**Cookies in Django 5**

response.set\_cookie(

    key,    # (Required) The name of the cookie (e.g., "username""session\_id").

    value='',# (Optional) The value you want to store. Default is an empty string.

    max\_age=None,          # (Optional) Cookie lifetime in seconds.

                           # Example: max\_age=60\*60\*24\*2  → cookie will last for 2 days.

    expires=None,          # (Optional) Expiration date/time for the cookie.

                           # Example: expires=datetime.utcnow() + timedelta(days=2)

                           # If both max\_age and expires are given, max\_age has priority.

    path='/',              # (Optional) URL path for which the cookie is valid.

                           # Default '/' means cookie is available for the entire site.

                           # Example: path='/home' → only accessible under /home URLs.

    domain=None,           # (Optional) Domain for which the cookie is valid.

                           # Example: domain="geekyshows.com" → works for that domain.

    secure=False,          # (Optional) If True → cookie only sent via HTTPS (secure connection).

                           # Prevents leakage over HTTP.

    httponly=False,        # (Optional) If True → cookie is \*\*not accessible via JavaScript\*\*.

                           # Helps protect against XSS (cross-site scripting) attacks.

    samesite=None          # (Optional) Restricts cross-site cookie sending.

                           # Options:

                           #   'Lax'   → cookies sent only for safe cross-site requests (default in Django 3.1+).

                           #   'Strict'→ cookies sent only from the same site (more secure).

                           #   'None'  → cookies sent in all requests, but must be Secure=True.

)

**Django Cookies – Notes**

**✅ What is a Cookie?**

* A **cookie** is a small piece of data stored in the user’s browser.
* It is sent by the server → stored in the client’s browser → sent back to the server with each request.
* Common uses:
  + User authentication (login sessions).
  + Storing preferences (language, theme, cart items).
  + Tracking user activity (analytics).

👉 Purpose: **Maintain state between requests** (since HTTP is stateless).

**📌 Your Code with Explanations**

**views.py**

from django.shortcuts import render

from datetime import datetime, timedelta, timezone   # for expires example

# ------------------ SET COOKIE ------------------

def setcookie(request):

    # Rendering a response from template

    response = render(request, 'student/setcookie.html')

    # Store cookie with key='pay\_id', value='pay123456'

    # max\_age=3600 → cookie will expire after 1 hour (3600 seconds)

    response.set\_cookie('pay\_id', 'pay123456', max\_age=3600)

    # Another way: use 'expires' → expire after 2 days

    # response.set\_cookie('pay\_id', 'ppp123',

    #                     expires=datetime.now(timezone.utc) + timedelta(days=2))

    return response

# ------------------ GET COOKIE ------------------

def getcookie(request):

    # Method 1 (Unsafe): Direct access – will throw KeyError if cookie does not exist

    # pay\_id = request.COOKIES['pay\_id']

    # Method 2 (Safe): Using .get() → returns None (or default) if cookie is missing

    pay\_id = request.COOKIES.get('pay\_id')  # returns value or None

    # Passing cookie value to template

    response = render(request, 'student/getcookie.html', {'pay\_id': pay\_id})

    return response

# ------------------ DELETE COOKIE ------------------

def delcookie(request):

    response = render(request, 'student/delcookie.html')

    # Delete cookie 'pay\_id' from browser

    response.delete\_cookie('pay\_id')

    return response

# ------------------ SET SIGNED COOKIE ------------------

def setsignedcookie(request):

    response = render(request, 'student/setsignedcookie.html')

    # set\_signed\_cookie adds a "signature" for security

    # salt='tk' → adds extra layer so value can’t be tampered with

    response.set\_signed\_cookie('token', 'tk12345', salt='tk')

    return response

# ------------------ GET SIGNED COOKIE ------------------

def getsignedcookie(request):

    # get\_signed\_cookie verifies signature

    # If tampered → raises BadSignature error

    # default="guesttoken123" → returned if cookie missing

    token = request.get\_signed\_cookie('token',

                                      default="guesttoken123",

                                      salt='tk')

    return render(request, 'student/getsignedcookie.html', {'token': token})

**urls.py**

from django.urls import path

from student.views import (

    getcookie, setcookie, delcookie,

    getsignedcookie, setsignedcookie

)

urlpatterns = [

    path('get/', getcookie, name='getcookie'),           # read cookie

    path('set/', setcookie, name='setcookie'),           # set cookie

    path('del/', delcookie, name='delcookie'),           # delete cookie

    path('getsigned/', getsignedcookie, name='getsignedcookie'),  # read signed cookie

    path('setsigned/', setsignedcookie, name='setsignedcookie'),  # set signed cookie

]

**🎯 Important Notes**

**🔹 set\_cookie() Parameters**

* key → cookie name (required).
* value → cookie value (string, int, etc.).
* max\_age → seconds until cookie expires (e.g., 3600 = 1 hour).
* expires → fixed datetime when cookie should expire.
* path → URL scope (default '/' → whole site).
* domain → domain restriction.
* secure=True → cookie only over HTTPS.
* httponly=True → cookie not accessible via JavaScript (XSS protection).
* samesite='Lax' | 'Strict' | 'None' → CSRF protection.

**🔹 Signed Cookies**

* Normal cookies can be modified by the client (unsafe).
* **Signed cookies** use a cryptographic signature → Django can detect tampering.
* set\_signed\_cookie() → creates secure cookie.
* get\_signed\_cookie() → reads + verifies cookie.

**✅ Best Practices**

1. Always use **httponly=True** for sensitive cookies.
2. Use **secure=True** if your site is HTTPS.
3. Prefer **signed cookies** for critical data.
4. For large data, use **sessions** instead of cookies.