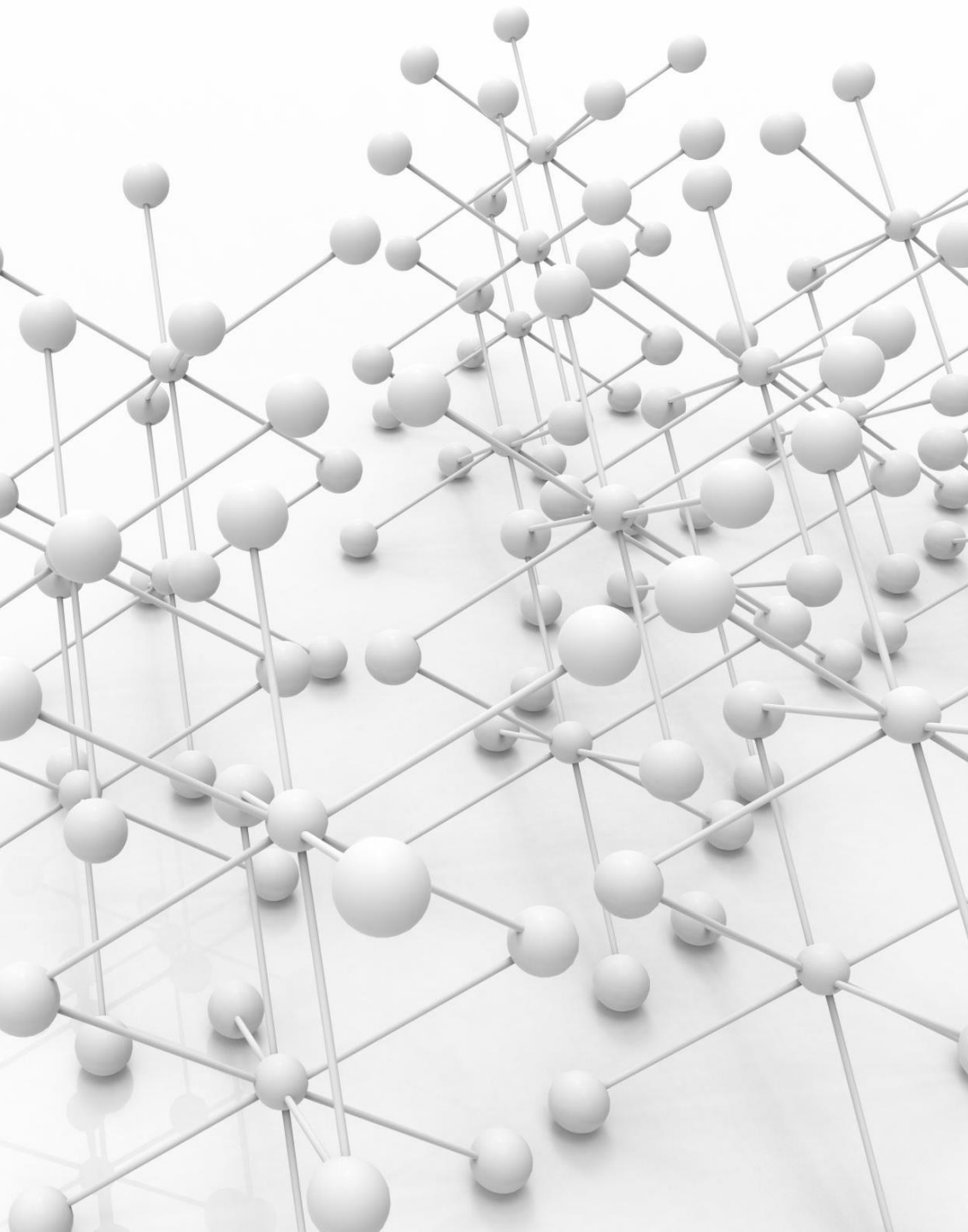
Functions that run when an event happens.



Like telling a friend "Call me back when dinner is ready."

Structure: Callback Functions Example

```
def button_clicked():  
    print("You clicked the button!")  
button.on_click(button_clicked) # Callback registration
```

Structure: Sub-routines

R O N S

T e c h

H u b

- Small tasks that support event handling.
- Break down complex actions into simple steps.

Structure: Sub-routines Examples

```
def save_user_data():  
    check_data()  
    write_to_database()  
    show_success_message()
```

R O N S

T e c h

H u b

Feature: Events

- Actions or occurrences in your program.
- Examples:
 - Mouse clicks.
 - Keyboard presses.
 - Timer completing.
 - Message received.
 - Screen touch.



R O N S

T e c h

H u b

Feature: Event Handlers

- Code that runs when an event occurs.
- Like a recipe for what to do.

Feature: Event Handlers Example

```
def handle_login_button(username, password):  
    if username and password:  
        login_user()  
    else:  
        show_error()
```




Feature: Event Loops



R O N S

Tech

Hub



Continuously checks for new events.



Distributes events to correct handlers.

Feature: Event Loops Example

```
while True:
    event = get_next_event()
    if event.type == "MOUSE_CLICK":
        handle_mouse_click()
    elif event.type == "KEY_PRESS":
        handle_key_press()
```




Feature: Service- Oriented Processing



R O N S

T e c h

H u b



Breaking program into separate services.

Each service handles specific events.

Like different departments in a company.

Example:

Print Service: Handles print requests.

Email Service: Handles email events.

Save Service: Handles save operations.



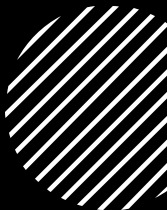
Feature: Time-Driven Events



R O N S

Tech

Hub



Events that occur at specific times.



Or after a certain delay.

Feature: Time-Driven Events Examples

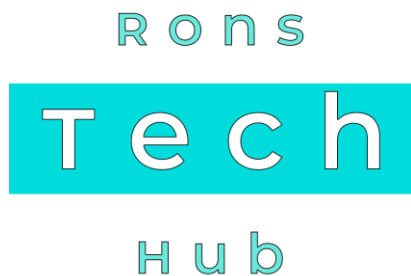
```
def remind_user():  
    show_message("Time for a break!") set_timer(30_minutes, remind_user)
```

R O N S

T e c h

H u b

Feature: Trigger Functions



Functions that start events.



Like pushing a domino to start a chain reaction.

Feature: Trigger Functions Example

```
def start_game():  
    trigger_game_start_event()  
    initialize_players()  
    start_timer()
```




Event Driven Real World Examples



R O N S

T e c h

H u b



Button clicks in mobile apps.



Form submissions on websites.



Game character movement.



Chat message notifications.



Auto-save in text editors.

Advantages Of Event Driven Programming

R O N S

T e c h

H u b



Better user interaction.



Responsive applications.



Efficient resource use.

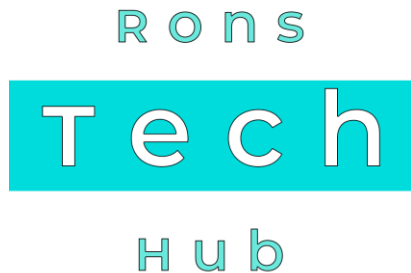


Easy to maintain.



Natural way to handle user input.

Disadvantages Of Event Driven Programming



Complexity: Harder to debug, flow is less clear.

Understanding: Code can be scattered, hard to see the big picture.

Shared Data: Managing data between handlers is tricky.

Testing: Simulating events is difficult.

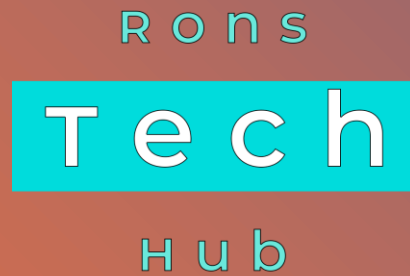
Callbacks: Nested callbacks can be messy.

Control: Flow is inverted, can be confusing.

Resources: Handlers might consume resources unnecessarily.



Next Time



Coding For The Web

