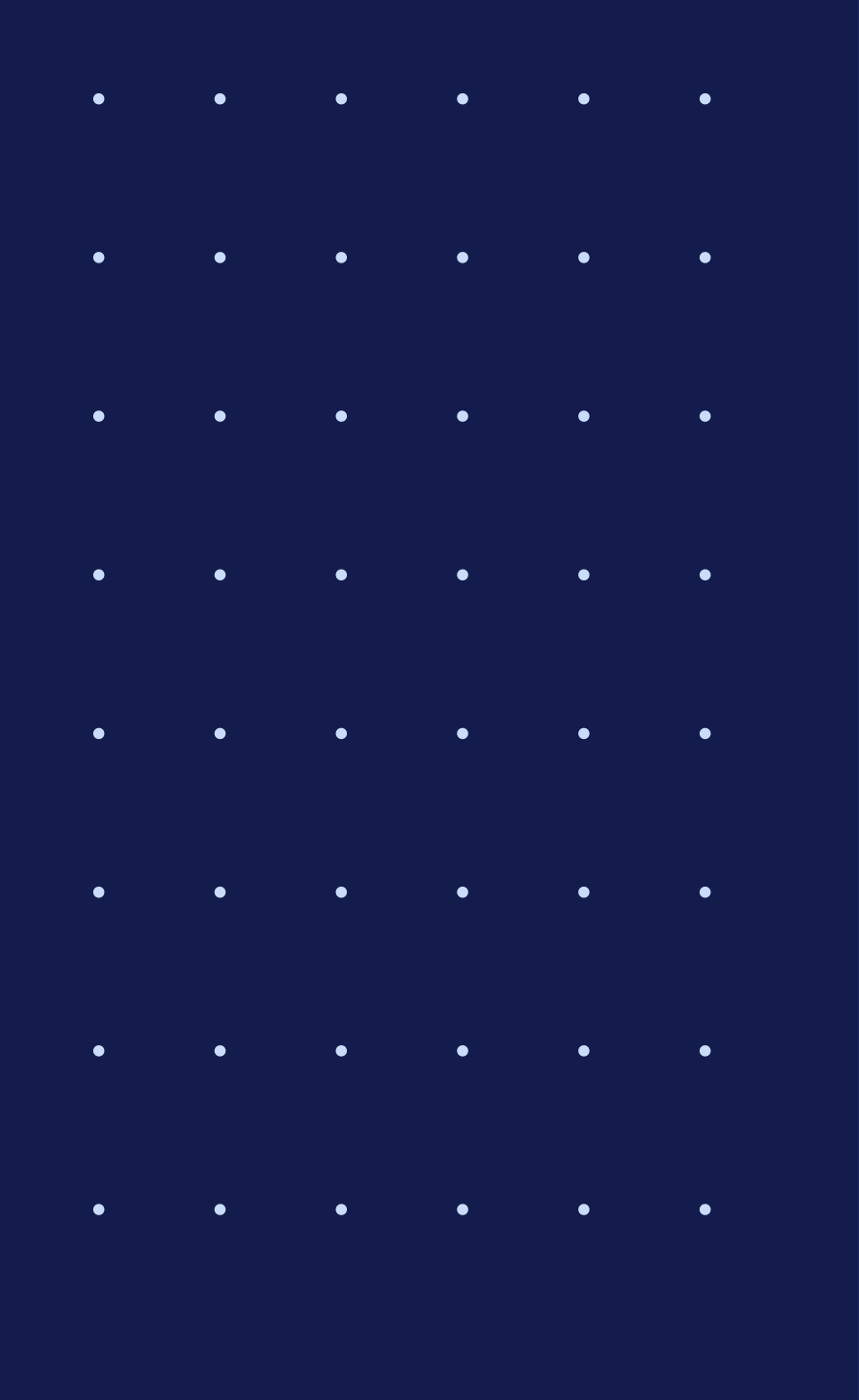




ТЕМА 2

Змінні, типи даних та оператори PHP

*Області видимості · суперглобальні змінні · константи · типи ·
оператори · передача за посиланням · дата й час · рядки*

Бекенд-розробка, сервіси та мікросервіси

Комп'ютерна інженерія · 3 курс · Лекція 2 з 18

12 питань сьогоднішньої теми

01

Змінні: оголошення та іменування

02

Області видимості змінних

03

Суперглобальні змінні

04

Вивід значень на екран

05

Константи

06

Магічні константи

07

Коментарі

08

Типи змінних

09

Оператори в PHP

10

Передача за посиланням

11

Дата та час

12

Рядки: форматування й парсинг

Оголошення та іменування

```
<?php
$name = "Мар'ян";      // string
$age = 21;              // int
$price = 19.99;         // float
$isActive = true;       // bool

// Тип визначається автоматично
$value = 10;            // int
$value = "десять";      // тепер string
```

- Починається з \$, далі літера/підкреслення
- Регістрозалежна: \$name ≠ \$Name
- Не може починатись з цифри
- camelCase або snake_case

Локальна vs глобальна область видимості

```
$counter = 10;    // глобальна змінна

function increment() {
    $counter = 1;    // це ІНША, локальна змінна
    $counter++;
    echo $counter;    // виведе 2
}

increment();
echo $counter;        // виведе 10 – глобальна не змінилась
```

Кожна функція має власну ізольовану область видимості. Щоб звернутись до глобальної змінної, потрібне ключове слово **global** або масив **\$GLOBALS**.

Статичні змінні: пам'ять між викликами

```
function countCalls() {  
    static $calls = 0;  
    $calls++;  
    echo "Викликів: $calls\n";  
}
```

```
countCalls(); // Викликів: 1  
countCalls(); // Викликів: 2  
countCalls(); // Викликів: 3
```

`static` зберігає значення локальної змінної між викликами функції — на відміну від звичайної локальної змінної.

Використання: лічильники, кешування результату, одноразова ініціалізація.

Доступні в будь-якій області видимості

Суперглобальна	Призначення
<code>\$_GET</code>	Параметри рядка запиту (?id=5)
<code>\$_POST</code>	Дані, надіслані методом POST
<code>\$_SERVER</code>	Інформація про сервер і середовище
<code>\$_SESSION</code>	Дані сесії користувача
<code>\$_COOKIE</code>	HTTP-куки клієнта
<code>\$_FILES</code>	Завантажені файли
<code>\$GLOBALS</code>	Доступ до всіх глобальних змінних

Дані з `$_GET` і `$_POST` завжди потрібно валідувати перед використанням.

echo, print_r, var_dump

echo / print

Виводить прості значення (рядки, числа)

```
echo "Привіт, $name";
```

print_r()

Виводить структуру масиву чи об'єкта

```
print_r($user);
```

var_dump()

Значення разом із типом даних і розміром

```
var_dump($user);
```

define() vs const

```
define('SITE_NAME', 'IFNTUOG Portal');  
const VERSION = '1.0.0';
```

```
echo SITE_NAME;    // без символу $  
echo VERSION;
```

Критерій	define()	const
Область застосування	Будь-де, у тому числі умовно	Верхній рівень файлу / клас
Час визначення	Під час виконання	Під час компіляції
У класах	Не застосовується	Константи класу

Значення залежить від контексту

`__LINE__`

Поточний номер рядка

`__FILE__`

Шлях до поточного файлу

`__DIR__`

Директорія поточного файлу

`__FUNCTION__`

Ім'я поточної функції

`__CLASS__`

Ім'я поточного класу

`__METHOD__`

Клас::метод

```
function logError($message) {  
    echo "[" . __FUNCTION__ . "] Рядок " . __LINE__ . ": $message";  
}  
logError("Файл не знайдено");  
// [logError] Рядок 2: Файл не знайдено
```

Три стилі коментарів у PHP

```
// Однорядковий коментар (стиль C++)  
# Однорядковий коментар (стиль Perl/shell)  
  
/*  
 * Багаторядковий коментар.  
 */  
  
/**  
 * PHPDoc-коментар – використовується IDE для підказок  
 * @param string $name Ім'я користувача  
 * @return string Привітання  
 */  
function greet($name) {  
    return "Привіт, $name!";  
}
```

Скалярні та складені типи

int

Ціле число

21

float

Число з крапкою

19.99

string

Рядок символів

"ТЕКСТ"

bool

true / false

true

Складені та спеціальні типи

array

object

callable

null

resource

Type juggling: PHP автоматично приводить значення до потрібного типу залежно від контексту. Для явного контролю — функції `is_*()` та приведення типів: `(int) $value`.

Арифметичні та оператори порівняння

Арифметичні / рядкові

+ **-** ***** **/** Основні арифметичні дії

% Остача від ділення

****** Піднесення до степеня

. **(крапка)** Конкатенація рядків

Порівняння

== Рівність (з приведенням типів)

=== Строга рівність (значення й тип)

!= **!==** Нерівність / строга нерівність

<=> Космічний корабель: -1, 0 або 1

Правило: завжди використовуйте **===** замість **==**. **"0" == false** і **"abc" == 0** можуть повертати **true** через неявне приведення типів.

Логічні оператори та null coalescing

Оператор	Значення
&& and	Логічне І
or	Логічне АБО
!	Логічне заперечення
?:	Скорочений тернарний оператор
??	Об'єднання з null
??=	Присвоєння, якщо ще не визначено

```
$username = $_GET['user'] ?? 'Гість';
```

```
$count = $count ?? 0;
```

```
$count ??= 0;    // те саме, коротше
```

```
$label = ($age >= 18)  
    ? 'дорослий'  
    : 'неповнолітній';
```

За значенням vs за посиланням

За значенням (за замовчуванням)

```
function addTax($price) {  
    $price = $price * 1.2;  
    return $price;  
}  
$original = 100;  
$withTax = addTax($original);  
echo $original; // 100  
echo $withTax;  // 120
```

За посиланням (&\$price)

```
function addTaxByRef(&$price) {  
    $price = $price * 1.2;  
}  
$original = 100;  
addTaxByRef($original);  
echo $original; // 120  
// змінилось напряму!
```

Функції date() і time()

```
echo time();           // 1798113600 (Unix-час)
echo date("Y-m-d");    // 2026-09-02
echo date("d.m.Y H:i:s"); // 02.09.2026 14:30:00
echo date("l, d F Y");  // Wednesday, 02 September 2026
```

Символ	Значення	Символ	Значення
Y	Рік, 4 цифри	H	Година, 24-год
m	Місяць, з нулем	i	Хвилини
d	День місяця	s	Секунди
l	Назва дня тижня	F	Повна назва місяця

Клас DateTime: арифметика над датами

```
$date = new DateTime('2026-09-02');  
$date->modify('+1 month');  
echo $date->format('Y-m-d');    // 2026-10-02  
  
$interval = new DateInterval('P7D'); // 7 днів  
$date->add($interval);  
echo $date->format('Y-m-d');    // 2026-10-09  
  
$deadline = new DateTime('2026-12-31');  
$diff = $date->diff($deadline);  
echo $diff->days . ' днів залишилось';
```

`strtotime('+1 week')` та `new DateTime('next monday')` розуміють природну мову дат англійською.

Одинарні, подвійні лапки, heredoc

```
$name = "Оксана";  
echo 'Привіт, $name';    // Привіт, $name (без підстановки)  
echo "Привіт, $name";    // Привіт, Оксана  
  
$html = <<<EOT  
<div>  
    <p>Користувач: $name</p>  
</div>  
EOT;
```

Форматування: sprintf і функції обробки

```
echo sprintf("Ціна: %.2f грн", 1234.5); // Ціна: 1234.50 грн
echo sprintf("Номер: %05d", 42);         // Номер: 00042
echo number_format(1234.5, 2, ',', ' '); // 1 234,50
```

Функція	Призначення
strlen(\$s)	Довжина рядка
substr(\$s, \$start, \$len)	Витягнути підрядок
str_replace(\$s, \$r, \$str)	Замінити входження
trim(\$s)	Прибрати пробіли з країв
explode() / implode()	Розбити / з'єднати рядок і масив
str_pad(\$s, \$len, \$ch)	Доповнити до довжини

Парсинг: explode() та регулярні вирази

```
$csv = "Іван,25,Львів";  
[$name, $age, $city] = explode(',', $csv);  
echo "$name живе у місті $city";  
  
// Парсинг за допомогою регулярного виразу  
$text = "Телефон: +380671234567";  
if (preg_match('/\+380\d{9}/', $text, $matches)) {  
    echo "Знайдено номер: " . $matches[0];  
}
```

Регулярні вирази — потужний, але «важкий» інструмент: для простих задач спочатку розгляньте прості рядкові функції.

Підсумок теми 2

- Змінні PHP типізуються динамічно, кожна функція має свою область видимості
- Суперглобальні масиви (`$_GET`, `$_POST`, `$_SERVER`) доступні всюди
- Завжди використовуйте `===` замість `==` для строгого порівняння
- Аргументи передаються за значенням; `&` робить передачу за посиланням
- `DateTime` і `sprintf()` — основні інструменти для дат і форматування рядків