INSTRUKCJA PRAKTYCZNA • KWALIFIKACJA INF.03

PĘTLE W PHP OD ZERA

10 lekcji krok po kroku.
Pętle **for**, **while**, **do...while** i **foreach**
oraz ich zastosowanie w formularzach i MySQL.

JAK KORZYSTAĆ Z MATERIAŁU

Czytaj krótkie wyjaśnienie, przepisz i uruchom przykład, zmień wskazane wartości, a następnie wykonaj **zadanie samodzielne**. Nie przechodź dalej, jeśli nie wiesz, dlaczego kod działa. Zapisuj pliki po każdej lekcji.

Poziom: od podstaw • **Środowisko:** XAMPP + edytor kodu + przeglądarka • **Czas:** 10 × 45 minut

Materiał ćwiczeniowy dla uczniów technikum. Zakres dostosowany do INF.03.5: programowanie skryptów po stronie serwera, instrukcje sterujące, formularze HTML i odczyt danych z relacyjnej bazy danych.

Wersja: listopad 2026 • Kod: PHP 8 / MySQL (MariaDB w XAMPP)

WAŻNE: KOD PRZEPISUJEMY SAMODZIELNIE

Samo kopiowanie i wklejanie gotowego kodu **nie nauczy Cię programowania**. Gdy samodzielnie wpisujesz każdą linię, ćwiczysz składnię PHP, zauważasz znaki specjalne i uczysz się wychwytywać błędy. **Dzięki temu więcej zostaje w pamięci**, a na sprawdzianie i egzaminie INF.03 łatwiej napiszesz program bez podpowiedzi. **Przykłady kodu w tym PDF celowo nie pozwalają na kopiowanie.**

Plan nauki – 10 godzin lekcyjnych

Lekcja	Temat	Co wykonasz
1	Start z PHP i idea pętli	Pierwszy skrypt, zmienne, warunek, licznik
2	Pętla for – od podstaw	Liczby i lista HTML
3	Pętla for w HTML	Wiersze tabeli i opcje formularza
4	for + POST	Powtarzanie tekstu podanego przez użytkownika
5	Pętla while	Licznik, suma, warunek zakończenia
6	while + POST	Symulator oszczędzania
7	Pętla do...while	Co najmniej jedno wykonanie
8	Pętla foreach	Tablice, lista produktów, tabela HTML
9	foreach + formularz	Obsługa wielu pól checkbox
10	while + baza MySQL	Wyświetlanie rekordów + zadanie INF.03

Każda lekcja jest zaplanowana na około **45 minut**: 10 min wprowadzenia, 15 min pracy z przykładem, 15 min samodzielnego zadania, 5 min sprawdzenia. Gdy tempo klasy jest wolniejsze, można przenieść zadanie do następnej lekcji.

Cele całego kursu

- Odróżniasz cztery rodzaje pętli i potrafisz wybrać właściwą.
- Piszesz warunki zakończenia pętli i unikniesz nieskończonego wykonywania kodu.
- Generujesz elementy HTML przy pomocy PHP: listy, wiersze tabeli i opcje select.
- Odbierasz dane z formularza metodą POST, sprawdzasz zakres i wyświetlasz bezpiecznie tekst użytkownika.
- Przetwarzasz dane tablicy foreach oraz wyniki zapytań MySQL za pomocą while.

Ważne zasady

ZANIM URUCHOMISZ PĘTLĘ

Sprawdź trzy rzeczy: **od jakiej wartości zaczynam? do kiedy powtarzam? co zmienia się w każdym obrocie?** Jeśli ostatnia odpowiedź brzmi „nic”, kod może się zapętlić.

Legenda: granatowy blok = kod do przepisania; żółty blok = zadanie do samodzielnej pracy; turkusowy blok = ważna uwaga; tabele = porównania i przewidywane wyniki.

LEKCJA 1 / 10 • 45 min

Zaczynamy od zera: PHP i powtarzanie

Po tej lekcji potrafisz: uruchomić plik PHP, wyświetlić tekst, użyć zmiennej oraz wyjaśnić, dlaczego potrzebujemy pętli.

> **ORGANIZACJA PRACY**
> Przygotuj plik **01_start.php** w folderze **C:\xampp\htdocs\petle_inf03** i otwórz go przez adres **http://localhost/petle_inf03/01_start.php**.

1. Co jest potrzebne?

PHP to język wykonywany **na serwerze**. W naszej pracowni serwerem będzie lokalny Apache z XAMPP. Przeglądarka dostaje wynik działania PHP (najczęściej HTML), a nie sam kod PHP.

- Uruchom **XAMPP Control Panel** i kliknij **Start** przy Apache.
- Otwórz folder **C:\xampp\htdocs** i utwórz w nim podfolder **petle_inf03**.
- W edytorze utwórz plik **01_start.php** (nie 01_start.php.txt).
- Otwieraj plik przez **http://localhost/petle_inf03/01_start.php**, nie dwukrotnym kliknięciem w Eksploratorze.

2. Pierwszy program

PLIK: **01_start.php**

```
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Start PHP</title>
</head>
<body>
    <h1>Moja pierwsza strona PHP</h1>
    <?php
    $imie = "Ala";
    echo "Witaj, " . $imie . "!";
    ?>
</body>
</html>
```

Wyjaśnienie: **<?php** otwiera kod PHP, **echo** wypisuje wynik, **$imie** przechowuje tekst, a **.** łączy napisy. Znak **;** kończy większość instrukcji PHP. **<meta charset="UTF-8">** pomaga prawidłowo wyświetlać polskie znaki.

3. Problem: to samo polecenie pięć razy

```
<?php
echo "Ćwiczę PHP<br>";
echo "Ćwiczę PHP<br>";
echo "Ćwiczę PHP<br>";
echo "Ćwiczę PHP<br>";
echo "Ćwiczę PHP<br>";
?>
```

Takie powtarzanie ręczne jest niewygodne. Jeśli trzeba wypisać ten sam tekst 100 razy, potrzebujemy **pętli** - instrukcji, która wykonuje blok kodu wielokrotnie.

4. Zobacz pierwszy przykład pętli

```
<?php
for ($i = 1; $i <= 5; $i++) {
    echo "Ćwiczę PHP: " . $i . "<br>";
}
?>
```

Nie musisz jeszcze znać całej składni. Zauważ tylko: zmienna **$i** zaczyna od 1, zwiększa się o 1 i wykonanie kończy się po wypisaniu liczby 5.

Powtórzenie	Wartość $i	Wynik
1	1	Ćwiczę PHP: 1
2	2	Ćwiczę PHP: 2
3	3	Ćwiczę PHP: 3
4	4	Ćwiczę PHP: 4
5	5	Ćwiczę PHP: 5

ZADANIE SAMODZIELNE | Pierwsze powtórzenia

1. Zapisz działający plik PHP z nagłówkiem HTML **„Poznaję pętle PHP”**.

2. Ustaw w zmiennej **$przedmiot** tekst „INF.03”.

3. Za pomocą pokazanej pętli wypisz **3 razy** komunikat „Uczę się INF.03 – numer 1”, „... numer 2” itd.

4. Zapisz zrzut ekranu działania strony (jeśli poleci nauczyciel).

Wskazówki: Przepisz przykład; zmień **5** na **3**. Połącz tekst ze zmiennymi kropką.

Sprawdź: Na stronie są dokładnie 3 komunikaty; nie ma widocznego kodu <?php.

LEKCJA 2 / 10 • 45 min

Pętla for: znam liczbę powtórzeń

Po tej lekcji potrafisz: rozpoznać start, warunek i krok; przewidzieć liczbę obrotów i napisać prostą pętlę for.

> **ORGANIZACJA PRACY**
> Przygotuj plik **02_for.php** w folderze **C:\xampp\htdocs\petle_inf03** i otwórz go przez adres **http://localhost/petle_inf03/02_for.php**.

1. Budowa pętli for

```
for ($i = 1; $i <= 5; $i++) {
    echo $i . "<br>";
}
```

Część	Znaczenie	Przykład
$i = 1	Początek - wartość ustawiana jeden raz.	Zaczynamy od 1.
$i <= 5	Warunek - czy wykonać kolejny obrót?	Tak, dopóki $i jest co najwyżej 5.
$i++	Krok - zmiana licznika po obrocie.	Powiększ $i o 1.
{ ... }	Instrukcje powtarzane w pętli.	Wypisz numer i przejdź do następnej linii.

Kolejność: **ustaw licznik → sprawdź warunek → wykonaj kod → zmień licznik → sprawdź warunek ponownie**.

2. Uwaga na liczbę powtórzeń

Pętla	Wypisywane liczby	Ile razy?
`for ($i=1; $i<=5; $i++)`	1, 2, 3, 4, 5	5
`for ($i=0; $i<5; $i++)`	0, 1, 2, 3, 4	5
`for ($i=1; $i<5; $i++)`	1, 2, 3, 4	4
`for ($i=2; $i<=10; $i+=2)`	2, 4, 6, 8, 10	5

Zapis **$i++** znaczy: zwiększ o 1. Zapis **$i += 2** znaczy: zwiększ o 2. Operator **<=** oznacza „mniejsze lub równe”, a **<** „ściśle mniejsze”.

3. Ćwiczenie prowadzone: liczby parzyste

PLIK: **02_for.php**

```
<?php
for ($i = 2; $i <= 12; $i += 2) {
    echo "Liczba: " . $i . "<br>";
}
?>
```

Sprawdź wynik: liczby 2, 4, 6, 8, 10, 12. Teraz zmień krok z **2** na **3** i zapisz, co wyświetli skrypt.

4. Pętla tworzy listę HTML

```
<h2>Stanowiska komputerowe</h2>
<ol>
<?php
for ($i = 1; $i <= 4; $i++) {
    echo "<li>Stanowisko nr " . $i . "</li>";
}
?>
</ol>
```

Wynikiem jest zwykły HTML: otwarcie listy **
**, cztery elementy **
** i zamknięcie listy **
. PHP nie zastępuje HTML - może go **wygenerować.

ZADANIE SAMODZIELNE | Lista stanowisk w sali komputerowej

1. Utwórz plik **02_stanowiska.php** z poprawną strukturą HTML5.

2. Wygeneruj w pętli **for** listę uporządkowaną 12 stanowisk.

3. Każdy element ma mieć treść „Komputer 1”, „Komputer 2” itd.

4. Pod listą wyświetl osobny napis „Wygenerowano 12 stanowisk”.

Wskazówki: Umieść <ol> przed PHP i po PHP. W każdej iteracji wypisz jeden <li>.

Sprawdź: Sprawdź pierwszy i ostatni element; ma być dokładnie 12 pozycji.

SPRAWDŹ, CZY ROZUMIESZ: 1. Ile razy wykona się `for ($i=0; $i<9; $i+=3)`? **2.** Jaka jest różnica pomiędzy **$i++** a **$i += 2**? **3.** Dlaczego instrukcja **echo** musi znaleźć się pomiędzy klamrami?

LEKCJA 3 / 10 • 45 min

for w HTML: tabele i pola select

Po tej lekcji potrafisz: dynamicznie wygenerować wiersze tabeli oraz opcje formularza z użyciem pętli for.

> **ORGANIZACJA PRACY**
> Przygotuj plik **03_html_for.php** w folderze **C:\xampp\htdocs\petle_inf03** i otwórz go przez adres **http://localhost/petle_inf03/03_html_for.php**.

1. Jak utworzyć wiele wierszy tabeli?

Kiedy znamy liczbę wierszy, nie musimy pisać wielu znaczników **<tr>**. Wystarczy utworzyć nagłówek tabeli tylko raz, a następnie powtarzać wiersze wewnątrz pętli.

PLIK: **03_html_for.php**

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Tabela z pętli</title>
</head>
<body>
<h1>Lista miejsc</h1>
<table border="1">
  <tr><th>Numer</th><th>Nazwa miejsca</th></tr>
  <?php
  for ($i = 1; $i <= 5; $i++) {
      echo "<tr>";
      echo "<td>" . $i . "</td>";
      echo "<td>Miejsce nr " . $i . "</td>";
      echo "</tr>";
  }
  ?>
</table>
</body>
</html>
```

Co dzieje się w każdym obrocie? PHP wypisuje jeden otwierający znacznik **<tr>**, dwie komórki **<td>**, a potem zamknięcie wiersza **</tr>**. Nagłówki **<th>** nie są powtarzane.

2. Generowanie opcji select w formularzu

Formularz może zawierać wybór liczby sztuk, np. od 1 do 10. Pętla pozwala uzyskać dziesięć opcji bez ręcznego wpisywania dziesięciu znaczników.

```
<form action="04_post_for.php" method="post">
  <label for="ile">Ile sztuk?</label>
  <select name="ile" id="ile">
    <?php
    for ($i = 1; $i <= 10; $i++) {
        echo '<option value="' . $i . '">' . $i . ' szt.</option>';
    }
    ?>
  </select>
  <button type="submit">Wyślij</button>
</form>
```

Atrybut **value** określa wartość przesłaną do PHP. **name="ile"** to nazwa pola, pod którą potem odczytamy wartość w **$_POST["ile"]**. Plik odbierający przygotujemy na następnej lekcji.

3. Dwie pułapki

NAJCZĘSTSZE BŁĘDY

Błąd 1: wypisanie <table> w każdej iteracji tworzy kilka osobnych tabel zamiast jednej. **Błąd 2:** brak </tr> lub </td> psuje strukturę tabeli. Najpierw narysuj jeden poprawny wiersz HTML, dopiero potem go powtarzaj.

ZADANIE SAMODZIELNE | Tabela dyżurów

1. W pliku **03_dyzury.php** utwórz tabelę HTML z kolumnami **Dzień** oraz **Dyżur**.

2. Pętla **for** ma wygenerować **7 wierszy** z treścią: „Dzień 1 – Dyżur nr 1”, aż do dnia 7.

3. Pod tabelą utwórz formularz z listą rozwijaną **Wybierz dzień** (opcje 1-7).

4. W formularzu użyj metody **post** (obsługę wybranego dnia dodamy później).

Wskazówki: Skorzystaj z dwóch przykładów: pierwszy generuje <tr>, drugi <option>.

Sprawdź: Sprawdź liczbę wierszy i opcji. W kodzie źródłowym strony powinno być siedem <tr> z danymi.

SPRAWDŹ, CZY ROZUMIESZ: 1. Które znaczniki tabeli nie powinny powtarzać się w każdej iteracji? **2.** Co oznacza atrybut **name** pola formularza? **3.** Dlaczego każdy wiersz tabeli ma zawierać zamykający znacznik </tr>?

LEKCJA 4 / 10 • 45 min

for i formularz POST: użytkownik steruje pętlą

Po tej lekcji potrafisz: odebrać dane metodą POST, sprawdzić zakres i wypisać określoną liczbę komunikatów za pomocą for.

> **ORGANIZACJA PRACY**
> Przygotuj plik **04_post_for.php** w folderze **C:\xampp\htdocs\petle_inf03** i otwórz go przez adres **http://localhost/petle_inf03/04_post_for.php**.

1. Droga danych: HTML → PHP → wynik

Użytkownik wybiera liczbę powtórzeń i wpisuje komunikat. Po kliknięciu przycisku **Wyślij** przeglądarka przesyła dane do pliku PHP metodą **POST**. Skrypt sprawdza dane, a następnie wykonuje pętlę. Dzięki temu liczba obrotów nie musi być wpisana na stałe.

Element	Znaczenie
method="post"	Przesyła dane w treści żądania HTTP, nie w adresie URL.
name="ile"	Nazwa pola odczytywanego jako $_POST["ile"].
isset($_POST["ile"])	Sprawdza, czy pole zostało przesłane.
filter_var(..., FILTER_VALIDATE_INT)	Sprawdza, czy wartość jest liczbą całkowitą.
htmlspecialchars(...)	Zamienia znaki specjalne w tekście na encje HTML przed wypisaniem.

2. Kompletny przykład w jednym pliku

PLIK: **04_post_for.php**

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Powtarzanie tekstu</title>
</head>
<body>
<h1>Powtarzanie tekstu - for</h1>
<form action="" method="post">
  <label>Tekst: <input name="tekst" required></label>
  <label>Ile razy (1-20):
    <input type="number" name="ile" min="1" max="20"
           value="3" required>
  </label>
  <button type="submit">Pokaż</button>
</form>
<?php
if (isset($_POST["tekst"], $_POST["ile"])) {
    $tekst = trim($_POST["tekst"]);
    $ile = filter_var($_POST["ile"], FILTER_VALIDATE_INT);
    if ($tekst !== "" && $ile !== false &&
        $ile >= 1 && $ile <= 20) {
        for ($i = 1; $i <= $ile; $i++) {
            echo "<p>$i. " . htmlspecialchars($tekst) . "</p>";
```

```
        }
    } else {
        echo "<p>Wpisz tekst i liczbę od 1 do 20.</p>";
    }
}
?>
</body>
</html>
```

Przeczytaj uważnie: znacznik **<form>** pokazuje pola, **isset** sprawdza, czy formularz wysłano, **trim** usuwa białe znaki na początku i końcu tekstu, **filter_var** sprawdza liczbę, a **for** powtarza komunikat dokładnie tyle razy, ile wybrano.

> **DLACZEGO WALIDACJA JEST POTRZEBNA?**
>
> Atrybuty **min**, **max** i **required** ułatwiają wpisywanie poprawnych danych w przeglądarce, ale nie zastępują kontroli w PHP. Skrypt po stronie serwera musi sam pilnować zakresu.

3. Testujemy

Wpisz	Oczekiwany wynik
tekst: Test, ile: 3	Trzy akapity: 1. Test, 2. Test, 3. Test.
tekst: Test, ile: 0	Komunikat o błędnym zakresie.
tekst: Test, ile: 21	Komunikat o błędnym zakresie.
tekst: Cześć, ile: 1	Tekst znacznika pokazany dosłownie, bez pogrubienia.

ZADANIE SAMODZIELNE | Generator powitań

1. Utwórz nowy plik **04_powitania.php**. Przygotuj formularz POST z polem **imie** oraz **ile**.

2. Imię ma być niepuste, a liczba powtórzeń ma wynosić od **1 do 15**.

3. Po poprawnym wysłaniu wyświetl numerowane powitania, np. „1. Witaj, Jan!”.

4. W przypadku błędnych danych wyświetl komunikat, zamiast uruchamiać pętlę.

Wskazówki: Zmień pole „tekst” na „imie” w przykładzie; zostaw warunek **isset**, walidację i **htmlspecialchars**.

Sprawdź: Dla Ola i 4 pokazują się cztery powitania. Dla 0 lub 16 nie ma powitań.

SPRAWDŹ, CZY ROZUMIESZ: 1. Jaka jest rola **isset**, a jaka rola **for**? **2.** Dlaczego nie wolno ufać tylko atrybutowi **max** w HTML? **3.** Jakiej funkcji użyć przed wypisaniem imienia do HTML?

LEKCJA 5 / 10 • 45 min

Pętla while: wykonuj, dopóki warunek jest prawdziwy

Po tej lekcji potrafisz: napisać pętlę while z licznikiem, wyznaczyć sumę i rozpoznać możliwość nieskończonej pętli.

> **ORGANIZACJA PRACY**
> Przygotuj plik **05_while.php** w folderze **C:\xampp\htdocs\petle_inf03** i otwórz go przez adres **http://localhost/petle_inf03/05_while.php**.

1. Kiedy wybieramy while?

Pętla **while** jest wygodna, gdy program ma pracować **dopóki spełniony jest warunek**. Warunek sprawdzany jest **przed każdym obrotem**. Jeśli od początku jest fałszywy, wnętrze pętli nie wykona się ani razu.

PLIK: **05_while.php**

```
<?php
$i = 1;                 // 1. Start
while ($i <= 5) {       // 2. Sprawdź warunek
    echo $i . "<br>";   // 3. Wykonaj polecenie
    $i++;               // 4. Zmień licznik
}
?>
```

2. Porównanie for i while

for	while
`for ($i=1; $i<=5; $i++) { ... }`	`$i=1; while ($i<=5) { ...; $i++; }`
Wszystkie elementy sterujące stoją w nagłówku pętli.	Start jest przed pętlą; krok zwykle wewnątrz klamer.
Wygodna przy znanej liczbie powtórzeń.	Wygodna przy powtarzaniu do spełnienia warunku.

3. Liczenie sumy: od 1 do 5

```
<?php
$i = 1;
$suma = 0;
while ($i <= 5) {
    $suma = $suma + $i;
    echo "Dodaję " . $i . ", suma: " . $suma . "<br>";
    $i++;
}
echo "Wynik końcowy: " . $suma;
?>
```

Obrót	Dodana liczba	Suma po dodaniu
1	1	1
2	2	3
3	3	6
4	4	10
5	5	15

Zmienna **$suma** musi istnieć **przed pętlą** i na początku mieć wartość **0**. W każdym obrocie dodajemy do niej kolejną liczbę. Po wyjściu z pętli zawiera końcowy wynik.

4. Nie zapomnij o zmianie licznika

```
$i = 1;
while ($i <= 5) {
    echo $i;
    // BRAK $i++ -> warunek stale pozostaje prawdziwy!
}
```

UWAGA: PĘTLA NIESKOŃCZONA

Nie uruchamiaj specjalnie powyższego błędnego fragmentu. Jeśli kod nie zmienia wartości wpływającej na warunek, serwer może wykonywać pętlę bardzo długo, aż przerwie ją limit czasu.

ZADANIE SAMODZIELNE | Sumator liczb od 1 do N

1. Utwórz plik **05_suma.php** i ustaw zmienną **$n = 8**.

2. Za pomocą **while** oblicz sumę liczb 1 + 2 + ... + 8.

3. Wyświetl każdą dodawaną liczbę oraz wynik końcowy.

4. Zmień **$n** na 3 i 10 oraz porównaj wyniki.

Wskazówki: Potrzebne są zmienne **$i = 1** oraz **$suma = 0**. W pętli dodawaj liczbę do sumy i zwiększaj **$i**.

Sprawdź: Dla n = 3 wynik 6; dla n = 8 wynik 36; dla n = 10 wynik 55.

SPRAWDŹ, CZY ROZUMIESZ: 1. W którym miejscu **while** sprawdza warunek? **2.** Dlaczego **$suma = 0** zapisujemy przed pętlą? **3.** Co się stanie, jeśli usuniesz **$i++**?

LEKCJA 6 / 10 • 45 min

while i formularz: symulator oszczędzania

Po tej lekcji potrafisz: wykorzystać dane z formularza do pętli kończącej się po osiągnięciu celu oraz testować obliczenia.

> **ORGANIZACJA PRACY**
> Przygotuj plik **06_oszczedzanie.php** w folderze **C:\xampp\htdocs\petle_inf03** i otwórz go przez adres **http://localhost/petle_inf03/06_oszczedzanie.php**.

1. Zadanie z życia: kiedy osiągnę cel?

Wpłacasz co tydzień określoną kwotę. Pętla **while** może liczyć, ile wpłat potrzeba, aby uzbierać wymaganą sumę. Nie znamy liczby wpłat - będzie ona wynikiem obliczenia.

2. Przykład bez formularza

```
<?php
$cel = 120;
$wplata = 25;
$saldo = 0;
$tydzien = 0;
while ($saldo < $cel) {
    $saldo += $wplata;
    $tydzien++;
    echo "Tydzień $tydzien: $saldo zł<br>";
}
echo "Cel osiągnięty po $tydzien tygodniach.";
?>
```

Wyniki: 25, 50, 75, 100, 125 zł. Ostatnia wpłata przekroczyła cel - to poprawne, ponieważ przy stałej wpłacie nie musimy trafić w kwotę idealnie. **Warunek** sprawdzamy po każdym wykonaniu całego obrotu.

3. Podłączamy formularz POST

PLIK: **06_oszczedzanie.php**

```
<form action="" method="post">
  <label>Cel (10-1000 zł):
    <input type="number" name="cel" min="10" max="1000"
           required>
  </label>
  <label>Wpłata tygodniowa (1-100 zł):
    <input type="number" name="wplata" min="1" max="100"
           required>
  </label>
  <button>Oblicz</button>
</form>
<?php
if (isset($_POST["cel"], $_POST["wplata"])) {
    $cel = filter_var($_POST["cel"], FILTER_VALIDATE_INT);
    $wplata = filter_var($_POST["wplata"], FILTER_VALIDATE_INT);
    if ($cel !== false && $wplata !== false &&
        $cel >= 10 && $cel <= 1000 &&
        $wplata >= 1 && $wplata <= 100) {
        $saldo = 0;
        $tydzien = 0;
        while ($saldo < $cel) {
            $saldo += $wplata;
            $tydzien++;
            echo "<p>Tydzień $tydzien: $saldo zł</p>";
```

```
        }
        echo "<strong>Liczba wpłat: $tydzien</strong>";
    } else {
        echo "<p>Podaj poprawne liczby w zakresach.</p>";
    }
}
?>
```

To fragment do umieszczenia **wewnątrz <body> kompletnego dokumentu HTML** z nagłówkiem, językiem i kodowaniem UTF-8. Kod pokazuje formularz i jego obsługę w jednym pliku.

4. Co sprawdzić po uruchomieniu?

Cel / wpłata	Liczba wpłat	Ostatni stan
100 / 20	5	100 zł
100 / 30	4	120 zł
10 / 1	10	10 zł
12 / 100	1	100 zł

ZADANIE SAMODZIELNE | Plan oszczędzania na zakup sprzętu

1. Zrób nowy plik **06_sprzet.php** z formularzem POST: **cena** (20-600 zł) i **odkladam** (5-100 zł).

2. Pętla **while** ma wypisywać kolejne tygodnie oraz łączną zaoszczędzoną kwotę.

3. Na końcu wyświetl „Potrzebujesz X tygodni”, gdzie X jest obliczone przez program.

4. Sprawdź przypadek, gdy wpłata jest większa od celu.

Wskazówki: Użyj wzoru z lekcji. Wybierz nazwy **$saldo**, **$tydzien**, **$cena**, **$odkladam**. Nie usuwaj walidacji.

Sprawdź: Dla cena = 90 i odkladam = 30 wynik to 3 tygodnie. Dla 20 i 100 wynik to 1 tydzień.

SPRAWDŹ, CZY ROZUMIESZ: **1.** Dlaczego w tym zadaniu **while** jest naturalnym wyborem? **2.** Po co wymagamy dodatniej wpłaty? **3.** Które wartości trzeba sprawdzić przed wejściem do pętli?

LEKCJA 7 / 10 • 45 min

do...while: najpierw wykonaj, potem sprawdź

Po tej lekcji potrafisz: wyjaśnić różnicę między while i do...while oraz napisać pętlę wykonywaną co najmniej jeden raz.

> **ORGANIZACJA PRACY**
> Przygotuj plik **07_do_while.php** w folderze **C:\xampp\htdocs\petle_inf03** i otwórz go przez adres **http://localhost/petle_inf03/07_do_while.php**.

1. Najważniejsza różnica

Pętla **do...while** sprawdza warunek **na końcu**. Dzięki temu jej wnętrze wykona się **co najmniej jeden raz**, nawet gdy warunek od początku jest fałszywy. W odróżnieniu od **while** kończy się średnikiem po nawiasie z warunkiem.

PLIK: **07_do_while.php**

```
<?php
$i = 1;
do {
    echo "Numer: " . $i . "<br>";
    $i++;
} while ($i <= 5);  // Tu musi być średnik!
?>
```

while	do...while
Najpierw sprawdza warunek.	Najpierw wykonuje blok.
Może wykonać 0 obrotów.	Wykona co najmniej 1 obrót.
`while (warunek) { ... }`	`do { ... } while (warunek);`

2. Eksperyment: warunek od początku fałszywy

```
<?php
$i = 10;
while ($i < 5) {
    echo "while: $i<br>";
}
do {
    echo "do...while: $i<br>";
} while ($i < 5);
?>
```

Wynik: nie pojawi się żaden komunikat zaczynający się od „while:”, ale pojawi się jeden komunikat „do...while: 10”.

3. Przykład: kolejne odczyty poziomu

Załóżmy, że odczyt rozpoczyna się od 0 i po każdym obrocie przybywa 15 jednostek. Najpierw pokażemy pierwszy odczyt, a następnie sprawdzimy, czy potrzeba kolejnego.

```
<?php
$poziom = 0;
do {
    $poziom += 15;
    echo "Aktualny poziom: $poziom<br>";
} while ($poziom < 50);
?>
```

Wynik: 15, 30, 45, 60. Gdy ustawisz przed pętlą **$poziom = 50**, skrypt i tak wykona jeden obrót (65). Użyj **do...while** tylko wtedy, gdy takie zachowanie ma sens w zadaniu.

ZADANIE SAMODZIELNE | Symulacja postępu zadania

1. Stwórz plik **07_postep.php**. Ustaw **$postep = 0** i **$krok = 20**.

2. Za pomocą **do...while** wypisuj postęp pracy, aż wyniesie co najmniej **100%**.

3. Zmień krok na 30 i sprawdź, ile odczytów otrzymasz.

4. W komentarzu PHP odpowiedz, dlaczego **do...while** wykonuje pierwszy odczyt bez sprawdzenia warunku.

Wskazówki: Najpierw dodaj **$krok** do **$postep**, potem go wypisz. Warunek: kontynuuj, jeśli postęp jest mniejszy od 100.

Sprawdź: Dla kroku 20: 20, 40, 60, 80, 100. Dla kroku 30: 30, 60, 90, 120.

SPRAWDŹ, CZY ROZUMIESZ: 1. Co się stanie, gdy warunek do...while od razu jest fałszywy? **2.** Po którym znaku w instrukcji **while (...)** należy dopisać średnik w do...while? **3.** Którą pętlę wybierzesz, jeśli nie chcesz wykonać kodu ani razu przy fałszywym warunku?

LEKCJA 8 / 10 • 45 min

foreach: przechodzimy po elementach tablicy

Po tej lekcji potrafisz: utworzyć prostą tablicę, odczytać każdy jej element pętlą foreach i zbudować tabelę HTML.

> **ORGANIZACJA PRACY**
> Przygotuj plik **08_foreach.php** w folderze **C:\xampp\htdocs\petle_inf03** i otwórz go przez adres **http://localhost/petle_inf03/08_foreach.php**.

1. Co to jest tablica?

Tablica przechowuje **wiele wartości w jednej zmiennej**. Na przykład: trzy nazwy części komputerowych. Gdy chcemy wypisać wszystkie elementy, idealnie nadaje się **foreach** - nie musimy ręcznie pilnować licznika.

PLIK: **08_foreach.php**

```
<?php
$czesci = ["Procesor", "RAM", "Dysk SSD"];
foreach ($czesci as $czesc) {
    echo "<p>" . htmlspecialchars($czesc) . "</p>";
}
?>
```

$czesci to cała tablica, a **$czesc** to jeden kolejny element. W każdym obrocie zmienna **$czesc** przyjmuje następną wartość. Pętla kończy się automatycznie, gdy nie ma więcej elementów.

2. Jak czytać foreach?

```
foreach ($tablica as $element) {
    // zrób coś z kolejnym elementem tablicy
}
```

Czytaj: *dla każdego elementu tablicy wykonaj polecenia w klamrach*. **for** jest wygodniejsza przy numerowaniu 1-10; **foreach** - gdy masz listę konkretnych wartości do wyświetlenia.

3. Tablica z kluczem i wartością

```
<?php
$ceny = [
    "Mysz" => 45,
    "Klawiatura" => 90,
    "Podkładka" => 20
];
foreach ($ceny as $nazwa => $cena) {
    echo htmlspecialchars($nazwa) . ": $cena zł<br>";
}
?>
```

Strzałka **=>** łączy klucz z wartością. W tym przykładzie **$nazwa** to klucz (np. Mysz), a **$cena** to wartość (45).

4. Dane z tablicy jako tabela HTML

```
<table border="1">
  <tr><th>Produkt</th><th>Cena</th></tr>
  <?php
  $ceny = ["Mysz" => 45, "Klawiatura" => 90,
           "Podkładka" => 20];
  foreach ($ceny as $nazwa => $cena) {
      echo "<tr>";
      echo "<td>" . htmlspecialchars($nazwa) . "</td>";
      echo "<td>" . $cena . " zł</td>";
      echo "</tr>";
  }
  ?>
</table>
```

ZAPAMIĘTAJ

foreach nie służy do zwiększania licznika jak for; on po prostu bierze kolejne elementy tablicy. Nie trzeba dodawać **$i++**. Każdy element tablicy zostanie obsłużony automatycznie.

ZADANIE SAMODZIELNE | Cennik usług serwisowych

1. Stwórz plik **08_cennik.php** i tablicę: „Czyszczenie” → 40, „Instalacja” → 80, „Diagnostyka” → 30, „Wymiana dysku” → 60.

2. W pętli **foreach** wygeneruj tabelę z kolumnami **Usługa** i **Cena**.

3. Dodaj na końcu strony napis „Liczba usług: 4” (możesz użyć **count($ceny)**).

4. Dopisz piątą usługę i sprawdź, czy tabela sama się powiększa.

Wskazówki: Użyj tablicy asocjacyjnej: `$ceny = ["..." => 40];`. W pętli zastosuj dwie zmienne.

Sprawdź: Po dodaniu piątej usługi pojawia się piąty wiersz danych bez zmiany pętli.

LEKCJA 9 / 10 • 45 min

foreach i formularz: wiele wybranych opcji

Po tej lekcji potrafisz: odebrać tablicę wybranych checkboxów z formularza POST, sprawdzić wartości i wygenerować listę wyników.

> **ORGANIZACJA PRACY**
> Przygotuj plik **09_checkboxy.php** w folderze **C:\xampp\htdocs\petle_inf03** i otwórz go przez adres **http://localhost/petle_inf03/09_checkboxy.php**.

1. Dlaczego dodajemy [] do nazwy pola?

Gdy użytkownik może zaznaczyć **kilka** checkboxów, wszystkie pola mają tę samą nazwę z końcówką **[]**, np. **uslugi[]**. PHP zapisuje wybrane wartości jako tablicę: **$_POST["uslugi"]**. Wtedy można je przejść pętlą **foreach**.

2. Kompletny przykład: wybór usług

PLIK: **09_checkboxy.php**

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Wybrane usługi</title>
</head>
<body>
<h1>Wybierz usługi</h1>
<form action="" method="post">
  <label><input type="checkbox" name="uslugi[]"
         value="Czyszczenie"> Czyszczenie</label><br>
  <label><input type="checkbox" name="uslugi[]"
         value="Instalacja"> Instalacja</label><br>
  <label><input type="checkbox" name="uslugi[]"
         value="Diagnostyka"> Diagnostyka</label><br>
  <button type="submit" name="wyslij" value="1">Pokaż</button>
</form>
<?php
$dozwolone = ["Czyszczenie", "Instalacja", "Diagnostyka"];
if (isset($_POST["wyslij"])) {
    $uslugi = $_POST["uslugi"] ?? [];
    if (is_array($uslugi)) {
        echo "<h2>Wybrane usługi</h2><ul>";
        $licznik = 0;
```

```
        foreach ($uslugi as $usluga) {
            if (is_string($usluga) &&
                in_array($usluga, $dozwolone, true)) {
                echo "<li>" . htmlspecialchars($usluga) . "</li>";
                $licznik++;
            }
        }
        echo "</ul><p>Poprawnych wyborów: $licznik</p>";
    } else {
        echo "<p>Niepoprawne dane formularza.</p>";
    }
}
?>
</body>
</html>
```

Nowe elementy: ?? [] znaczy „gdy brak pola, użyj pustej tablicy”. **is_array()** sprawdza, czy otrzymano tablicę, **in_array()** sprawdza, czy wybrana wartość należy do dozwolonych, a **is_string()** sprawdza, czy jest tekstem. Licznik zwiększa się tylko dla zaakceptowanych wartości.

Jeśli checkbox nie jest zaznaczony, jego wartość nie jest wysyłana. Dlatego brak **$_POST["uslugi"]** jest normalny. Pusta tablica sprawia, że **foreach** nie wykona ani jednego obrotu.

3. Ważna różnica

Nazwa w HTML	Postać w PHP	Typowy wybór pętli
`name="kolor"`	Jeden napis: $_POST["kolor"]	Bez pętli
`name="kolory[]"`	Tablica: $_POST["kolory"]	foreach

ZADANIE SAMODZIELNE | Ankieta: technologie internetowe

1. Utwórz **09_technologie.php** i formularz POST z checkboxami: HTML, CSS, PHP, SQL.

2. Pola mają nosić nazwę **technologie[]**. Dodaj przycisk Wyślij.

3. Po wysłaniu wygeneruj listę wybranych technologii za pomocą **foreach**.

4. Pokaż komunikat, jeśli użytkownik nie zaznaczył żadnej opcji.

Wskazówki: Przenieś konstrukcję z przykładu. Gdy tablica jest pusta, sprawdź **count($technologie) === 0**.

Sprawdź: Zaznacz PHP i SQL: w liście mają pojawić się tylko te dwie pozycje. Bez wyboru pojawia się komunikat.

SPRAWDŹ, CZY ROZUMIESZ: 1. Dlaczego zapisujemy **name="uslugi[]"**? **2.** Co zwróci PHP, gdy żaden checkbox nie jest zaznaczony? **3.** Dlaczego sprawdzamy, czy otrzymana wartość należy do dozwolonych?

LEKCJA 10 / 10 • 45 min

Praktyka INF.03: while i dane z MySQL

Po tej lekcji potrafisz: połączyć PHP z MySQL, przejść pętlą po rekordach SELECT i przedstawić dane w HTML.

> **ORGANIZACJA PRACY**
> Przygotuj plik **10_produkty.php** w folderze **C:\xampp\htdocs\petle_inf03** i otwórz go przez adres **http://localhost/petle_inf03/10_produkty.php**.

1. Dlaczego pętle są ważne na egzaminie?

W zadaniu praktycznym może pojawić się polecenie, aby **wyświetlić wszystkie rekordy** zwrócone przez zapytanie SQL. Ich liczba nie zawsze jest znana. Pętla **while** pozwala pobierać kolejne wiersze wyniku tak długo, aż skończą się dane.

2. Przygotuj prostą bazę w phpMyAdmin

W XAMPP uruchom **Apache** i **MySQL**. Otwórz **http://localhost/phpmyadmin**, wejdź w zakładkę SQL i wykonaj poniższy kod. To lokalne dane testowe. Przed utworzeniem bazy upewnij się, że nie masz już ważnej bazy o tej samej nazwie.

PLIK: **SQL - baza ćwiczeniowa**

```
CREATE DATABASE IF NOT EXISTS petle_inf03
  CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE petle_inf03;
CREATE TABLE IF NOT EXISTS produkty (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nazwa VARCHAR(60) NOT NULL,
    cena INT NOT NULL
);
INSERT INTO produkty (nazwa, cena) VALUES
('Mysz', 45),
('Klawiatura', 90),
('Podkładka', 20),
('Pendrive', 35);
```

Jeżeli uruchomisz **INSERT** drugi raz, pojawią się duplikaty. Kod uruchom raz na pustej tabeli. Sprawdź zakładkę **Przeglądaj**: powinny pojawić się cztery rekordy.

3. Pełny skrypt odczytu bazy

PLIK: **10_produkty.php**

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <title>Produkty z bazy</title>
</head>
<body>
<h1>Lista produktów</h1>
<table border="1">
<tr><th>ID</th><th>Nazwa</th><th>Cena</th></tr>
<?php
$polaczenie = mysqli_connect("localhost", "root", "",
                             "petle_inf03");
if (!$polaczenie) {
    exit("Błąd połączenia z bazą danych.");
}
mysqli_set_charset($polaczenie, "utf8mb4");
$sql = "SELECT id, nazwa, cena FROM produkty ORDER BY id";
$wynik = mysqli_query($polaczenie, $sql);
if ($wynik) {
    while ($wiersz = mysqli_fetch_assoc($wynik)) {
        $id = (int) $wiersz["id"];
        $nazwa = htmlspecialchars($wiersz["nazwa"]);
        $cena = (int) $wiersz["cena"];
```

```
        echo "<tr><td>$id</td><td>$nazwa</td>";
        echo "<td>$cena zł</td></tr>";
    }
} else {
    echo '<tr><td colspan="3">Błąd zapytania.</td></tr>';
}
mysqli_close($polaczenie);
?>
</table>
</body>
</html>
```

4. Jak działa while w połączeniu z SQL?

Funkcja / instrukcja	Działanie
mysqli_connect	Otwiera połączenie z bazą na serwerze MySQL/MariaDB.
mysqli_query	Wysyła zapytanie SELECT i zwraca wynik.
mysqli_fetch_assoc	Pobiera kolejny rekord jako tablicę z nazwami kolumn; po końcu zwraca null.
while (...)	Wykonuje blok dla każdego znalezionego rekordu.
mysqli_close	Zamyka połączenie.

W tym wzorcu nie ma **$i++**, ponieważ **mysqli_fetch_assoc()** samo przechodzi do kolejnego rekordu przy każdym wywołaniu. Gdy rekordy się skończą, warunek pętli przestaje być spełniony.

BEZPIECZEŃSTWO W ZADANIACH INF.03

Nie dołączaj niesprawdzonych danych z formularza bezpośrednio do zapytań SQL. W tym przykładzie zapytanie jest **stałe** (bez danych od użytkownika), dlatego nie wymaga parametrów. Zwracane napisy wyświetlamy przez **htmlspecialchars**.

5. Samodzielne mini-zadanie w stylu INF.03

ZADANIE SAMODZIELNE | Wykaz urządzeń

1. W phpMyAdmin utwórz tabelę **urzadzenia** z polami: **id INT AUTO_INCREMENT PRIMARY KEY**, **nazwa VARCHAR(60)**, **ilosc INT**.

2. Dodaj cztery rekordy: Laptop (8), Monitor (10), Drukarka (2), Router (3).

3. Przygotuj **10_urzadzenia.php**, który pobierze dane zapytaniem SELECT i wyświetli je w tabeli HTML.

4. Każdy rekord ma być wypisywany wewnątrz **while (mysqli_fetch_assoc(...))**. Dodaj nagłówek oraz stopkę strony.

Wskazówki: Zmień w przykładzie nazwę tabeli, zapytanie i nazwę kolumny cena na ilosc. Pamiętaj o prawidłowej bazie i połączeniu.

Sprawdź: W przeglądarce pojawiają się cztery urządzenia i ich ilości. Dodanie piątego rekordu w bazie automatycznie dodaje piąty wiersz na stronie.

6. Ostatnia próba: mini-projekt do powtórki

Jeśli skończysz wcześniej lub nauczyciel przeznaczy dodatkowy czas: utwórz formularz POST „Ile etykiet wydrukować?” (1-10), który użyje pętli **for** do wygenerowania numerowanych etykiet. Obok pokaż listę trzech przykładowych produktów z tablicy, używając **foreach**. Rozwiązanie ma być w jednym pliku PHP.

POWTÓRKA • INF.03

Szybka ściąga: którą pętlę wybrać?

Rodzaj	Kiedy stosować	Schemat
for	Znasz liczbę powtórzeń.	`for ($i=1; $i<=5; $i++)`
while	Powtarzasz do spełnienia warunku.	`while ($x < 10)`
do...while	Kod musi wykonać się przynajmniej raz.	`do { ... } while ($x<10);`
foreach	Przechodzisz po elementach tablicy.	`foreach ($a as $x)`

Najczęstsze błędy uczniów i szybka naprawa

Objaw	Przyczyna	Jak naprawić
Pętla nie kończy pracy.	Brak zmiany licznika lub nieosiągalny koniec.	Sprawdź krok, zakres i warunek.
Wynik zawiera 9 zamiast 10 pozycji.	Pomyłka w < oraz <=.	Rozpisz wartości licznika na kartce.
Notice/Warning: Undefined array key.	Formularz nie przesłał pola.	Sprawdź isset; przy checkbox użyj ?? [].
Po otwarciu widać kod PHP.	Plik otwarto bez serwera.	Użyj localhost i uruchom Apache.
Nie widać polskich znaków.	Brak UTF-8.	Ustaw UTF-8 w edytorze i meta charset.
PHP nie pokazuje treści formularza.	Brakuje name lub mylisz GET z POST.	Sprawdź name, method i $_POST.
Baza jest pusta mimo zapytania.	Błędne połączenie, tabela lub SELECT.	Sprawdź phpMyAdmin i wynik zapytania.

Sprawdzian próbny – odpowiedz bez uruchamiania kodu

- Ile razy wykona się pętla `for ($i = 0; $i < 12; $i += 3)`?
- Jakie liczby wypisze pętla `for ($i = 4; $i >= 1; $i--)`?
- Czym różni się zachowanie **while** od **do...while** przy fałszywym warunku?
- Jak odebrać w PHP dane z pola formularza z atrybutem **name="ilosc"** i metodą POST?
- Jak nazywa się instrukcja PHP przechodząca po elementach tablicy?
- Jaką pętlą wypiszesz nieznaną z góry liczbę rekordów wyniku SELECT?
- Po co stosować **htmlspecialchars** przy wyświetlaniu danych użytkownika?

- Co zmienić, jeśli pętla **while** wypisuje nieskończenie wiele wartości?

Lista kontrolna przed oddaniem pracy

- ☐ Plik kończy się na **.php** i działa przez **http://localhost/...**.
- ☐ Dokument HTML ma **<!DOCTYPE html>**, **lang="pl"**, **<meta charset="UTF-8">** i sensowny **<title>**.
- ☐ W pętli jest poprawny warunek końca, a liczba powtórzeń jest prawidłowa.
- ☐ Formularz ma właściwe **method**, pola mają **name**, a PHP sprawdza dane przed użyciem.
- ☐ Teksty użytkownika są bezpiecznie wyświetlane w HTML.
- ☐ W tabelach HTML znajdują się prawidłowe **<tr>**, **<th>** i **<td>**.
- ☐ Przetestowano poprawne dane, skrajne wartości oraz dane błędne.

ŹRÓDŁA I ZAKRES

Zgodność z INF.03 i źródła

Zakres przyjęto na podstawie jednostki **INF.03.5 – Programowanie aplikacji internetowych**, w szczególności stosowania instrukcji sterujących, tworzenia skryptów w PHP, przesyłania danych formularza oraz używania bibliotek do obsługi baz danych. Wykorzystano przykłady zgodne z pracą w XAMPP i z typowymi zadaniami egzaminacyjnymi. Materiał jest **autorską instrukcją dydaktyczną**, a nie oficjalnym arkuszem CKE.

Bibliografia / dokumentacja

1. Ministerstwo Edukacji – Podstawa programowa technik programista (INF.03.5)
https://infozawodowe.men.gov.pl/image/professionCoreCurriculum/technik-programista_d4f979c1441b4c648dd1db5cfdf1bccb.pdf

2. Centralna Komisja Egzaminacyjna – arkusz praktyczny INF.03, czerwiec 2025 (przykład)
https://arkusze.pl/zawodowy/inf03-2025-czerwiec-egzamin-zawodowy-praktyczny.pdf

3. Dokumentacja PHP: for
https://www.php.net/manual/en/control-structures.for.php

4. Dokumentacja PHP: while
https://www.php.net/manual/en/control-structures.while.php

5. Dokumentacja PHP: do-while
https://www.php.net/manual/en/control-structures.do.while.php

6. Dokumentacja PHP: foreach
https://www.php.net/manual/en/control-structures.foreach.php

7. Dokumentacja PHP: isset
https://www.php.net/manual/en/function.isset.php

8. Dokumentacja PHP: htmlspecialchars
https://www.php.net/manual/en/function.htmlspecialchars.php

Dostęp: 8 listopada 2026. Treść ćwiczeń jest nowa i została przygotowana do stopniowej nauki w pracowni. Przykład SQL i nazwy danych są fikcyjne. Przed pracą z bazą na rzeczywistym serwerze należy wykonać kopię zapasową.