



**SHRI VIDHYABHARATHI MATRIC HR.SEC.SCHOOL**  
**SAKKARAMPALAYAM, AGARAM (PO) ELACHIPALAYAM**  
**TIRUCHENGODE(TK), NAMAKKAL (DT) PIN-637202**

**Cell : 99655-31727, 94432-31727**

**+2 -COMPUTER SCIENCE PUBLIC EXAMINATION-2023**

**ANSWER-KEY**

**STD: XII**

**MARKS : 70**

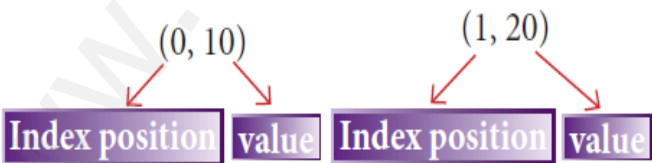
**SUBJECT: COMPUTER SCIENCE**

Q. NO	CONTENT	MARK
	<b>PART-I</b>	<b>10X1=10</b>
<b>I.</b>	<b>CHOOSE THE CORRECT ANSWER:</b>	
1	b)Public Member	<b>1</b>
2	c)Operator	<b>1</b>
3	b)Subroutine	<b>1</b>
4	b)3	<b>1</b>
5	d).	<b>1</b>
6	a)Hierarchical	<b>1</b>
7	b)+	<b>1</b>
8	b)Wrapping	<b>1</b>
9	b)DROP TABLE	<b>1</b>
10	b)MAX()	<b>1</b>
11	a)Concrete data type	<b>1</b>
12	d)Recursion	<b>1</b>
13	d)Binary mode	<b>1</b>
14	a)Memoization	<b>1</b>
15	c){1,3,6,9}	<b>1</b>
<b>II.</b>	<b>PART - II</b>	<b>6X2=12</b>
16	<ul style="list-style-type: none"> <li>A tuple is a comma-separated sequence of values surrounded with parentheses. Tuple is similar to a list.</li> <li>The difference between the two is that you cannot change the elements of a tuple once it is assigned whereas in a list, elements can be changed.</li> <li><b>Example:</b> colour= ('red', 'blue', 'Green')</li> </ul>	<b>2</b>
17	<ul style="list-style-type: none"> <li>Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program.</li> <li>In other words which parts our program can see or use it.</li> </ul>	<b>2</b>

18	<ul style="list-style-type: none"> <li>Python will not allow deleting a particular character in a string. Whereas you can remove entire string, variable using <b>del</b> command.</li> </ul> <p><b>Example:</b></p> <pre>&gt;&gt;&gt; str1="How about you" &gt;&gt;&gt; print (str1) How about you &gt;&gt;&gt; del str1 &gt;&gt;&gt; print (str1) Traceback (most recent call last):   File "&lt;pyshell#14&gt;", line 1, in &lt;module&gt;     print (str1) NameError: name 'str1' is not defined</pre>	2
19	<ul style="list-style-type: none"> <li>for loop uses the range() function in the sequence to specify the initial, final and increment values.</li> <li>range() generates a list of values starting from <b>start</b> till <b>stop-1</b>.</li> </ul> <p><b>The syntax of range() is as follows:</b></p> <pre>range (start, stop, [step])</pre> <p>Where, start – refers to the initial value  stop – refers to the final value  step – refers to increment value, this is optional part.</p> <p><b>Example:</b> range (1, 30, 1) will start the range of values from 1 and end at 29</p>	2
20	<ul style="list-style-type: none"> <li>Class is the main building block in Python. A class is a way of binding data members and member function together. Class is a template for the object.</li> <li>In python the class is defined by using the keyword class.</li> <li>Every class has unique name followed by a colon(:)</li> </ul> <p><b>Syntax:</b></p> <pre>class class_name:     statement_1     statement_2     .....     .....     statement_n</pre> <p><b>Example:</b></p> <pre>Class student:     X,y=10,20</pre>	2
21	<ul style="list-style-type: none"> <li>A Data Manipulation Language (DML) is a computer programming language used for adding (inserting), removing (deleting), and modifying (updating) data in a database. In SQL, the data manipulation language comprises the SQL-data change statements, which modify stored data but not the schema of the database table.</li> </ul>	2

	<ul style="list-style-type: none"> <li>After the database schema has been specified and the database has been created, the data can be manipulated using a set of procedures which are expressed by DML.</li> </ul>	
22	<ul style="list-style-type: none"> <li>The default mode of csv file in reading and writing is text mode.</li> </ul>	2
23.	<ul style="list-style-type: none"> <li>Charts</li> <li>Table</li> <li>Graphs</li> <li>Maps</li> <li>Infographics</li> <li>Dashboards</li> </ul>	2
24	<b>[1,4,9,16,25,36,49,64,81,100] /error (Based on Govt key)</b>	
III.	<b>PART - III</b>	<b>6X3=18</b>
25	<ul style="list-style-type: none"> <li>The class template specifies the interfaces to enable an object to be created and operated properly.</li> <li>An object's attributes and behavior is controlled by sending functions to the object.</li> </ul>	
26	<ul style="list-style-type: none"> <li>Dynamic programming is an algorithmic design method that can be used when the solution to a problem can be viewed as the result of a sequence of decisions.</li> <li>Dynamic programming approach is similar to divide and conquer.</li> <li>The given problem will be divided into smaller overlapping sub-problems.</li> <li>Dynamic programming is used whenever problems can be divided into similar sub-problems. so that their results can be re-used to complete the process.</li> <li>Dynamic programming approaches are used to find the solution in optimized way. For every inner sub problem, dynamic algorithm will try to check the results of the previously solved sub-problems. The solutions of overlapped sub-problems are combined in order to get the better solution.</li> </ul>	
27	<ul style="list-style-type: none"> <li>Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false. It simply allows testing a condition in a single line replacing the multiline if-else making the code compact.</li> </ul> <p><b>Syntax: Variable Name = [on_true] if [Test expression] else [on_false]</b></p> <p><b>Example:</b> min= 50 if 49&lt;50 else 70</p>	3

28	while <condition>: statements block 1 [else: statements block2]		3
29	<b>ceil ( )</b>	<b>floor( )</b>	2
	It returns the smallest integer greater than or equal to x.	It returns the largest integer less than or equal to x.	
	<b>Syntax:</b> math.ceil (x)	<b>Syntax:</b> math.floor (x)	1
	<b>Example:</b> x= 26.7 y= -26.7 z= -23.2 print (math.ceil (x)) <b>Output:</b> 27	<b>Example:</b> x=26.7 y=-26.7 z=-23.2 print (math.floor (x)) <b>Output:</b> 26	
30	• The main difference between the csv.reader() and DictReader() is in simple terms csv. reader and csv.writer work with <b>list/tuple</b> , while csv.DictReader and csv.DictWriter work with dictionary.		3
31	<b>fetchone()</b>	<b>fetchmany()</b>	
	returns the next row of a query result set or None in case there is no row left.	returns the next number of rows (n) of the result set.	
	import sqlite3 connection=sqlite3.connect("Academy.db") cursor=connection.cursor() cursor.execute("SELECT*FROMstudent") print("fetching all records one by one:") result = cursor.fetchone() while result is not None: print(result) result = cursor.fetchone()	import sqlite3 connection=sqlite3.connect("Academy.db") cursor=connection.cursor() cursor.execute("SELECT*FROMstudent") print("fetching all records one by one:") result = cursor.fetchmany() while result is not None: print(result) result = cursor.fetchmany()	
32	<pre> str1=input ("Enter string") str2= '' index=len(str1) for i in str1:     str2=str1[0:index]     index -=1     print(str2)                     </pre>		3
33	1. Type the C++ program 2. Type the Python program and save it. 3. Type the command Python pali.py -i pali_cpp (or) Type the command Python filename -input mode c++ file name.		3

III.	Part-IV	5X5=25
34 a.	<ul style="list-style-type: none"> <li>• To enable us to implement the concrete level of our data abstraction, Some languages like Python provides a compound structure called Pair which is made up of list or Tuple. The first way to implement pairs is with the List construct.</li> <li>• List is constructed by placing expressions within square brackets separated by commas. Such an expression is called a list literal. List can store multiple values. Each value can be of any type and can even be another list.</li> </ul> <p><b>Example</b> for List is [10, 20].</p> <ul style="list-style-type: none"> <li>• The elements of a list can be accessed in two ways. The first way is via our familiar method of multiple assignment, which unpacks a list into its elements and binds each element to a different name.</li> </ul> <pre>lst := [10, 20] x, y := lst</pre> <ul style="list-style-type: none"> <li>• In the above example <b>x</b> will become 10 and <b>y</b> will become 20.</li> <li>• A second method for accessing the elements in a list is by the element selection operator, also expressed using square brackets. Unlike a list literal, a square- brackets expression directly following another expression does not evaluate to a list value, but instead selects an element from the value of the preceding expression.</li> </ul> <pre>lst[0] 10 lst[1] 20</pre> <ul style="list-style-type: none"> <li>• In both the example mentioned above mathematically we can represent list similar to a set.</li> </ul> <p>lst[(0, 10), (1, 20)] – where</p> <div style="text-align: center;">  </div> <p>Any way of bundling two values together into one can be considered as a pair. Lists are a common method to do so. Therefore List can be called as Pairs.</p>	5
34 b)	<ul style="list-style-type: none"> <li>• Linear search also called sequential search is a sequential method for finding a particular value in a list. This method checks the search element with each element in sequence until the desired element is found or the list is exhausted. In this searching algorithm, list need not be ordered.</li> </ul> <p><b>Pseudo code</b></p> <p>(i) Traverse the array using for loop</p> <p>(ii) In every iteration, compare the target search key value with the current</p>	



	<p>value of the list.</p> <ul style="list-style-type: none"><li>• If the values match, display the current index and value of the array</li><li>• If the values do not match, move on to the next array element.</li></ul> <p>(iii) If no match is found, display the search element not found.</p> <p>To search the number 25 in the array given below, linear search will go step by step in a sequential order starting from the first element in the given array if the search element is found that index is returned otherwise the search is continued till the last index of the array. In this example number 25 is found at index number 3.</p> <table border="1"><tr><td>index</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>values</td><td>10</td><td>12</td><td>20</td><td>25</td><td>30</td></tr></table> <p><b>Example 1:</b> Input: values[] = {5, 34, 65, 12, 77, 35} target = 77 Output: 4</p>	index	0	1	2	3	4	values	10	12	20	25	30	<p>5</p> <p>2.5</p>
index	0	1	2	3	4									
values	10	12	20	25	30									
35 a)	<ul style="list-style-type: none"><li>• Python breaks each logical line into a sequence of elementary lexical components known as <b>Tokens</b>. The normal token types are.<ol style="list-style-type: none"><li>1) Identifiers,</li><li>2) Keywords,</li><li>3) Operators,</li><li>4) Delimiters and</li><li>5) Literals.</li></ol></li></ul> <p>Whitespace separation is necessary between tokens, identifiers or keywords.</p> <p><b>Identifiers :</b></p> <ul style="list-style-type: none"><li>• An Identifier is a name used to identify a variable, function, class, module or object.</li><li>• An identifier must start with an alphabet (A..Z or a..z) or underscore ( _ ).</li><li>• Identifiers may contain digits (0 .. 9)</li><li>• Python identifiers are case sensitive i.e. uppercase and lowercase letters are distinct.</li><li>• Identifiers must not be a <b>python</b> keyword.</li><li>• Python does not allow punctuation character such as %, \$, @ etc., within identifiers.</li></ul> <p><b>Example of valid identifiers:</b> Sum, total_marks, regno, num1</p>	<p>5</p>												

### Keywords:

- Keywords are special words used by Python interpreter to recognize the structure of program.
- As these words have specific meaning for interpreter, they cannot be used for any other purpose.

**Example: class, for, and**

### Operators :

- In computer programming languages operators are special symbols which represent computations, conditional matching etc.
- The value of an operator used is called **operands**.
- Operators are categorized as Arithmetic, Relational, Logical, Assignment etc.
- Value and variables when used with operator are known as **operands**.

#### (i) Arithmetic operators:

- An arithmetic operator is a mathematical operator that takes two operands and performs a calculation on them.
- They are used for simple arithmetic. Most computer languages contain a set of such operators that can be used within equations to perform different types of sequential calculations.

**Example: >>>a+b**

#### (ii) Relational or Comparative operators:

- A Relational operator is also called as **Comparative** operator which checks the relationship between two operands.
- If the relation is true, it returns **True**; otherwise it returns **False**.

**Example: >>> a==b**

#### (iii) Logical operators:

- In python, Logical operators are used to perform logical operations on the given relational expressions.
- There are three logical operators they are **and**, **or** and **not**.

>>> a>b or a==b

#### (iv) Assignment operators:

- In Python, = is a simple assignment operator to assign values to variable. Let **a = 5** and **b = 10** assigns the value 5 to **a** and 10 to **b** these two assignment statement can also be given as **a, b=5,10** that assigns the value 5 and 10 on the right to the variables a and b respectively.
- There are various compound operators in Python like +=, -=, \*=, /=, %=, \*\*= and //= are also available.

**Example: >>> x=10**

	<p><b>(v) Conditional operator:</b></p> <ul style="list-style-type: none"> <li>Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false.</li> <li>It simply allows testing a condition in a single line replacing the multiline if-else making the code compact.</li> </ul> <p><b>Syntax:</b>  <b>Variable Name = [on_true] if [Test expression] else [on_false]</b></p> <p><b>Example:</b>  min= 50 if 49&lt;50 else 70 # min = 50  min= 50 if 49&gt;50 else 70 # min = 70</p> <p><b>Delimiters :</b></p> <ul style="list-style-type: none"> <li>Python uses the symbols and symbol combinations as delimiters in expressions, lists, dictionaries and strings.</li> </ul> <p><b>Example: ( , )</b></p> <p><b>Literals:</b> Literal is a raw data given in a variable or constant. In Python, there are various types of literals.</p> <ol style="list-style-type: none"> <li>1) Numeric</li> <li>2) String</li> <li>3) Boolean</li> </ol>	
b)	<p><b>(a) id()</b></p> <p><b>(b) chr()</b></p> <p><b>(c) round()</b></p> <p><b>(d) type()</b></p> <p><b>(e) pow()</b></p> <p><b>(a) id()</b></p> <ul style="list-style-type: none"> <li><b>Description:</b> id ( ) Return the “identity” of an object. i.e. the address of the object in memory. <b>Note:</b> the address of x and y may differ in your system.</li> <li><b>Syntax:</b> id (object)</li> </ul> <p><b>Example:</b>  x=15  y='a'  print ('address of x is :',id (x))  print ('address of y is :',id (y))</p> <p><b>Output:</b>  address of x is : 1357486752  address of y is : 13480736</p> <p><b>(b) chr ()</b></p> <ul style="list-style-type: none"> <li><b>Description:</b> Returns the Unicode character for the given ASCII value.</li> </ul>	5



- This function is inverse of ord() function.

- **Syntax:** chr (i)

- **Example:** c=65

d=43

print (chr (c))

print(chr (d))

**Output:**

A

+

**(c) round()**

- **Description:** Returns the nearest integer to its input.

1. First argument (number) is used to specify the value to be rounded

2. Second argument (n digits) is used to specify the number of decimal digits desired after rounding.

- **Syntax:** round (number [,n digits])

- **Example:** x= 17.9

y= 22.2

z= -18.3

print ('x value is rounded to', round (x))

print ('y value is rounded to', round (y))

print ('z value is rounded to', round (z))

**Output:1**

x value is rounded to 18

y value is rounded to 22

z value is rounded to -18

n1=17.89

print (round (n1,0))

print (round (n1,1))

print (round (n1,2))

**(d) type()**

- **Description:** Returns the type of object for the given single object.

**Note:** This function used with single object parameter.

- **Syntax:** type (object)

- **Example:**

x= 15.2

y= 'a'

s= True

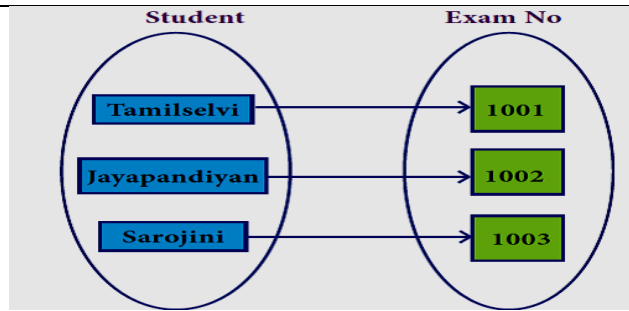
print (type (x))

print (type (y))

print (type (s))

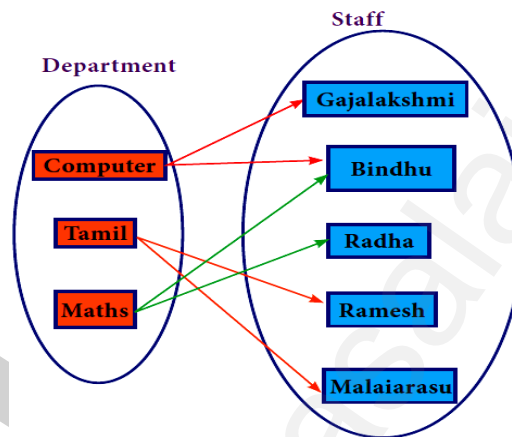
**Output:**

	<pre> &lt;class 'float'&gt; &lt;class 'str'&gt; &lt;class 'bool'&gt;  (e) pow() • <b>Description:</b> Returns the computation of ab i.e. (a**b) a raised to the power of b. • <b>Syntax:</b> pow (a,b) • <b>Example:</b> a= 5                b= 2                c= 3.0                print (pow (a,b))                print (pow (a,c))                print (pow (a+b,3))  <b>Output:</b> 25 125.0 343 </pre>	
36 a)	<ul style="list-style-type: none"> <li>In Python, a tuple can be defined inside another tuple; called Nested tuple. In a nested tuple, each tuple is considered as an element. The for loop will be useful to access all the elements in a nested tuple.</li> </ul> <p><b>Example:</b></p> <pre> Toppers = (("Vinodini", "XII-F", 98.7), ("Soundarya", "XII-H", 97.5),            ("Tharani", "XII-F", 95.3), ("Saisri", "XII-G", 93.8)) for i in Toppers:     print(i)  <b>Output:</b> ('Vinodini', 'XII-F', 98.7) ('Soundarya', 'XII-H', 97.5) ('Tharani', 'XII-F', 95.3) ('Saisri', 'XII-G', 93.8) </pre>	
b)	<ol style="list-style-type: none"> <li>One-to-One Relationship</li> <li>One-to-Many Relationship</li> <li>Many-to-One Relationship</li> <li>Many-to-Many Relationship</li> </ol> <p><b>One-to-One Relationship:</b></p> <ul style="list-style-type: none"> <li>In One-to-One Relationship, one entity is related with only one other entity. One row in a table is linked with only one row in another table and vice versa.</li> <li>For example: A student can have only one exam number.</li> </ul>	



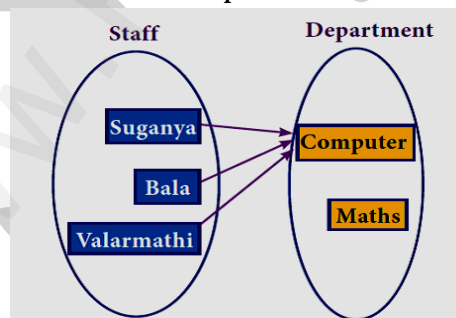
### One-to-Many Relationship:

- In One-to-Many relationship, one entity is related to many other entities. One row in a table A is linked to many rows in a table B, but one row in a table B is linked to only one row in table A.
- For example: One Department has many staff members.



### Many-to-One Relationship:

- In Many-to-One Relationship, many entities can be related with only one in the other entity. For example: A number of staff members working in one Department. Multiple rows in staff members table is related with only one row in Department table.



### Many-to-Many Relationship:

- A many-to-many relationship occurs when multiple records in a table are associated with multiple records in another table.

#### Example 1: Customers and Product

Customers can purchase various products and Products can be purchased by many customers

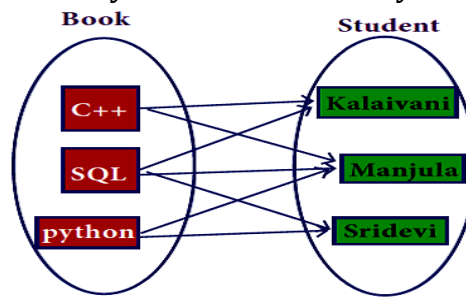
#### Example 2: Students and Courses

A student can register for many Courses and a Course may include

many students

### Example 3: Books and Student.

Many Books in a Library are issued to many students.



37. This method parses command-line options and parameter list. Following is the **syntax**:

a)

**<opts>,<args>=getopt.getopt(argv, options, [long\_options])**

Here is the detail of the parameters –

- ❖ **argv** – This is the argument list of values to be parsed (splited). In our program the complete command will be passed as a list.
- ❖ **options** – This is string of option letters that the Python program recognize as, for input or for output, with options (like 'i' or 'o') that followed by a colon (:). Here colon is used to denote the mode.
- ❖ **long\_options** – This parameter is passed with a list of strings. Argument of Long options should be followed by an equal sign ('='). In our program the C++ file name will be passed as string and 'i' also will be passed along with to indicate it as the input file.

#### **getopt() method returns value consisting of two elements**

- ❖ Each of these values are stored separately in two different list (arrays) **opts** and **args**. Opts contains list of splitted strings like mode, path and args contains any string if at all not splitted because of wrong path or mode.
- ❖ args will be an empty array if there is no error in splitting strings by getopt().

**opts, args = getopt.getopt (argv, "i:", ['ifile='])**

where opts contains	[['-i', 'c:\\pyprg\\p4']]
-i :-	option nothing but mode should be followed by :
'c:\\pyprg\\p4'	value nothing but the absolute path of C++ file.

- ❖ In our examples since the entire command line commands are

	<p>parsed and no leftover argument, the second argument args will be empty []. If args is displayed using print() command it displays the output as [].</p>			
b)	<b>Basis of Comparison</b>	<b>DBMS</b>	<b>RDBMS</b>	5
	Expansion	Database Management System.	Relational Database Management System	
	Data storage	Navigational model ie data by linked records	Relational model (in tables). ie data in tables as row and column	
	Data redundancy	Exhibit	Not Present	
	Normalization	Not performed	RDBMS uses normalization to reduce redundancy	
	Data access	Consumes more time	Faster, compared to DBMS.	
	Keys and indexes	Does not use.	used to establish relationship. Keys are used in RDBMS.	
	Transaction management	Inefficient, Error prone and insecure	Efficient and secure.	
	Distributed Databases	Not supported	Supported by RDBMS.	
	Example	Dbase, FoxPro.	SQL server, Oracle, mysql, MariaDB, SQLite	
38. a)	<p>❖ Histogram refers to a graphical representation; that displays data by way of bars to show the frequency of numerical data. A bar graph is a pictorial representation of data that uses bars to compare different categories of data.</p> <p>❖ A histogram represents the frequency distribution of continuous variables. Conversely, a bar graph is a diagrammatic comparison of discrete variables.</p> <p>❖ Histogram presents numerical data whereas bar graph shows categorical data.</p> <p>The histogram is drawn in such a way that there is no gap between the bars. On the other hand, there is proper spacing between bars in a bar graph that indicates discontinuity.</p> <p>❖ Items of the histogram are numbers, which are categorised together, to represent ranges of data. As opposed to the bar graph, items are considered as individual entities.</p> <p>❖ In the case of a bar graph, it is quite common to rearrange the blocks, from highest to lowest. But with histogram, this cannot be done, as they are shown in the sequence of classes.</p> <p>❖ The width of rectangular blocks in a histogram may or may not be same while the width of the bars in a bar graph is always same.</p>			5



b)	<ul style="list-style-type: none"><li>Continue statement unlike the break statement is used to skip the remaining part of a loop and start with next iteration.</li></ul> <p><b>Syntax:</b></p> <pre>continue</pre> <p><b>Example:</b></p> <pre>for word in "Jump Statement":     if word == "e":         continue     print (word, end="") print ("\n End of the program")</pre>	5
----	---	---

\*\*\*\*\*

**Prepared by:**

**Department of Computer science**

**SHRI VIDHYABHARATHI MATRIC HR.SEC.SCHOOL**

**SAKKARAMPALAYAM , AGARAM (PO) ELACHIPALAYAM**

**TIRUCHENGODE(TK), NAMAKKAL (DT) PIN-637202**

**Cell : 8675047607**

\*\*\*\*\*

ஆண்டுதோறும் மாநிலம் மற்றும் மாவட்ட அளவில் சிறப்பிடம்  
பெற்ற பள்ளி

**2022 – 2023** ஆம் கல்வி ஆண்டிற்கான

**NEET & +1 அட்மிசன்** நடைபெறுகிறது.