SUBHARAM GOVT.DEGREE COLLEGE –

PUNGANUR

Student Study Project – Department of Computer Science

S. No	Year	Name of the Student	Program	Name of the Mentor	Department	Project Title
1	2020-21	N.HARISH	III B Sc	Smt. M.SRAVANI	Computer Science	Creating Numeric Arrays in PHP
2	2020-21	V.SREEVANI	III B Sc	Smt. M.SRAVANI	Computer Science	My SQL Database using in PHP
3	2020-21	M.NAGESH	III B Sc	Smt. M.SRAVANI	Computer Science	My SQL vs My SQLi
4	2021-22	M.SIVARAJ	III B Sc	Smt. M.SRAVANI	Computer Science	Forms in CSS
5	2021-22	A.BHANU PRAKASH	II B Sc	Smt. M.SRAVANI	Computer Science	HTML- Looping Statements
6	2022-23	M.MAIYARSON	III B Sc	Smt. M.SRAVANI	Computer Science	Data types in PHP
7	2022-23	ASWINI	III B Sc	Smt. M.SRAVANI	Computer Science	Different Networks - Types
8	2022-23	C.SOWMYA	III B Sc	Smt. M.SRAVANI	Computer Science	Analog & Digital Signals
9	2023-24	D.THEJA	III B Sc	A.KARTHIK	Computer Science	OOPs Concepts
10	2023-24	K.CHANDANA	III B Sc	A.KARTHIK	Computer Science	In object oriented programming – Stack & Queue

SUBHARAM GOVT. DEGREE COLLEGE

PUNGANUR

Department of Computer Science

Student Study Project

Year: 2020-21

Topic: Creating Numeric Arrays in PHP

Submitted by: N. HAREESH

Supervised by: Smt. M. Sravani,

Lecturer in Computer Science

SRGDC, PUNGANUR

Numeric Array:

1. In numeric array each element having numeric key associated with it that is staring from 0.

2. You can use array() function to create array.

3. The general syntax is given below:

```
$array_name = array (value1, value2 .... valueN);
```

4. for example:

```
<?php
    $myarray = array(10,20,30);
    print_r ($myarray);
?>
```

output:

```
array([0]=>10 [1]=>20 [2]=>30)
```

You can refer to individual element of an array in PHP script using its key value as shown below:

```
<?php
    $my array = array (10, 20, 30);
    echo $my array[1];
?>
```

it will display

20

In numeric Array you can use for, while or do while loop to iterate through each element in array because in numeric array key values are consecutive

```
<?php
    $myarray = array(10, 20, 30);
     for ($i=0; $i<3; $i++)
{ echo $myarray[$i]."<br>";
}
?>
```

output:

10
20
30

SUBHARAM GOVT. DEGREE COLLEGE

PUNGANUR

Department of Computer Science

Student Study Project

Year: 2021-22

Topic: Forms in *CSS*

Submitted by: M.SHIVARAJ

Supervised by: Smt. M. Sravani,

Lecturer in Computer Science

SRGDC, PUNGANUR

2021-22

S.R.G.D.C,

Punganur

Computer Science

III rd B.S.C. [M.S.cs]

Semester - V

Web interface Designing Technologies

Project Work

by

M. Sivaraj

Topic:-

forms in CSS

Write about forms in CSS?

Styling Input fields

Width :- Use the width property to determine the width of the input field:

Example:-

```
input {
    Width : 100%;
}
```

The example above applies to all <input> elements. If you only want to style a specific input type, your can use attribute selectors:

input [type = text] - Will only select text fields

input [type = password] - Will only select password fields.

input [type = number] - Will only select number fields etc.....

1. Padded Inputs:-

Use the padding property to add space inside the text field.

```
Ex: input [type = text] {
        Width : 100%;
        Padding : 12px 20px;
        margin : 8px 0;
    }
```

2. Bordered Inputs:-

Use the border property to change the border size and colour, and use the border radius property to add rounded corners.

```
Ex:- input [type = text] {
        border : none;
        border - bottom : 2px solid red;
    }
```

3. Colored Inputs:-

Use the back ground - color property to add a back ground color to the input, and the color property to change the text color:

```
Ex:- input [type = text] {
        back ground - color : blue;
        color : white;
    }
```

4. Focused Inputs:- By default, some browsers will add a blue out line around the input when it gets focus (clicked on). You can remove this behavior by adding outline: none; to the input.

Use the :focus selector to do something with the input field when it gets focus:

Ex:-

```
input [type =text]: focus {
  border : 3px solid # 555;
  }
```

5. Input with Icon / Image: If you want an icon inside the input use the background-image property and position it with the background-position property.

Ex:-

```
input [type = text] {
  background-color : white;
  background-image : url ('search icon.png');
  background-position: 10px 10px;
  background-repeat : no-repeat;
  Padding-left : 40 px;
  }
```

6. Animated Search Input: In this example We use the Css transition property to animate the width of the search input when it gets focus.

Ex:-

```
input [type = text] {
  transition : width 0.4s ease-in-out;
  }
  input [type =text]: focus {
   width : 100%
  }
```

7. Styling Input Buttons: Like most elements, buttons have default styles. Use the background-color property to change the background colour of a Button.

Ex:-

```
input [type=button], input [type = submit], input [type=reset]
  {
   background-color : red;
   border : none;
   color : white;
   Padding : 16px 32px;
   margin : 4px 2px;
   cursor : pointer; }
```

SUBHARAM GOVT. DEGREE COLLEGE

PUNGANUR

Department of Computer Science

Student Study Project

Year: 2022-23

Topic: Data Types in PHP

Submitted by: M. MAIYARSON

Supervised by: Smt. M. Sravani,
Lecturer in Computer Science
SRGDC, PUNGANUR

2022-23

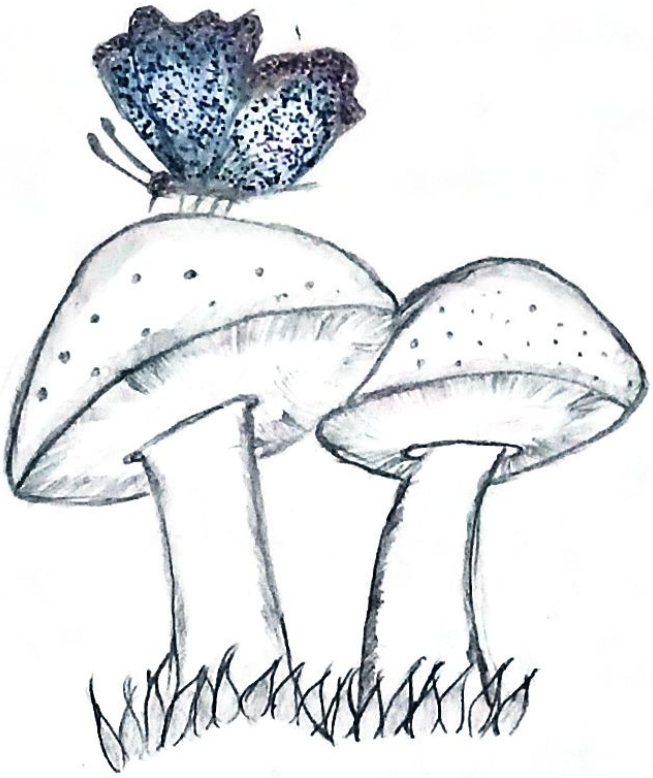
Subharam Giri

Degree College

Computer science :

project work

Title : Data types in PHP



PHP supports a variety of data types. Here are some of the most commonly used data types in PHP

* Integer : Represent whole numbers, e.g; '$integer = 42'.
* Float : Represent numbers with decimal points, e.g; '$float = 3.14'.
* string : Represent sequence of characters e.g; '$string = "Hello, world!";
* Boolean : Represent true or false values, e.g; '$isTrue = true;
* Array : Represent ordered, indexed, or associative collection of data, e.g; '$colors = ["red", "green", "blue"]'.
* object : Represent instances of user-defined classes or built-in classes, e.g; '$obj = new myclass();
* NULL : Represent a variable with no value or undefined. e.g; '$var = null;

* Compound Types : These include arrays object, which can store multiple values and have their own methods and properties.

* Mixed (PHP 8.0+) : A union type that can hold values of multiple types.

* Union Types (PHP 8.0+) : Allows a variable to accept multiple data types.

* Never (PHP 8.0+) : Represent a type that should never occur.

* void (PHP 7.1+) : Represent the absence of a return value in functions.

SUBHARAM GOVT. DEGREE COLLEGE

PUNGANUR

Department of Computer Science

Student Study Project

Year: 2023-24

Topic: In Object Oriented Programming –
Stack & Queue

Submitted by: K. CHANDANA

Supervised by: Sri. A. KARTHIK,
Lecturer in Computer Science
SRGDC, PUNGANUR

Subharam
Degree
College

computer Science:

Project work - 2023-24

Topic : stack and queue

K. chandana
III. B.Sc

In object-oriented Programming (OOP), "stack" and "queue" can refer to two different concepts:

1.* stack in OOP:*

In the content of OOP, a "stack" might refer to a data structure that follows the last in, first out (LIFO) Principle. it's a collection of elements with two main operations: Push (to add an element to the top) and POP (to remove the top element). stacks are often used to manage function calls, expressions evaluation, and undo machanisms.

Example (Pseudo-code):

```
CPP
  class stack {

Private:
    int data [max_SIZE];
    int top;

public:
    stack() {top = -1;}
    void push (int element) {data [++top] = element;}
    int POP () {return data [top --];}

  };
```

2.* queue in ODP:*- in OOP, a "queue" might refer to a data structure that follows the first in, first out (FIFO) Principle, it's a collection of

Elements with two main operations: enqueue (to add an elements to the back) and dequeue (to remove the front element). Queues are often used in Scenarios like task scheduling, Point job management, and breadth-first search -es.

Example (Pseudo-code):

CPP

```
class Queue {
Private:
    int data [max-size];
    int front, rear;
Public:
   Queue () { front = rear = -1; }
   void enqueue (int element) { data [++rear] =
                                element; }
   int dequeue () { return data [++ front]; }
};
```

These are simplified examples to illustrate the concepts. In real-world scenarios, you might use the built-in implementations provided by Programming Languages (like c++'s '<stack>' and '<queue'), which are more versatile and efficient. These data structures play a crucial role in organizing and managing data in various applications.