

FUNCTIONS

TIME INTELLIGENCE FUNCTIONS

MATH AND TRIG FUNCTIONS

FILTER FUNCTIONS

PARENT AND CHILD FUNCTIONS

DATE AND TIME FUNCTIONS

STATISTICAL FUNCTIONS

INFORMATION FUNCTIONS

TEXT FUNCTIONS

LOGICAL FUNCTIONS

OTHER FUNCTIONS

TIME INTELLIGENCE FUNCTIONS

Function	Description	Syntax	Link
TOTALYTD Function (DAX)	Evaluates the year-to-date value of the expression in the current context.	<code>TOTALYTD(<expression>,<dates>[,<filter>][,<year_end_date>])</code>	Edit
TOTALQTD Function (DAX)	Evaluates the value of the expression for the dates in the quarter to date, in the current context.	<code>TOTALQTD(<expression>,<dates>[,<filter>])</code>	Edit
TOTALMTD Function (DAX)	Evaluates the value of the expression for the month to date, in the current context.	<code>TOTALMTD(<expression>,<dates>[,<filter>])</code>	Edit
STARTOFTYEAR Function (DAX)	Returns the first date of the year in the current context for the specified column of dates.	<code>STARTOFTYEAR(<dates>)</code>	Edit
STARTOFQUARTER Function (DAX)	Returns the first date of the quarter in the current context for the specified column of dates.	<code>STARTOFQUARTER(<dates>)</code>	Edit
STARTOFMONTH Function (DAX)	Returns the first date of the month in the current context for the specified column of dates.	<code>STARTOFMONTH(<dates>)</code>	Edit
SAMEPERIODLASTYEAR Function (DAX)	Returns a table that contains a column of dates shifted one year back in time from the dates in the specified dates column, in the current context.	<code>SAMEPERIODLASTYEAR(<dates>)</code>	Edit
PREVIOUSYEAR Function (DAX)	Returns a table that contains a column of all dates from the previous year, given the last date in the dates column, in the current context.	<code>PREVIOUSYEAR(<dates>[,<year_end_date>])</code>	Edit
PREVIOUSQUARTER Function (DAX)	Returns a table that contains a column of all dates from the previous quarter, based on the first date in the dates column, in the current context.	<code>PREVIOUSQUARTER(<dates>)</code>	Edit
PREVIOUSMONTH Function (DAX)	Returns a table that contains a column of all dates from the previous month, based on the first date in the dates column, in the current context.	<code>PREVIOUSMONTH(<dates>)</code>	Edit
PREVIOUSDAY Function (DAX)	Returns a table that contains a column of all dates representing the day that is previous to the first date in the dates column, in the current context.	<code>PREVIOUSDAY(<dates>)</code>	Edit
PARALLELPERIOD Function (DAX)	Returns a table that contains a column of dates that represents a period parallel to the dates in the specified dates column, in the current context, with the dates shifted a number of intervals either forward in time or back in time.	<code>PARALLELPERIOD(<dates>,<number_of_intervals>,<interval>)</code>	Edit
OPENINGBALANCEYEAR Function (DAX)	Evaluates the expression at the first date of the year in the current context.	<code>OPENINGBALANCEYEAR(<expression>,<dates>[,<filter>][,<year_end_date>])</code>	Edit
OPENINGBALANCEQUARTER Function (DAX)	Evaluates the expression at the first date of the quarter, in the current context.	<code>OPENINGBALANCEQUARTER(<expression>,<dates>[,<filter>])</code>	Edit
OPENINGBALANCEMONTH Function (DAX)	Evaluates the expression at the first date of the month in the current context.	<code>OPENINGBALANCEMONTH(<expression>,<dates>[,<filter>])</code>	Edit
NEXTYEAR Function (DAX)	Returns a table that contains a column of all dates in the next year, based on the first date in the dates column, in the current context.	<code>NEXTYEAR(<dates>[,<year_end_date>])</code>	Edit
NEXTQUARTER Function (DAX)	Returns a table that contains a column of all dates in the next quarter, based on the first date specified in the dates column, in the current context.	<code>NEXTQUARTER(<dates>)</code>	Edit
NEXTMONTH Function (DAX)	Returns a table that contains a column of all dates from the next month, based on the first date in the dates column in the current context.	<code>NEXTMONTH(<dates>)</code>	Edit
NEXTDAY Function (DAX)	Returns a table that contains a column of all dates from the next day, based on the first date specified in the dates column in the current context.	<code>NEXTDAY(<dates>)</code>	Edit
LASTNONBLANK Function (DAX)	Returns the last value in the column, column, filtered by the current context, where the expression is not blank.	<code>LASTNONBLANK(<column>,<expression>)</code>	Edit
LASTDATE Function (DAX)	Returns the last date in the current context for the specified column of dates.	<code>LASTDATE(<dates>)</code>	Edit
FIRSTNONBLANK Function (DAX)	Returns the first value in the column, column, filtered by the current context, where the expression is not blank.	<code>FIRSTNONBLANK(<column>,<expression>)</code>	Edit
FIRSTDAY Function (DAX)	Returns the first date in the current context for the specified column of dates.	<code>FIRSTDAY(<dates>)</code>	Edit
ENDOFTYEAR Function (DAX)	Returns the last date of the year in the current context for the specified column of dates.	<code>ENDOFTYEAR(<dates>[,<year_end_date>])</code>	Edit
ENDOFQUARTER Function (DAX)	Returns the last date of the quarter in the current context for the specified column of dates.	<code>ENDOFQUARTER(<dates>)</code>	Edit
ENDOFTMONTH Function (DAX)	Returns the last date of the month in the current context for the specified column of dates.	<code>ENDOFTMONTH(<dates>)</code>	Edit
DATESYTD Function (DAX)	Returns a table that contains a column of the dates for the year to date, in the current context.	<code>DATESYTD(<dates>[,<year_end_date>])</code>	Edit
DATESQTD Function (DAX)	Returns a table that contains a column of the dates for the quarter to date, in the current context.	<code>DATESQTD(<dates>)</code>	Edit
DATESMTD Function (DAX)	Returns a table that contains a column of the dates for the month to date, in the current context.	<code>DATESMTD(<dates>)</code>	Edit
DATESINPERIOD Function (DAX)	Returns a table that contains a column of dates that begins with the start_date and continues for the specified number_of_intervals.	<code>DATESINPERIOD(<dates>,<start_date>,<number_of_intervals>,<interval>)</code>	Edit
DATESBETWEEN Function (DAX)	Returns a table that contains a column of dates that begins with the start_date and continues until the end_date.	<code>DATESBETWEEN(<dates>,<start_date>,<end_date>)</code>	Edit
DATEADD Function (DAX)	Returns a table that contains a column of dates, shifted either forward or backward in time by the specified number of intervals from the dates in the current context.	<code>DATEADD(<dates>,<number_of_intervals>,<interval>)</code>	Edit
CLOSINGBALANCEYEAR Function (DAX)	Evaluates the expression at the last date of the year in the current context.	<code>CLOSINGBALANCEYEAR(<expression>,<dates>[,<filter>][,<year_end_date>])</code>	Edit
CLOSINGBALANCEQUARTER Function (DAX)	Evaluates the expression at the last date of the quarter in the current context.	<code>CLOSINGBALANCEQUARTER(<expression>,<dates>[,<filter>])</code>	Edit
CLOSINGBALANCEMONTH Function (DAX)	Evaluates the expression at the last date of the month in the current context.	<code>CLOSINGBALANCEMONTH(<expression>,<dates>[,<filter>])</code>	Edit

FILTER FUNCTIONS

Function	Description	Syntax	Link
VALUES Function (DAX)	Returns a one-column table that contains the distinct values from the specified table or column. In other words, duplicate values are removed and only unique values are returned.	VALUES(<TableNameOrColumnName>)	🔗
USERELATIONSHIP Function (DAX)	Specifies the relationship to be used in a specific calculation as the one that exists between columnName1 and columnName2.	USERELATIONSHIP(<columnName1>, <columnName2>)	🔗
SUBSTITUTEWITHINDEX Function (DAX)	Returns a table which represents a left semijoin of the two tables supplied as arguments. The semi-join is performed by using common columns, determined by common column names and common data type . The columns being joined on are replaced with a single column in the returned table which is of type integer and contains an index. The index is a reference into the right join table given a specified sort order.	SUBSTITUTEWITHINDEX(<table>, <index-ColumnName>, <indexColumnsTable>, [<orderBy_expression>, [<order>][, <order-By_expression>, [<order>]]...])	🔗
SELECTEDVALUE Function (DAX)	Returns the value when the context for columnName has been filtered down to one distinct value only. Otherwise returns alternateResult.	SELECTEDVALUE(<columnName>[, <alternateResult>])	🔗
RELATEDTABLE Function (DAX)	Evaluates a table expression in a context modified by the given filters.	RELATEDTABLE(<tableName>)	🔗
RELATED Function (DAX)	Returns a related value from another table.	RELATED(<column>)	🔗
KEEPFILTERS Function (DAX)	Modifies how filters are applied while evaluating a CALCULATE or CALCULATETABLE function.	KEEPFILTERS(<expression>)	🔗
ISFILTERED Function (DAX)	Returns TRUE when columnName is being filtered directly. If there is no filter on the column or if the filtering happens because a different column in the same table or in a related table is being filtered then the function returns FALSE.	ISFILTERED(<columnName>)	🔗
ISCROSSFILTERED Function (DAX)	Returns TRUE when columnName or another column in the same or related table is being filtered.	ISCROSSFILTERED(<columnName>)	🔗
HASONEVALUE Function (DAX)	Returns TRUE when the context for columnName has been filtered down to one distinct value only. Otherwise is FALSE.	HASONEVALUE(<columnName>)	🔗
HASONEFILTER Function (DAX)	Returns TRUE when the number of directly filtered values on columnName is one.		
FILTERS Function (DAX)	Returns the values that are directly applied as filters to columnName.	FILTERS(<columnName>)	🔗
FILTER Function (DAX)	Returns a table that represents a subset of another table or expression.	FILTER(<table>,<filter>)	🔗
EARLIER Function (DAX)	Returns the current value of the specified column in an outer evaluation pass of the specified column.	EARLIER(<column>)	🔗
EARLIER Function (DAX)	Returns the current value of the specified column in an outer evaluation pass of the mentioned column.	EARLIER(<column>, <number>)	🔗
DISTINCT Function (DAX)	Returns a one-column table that contains the distinct values from the specified column. In other words, duplicate values are removed and only unique values are returned.	DISTINCT(<column>)	🔗
CROSSFILTER Function	Specifies the cross-filtering direction to be used in a calculation for a relationship that exists between two columns.	CROSSFILTER(<columnName1>, <columnName2>, <direction>)	🔗
CALCULATETABLE Function (DAX)	Evaluates a table expression in a context modified by the given filters.	CALCULATETABLE(<expression>, <filter1>,<filter2>,...)	🔗
CALCULATE Function (DAX)	Evaluates an expression in a context that is modified by the specified filters.	CALCULATE(<expression>,<filter1>,<filter2>...)	🔗
ALLSELECTED Function (DAX)	Removes context filters from columns and rows in the current query, while retaining all other context filters or explicit filters.	ALLSELECTED([<tableName> <columnName>])	🔗
ALLNOBLANKROW Function (DAX)	From the parent table of a relationship, returns all rows but the blank row, or all distinct values of a column but the blank row, and disregards any context filters that might exist.	ALLNOBLANKROW({<table> <column>[,<column>[,...]]})	🔗
ALL Function (DAX)	Returns all the rows in a table, or all the values in a column, ignoring any filters that might have been applied. This function is useful for clearing filters and creating calculations on all the rows in a table.	ALL({<table> <column>[,<column>[,...]]})	🔗
ALLEXCEPT Function (DAX)	Removes all context filters in the table except filters that have been applied to the specified columns.	ALLEXCEPT(<table>,<column>[,<column>[,...]])	🔗
ADDMISSINGITEMS Function (DAX)	Adds combinations of items from multiple columns to a table if they do not already exist. The determination of which item combinations to add is based on referencing source columns which contain all the possible values for the columns.	ADDMISSINGITEMS(<showAllColumn>[,<show-AllColumn>...], <table>, <groupingColumn>[,<groupingColumn>...][, filterTable]...)	🔗

DATE AND TIME FUNCTIONS

Function	Description	Syntax	Link
CALENDARAUTO Function (DAX)	Returns a table with a single column named "Date" that contains a contiguous set of dates. The range of dates is calculated automatically based on data in the model.	CALENDARAUTO([fiscal_year_end_month])	↗
CALENDAR Function (DAX)	Returns a table with a single column named "Date" that contains a contiguous set of dates. The range of dates is from the specified start date to the specified end date, inclusive of those two dates.	CALENDAR(<start_date>, <end_date>)	↗
DATEDIFF Function (DAX)	Returns the count of interval boundaries crossed between two dates.	DATEDIFF(<start_date>, <end_date>, <interval>)	↗
DATE Function (DAX)	Returns the specified date in datetime format.	DATE(<year>, <month>, <day>)	↗
DATEVALUE Function (DAX)	Converts a date in the form of text to a date in datetime format.	DATEVALUE(date_text)	↗
DAY Function (DAX)	Returns the day of the month, a number from 1 to 31.	DAY(<date>)	↗
EDATE Function (DAX)	Returns the date that is the indicated number of months before or after the start date. Use EDATE to calculate maturity dates or due dates that fall on the same day of the month as the date of issue.	EDATE(<start_date>, <months>)	↗
EOMONTH Function (DAX)	Returns the date in datetime format of the last day of the month, before or after a specified number of months. Use EOMONTH to calculate maturity dates or due dates that fall on the last day of the month.	EOMONTH(<start_date>, <months>)	↗
HOUR Function (DAX)	Returns the hour as a number from 0 (12:00 A.M.) to 23 (11:00 P.M.).	HOUR(<datetime>)	↗
MINUTE Function (DAX)	Returns the minute as a number from 0 to 59, given a date and time value.	MINUTE(<datetime>)	↗
MONTH Function (DAX)	Returns the month as a number from 1 (January) to 12 (December).	MONTH(<datetime>)	↗
NOW Function (DAX)	Returns the current date and time in datetime format.	NOW()	↗
SECOND Function (DAX)	Returns the seconds of a time value, as a number from 0 to 59.	SECOND(<time>)	↗
TIME Function (DAX)	Converts hours, minutes, and seconds given as numbers to a time in datetime format.	TIME(hour, minute, second)	↗
TIMEVALUE Function (DAX)	Converts a time in text format to a time in datetime format.	TIMEVALUE(time_text)	↗
TODAY Function (DAX)	Returns the current date.	TODAY()	↗
WEEKDAY Function (DAX)	Returns a number from 1 to 7 identifying the day of the week of a date. By default the day ranges from 1 (Sunday) to 7 (Saturday).	WEEKDAY(<date>, <return_type>)	↗
WEEKNUM Function (DAX)	Returns the week number for the given date and year according to the return_type value. The week number indicates where the week falls numerically within a year.	WEEKNUM(<date>, <return_type>)	↗
YEARFRAC Function (DAX)	Calculates the fraction of the year represented by the number of whole days between two dates. Use the YEARFRAC worksheet function to identify the proportion of a whole year's benefits or obligations to assign to a specific term.	YEARFRAC(<start_date>, <end_date>, <basis>)	↗
YEAR Function (DAX)	Returns the year of a date as a four digit integer in the range 1900-9999.	YEAR(<date>)	↗

INFORMATION FUNCTIONS

Function	Description	Syntax	Link
USERNAME Function (DAX)	Returns the domain name and username from the credentials given to the system at connection time	USERNAME()	
LOOKUPVALUE Function (DAX)	Returns the value in result_columnName for the row that meets all criteria specified by search_columnName and search_value.	LOOKUPVALUE(<result_columnName>, <search_columnName>, <search_value>)	
ISTEXT Function (DAX)	Checks if a value is text, and returns TRUE or FALSE.	ISTEXT(<value>)	
ISONORAFTER Function (DAX)	A boolean function that emulates the behavior of a 'Start At' clause and returns true for a row that meets all of the condition parameters.	ISONORAFTER(<scalar_expression>, <scalar_expression>[, sort_order [, <scalar_expression>, <scalar_expression>[, sort_order]]...])	
ISNUMBER Function (DAX)	Checks whether a value is a number, and returns TRUE or FALSE.	ISNUMBER(<value>)	
ISNOTTEXT Function (DAX)	Checks if a value is not text (blank cells are not text), and returns TRUE or FALSE.	ISNOTTEXT(<value>)	
ISLOGICAL Function (DAX)	Checks whether a value is a logical value, (TRUE or FALSE), and returns TRUE or FALSE.	ISLOGICAL(<value>)	
ISEVEN Function (DAX)	Returns TRUE if number is even, or FALSE if number is odd.	ISEVEN(number)	
ISERROR Function (DAX)	Checks whether a value is an error, and returns TRUE or FALSE.	ISERROR(<value>)	
ISBLANK Function (DAX)	Checks whether a value is blank, and returns TRUE or FALSE.	ISBLANK(<value>)	
CUSTOMDATA Function (DAX)	Returns the content of the CustomData property in the connection string.	CUSTOMDATA()	
CONTAINS Function (DAX)	Returns true if values for all referred columns exist, or are contained, in those columns.	CONTAINS(<table>, <columnName>, <value> [, <columnName>, <value>]...)	

LOGICAL FUNCTIONS

Function	Description	Syntax	Link
TRUE Function (DAX)	Returns the logical value TRUE.	TRUE()	
SWITCH Function (DAX)	Evaluates an expression against a list of values and returns one of multiple possible result expressions.	SWITCH(<expression>, <value>, <result>[, <value>, <result>]...[, <else>])	
OR Function (DAX)	Checks whether one of the arguments is TRUE to return TRUE. The function returns FALSE if both arguments are FALSE.	OR(<logical1>,<logical2>)	
NOT Function (DAX)	Changes FALSE to TRUE, or TRUE to FALSE.	NOT(<logical>)	
IF Function (DAX)	Checks if a condition provided as the first argument is met. Returns one value if the condition is TRUE, and returns another value if the condition is FALSE.	IF(logical_test>,<value_if_true>, value_if_false)	
IFERROR Function (DAX)	Evaluates an expression and returns a specified value if the expression returns an error		
FALSE Function (DAX)	Returns the logical value FALSE.	FALSE()	
AND Function (DAX)	Checks whether both arguments are TRUE, and returns TRUE if both arguments are TRUE. Otherwise returns false.	AND(<logical1>,<logical2>)	

MATH AND TRIG FUNCTIONS

Function	Description	Syntax	Link
TRUNC Function (DAX)	Truncates a number to an integer by removing the decimal, or fractional, part of the number.	TRUNC(<number>, <num_digits>)	🔗
SUMX Function (DAX)	Returns the sum of an expression evaluated for each row in a table.	SUMX(<table>, <expression>)	🔗
SUM Function (DAX)	Adds all the numbers in a column.	SUM(<column>)	🔗
SQRT Function (DAX)	Returns the square root of a number.	SQRT(<number>)	🔗
SIGN Function (DAX)	Determines the sign of a number, the result of a calculation, or a value in a column. The function returns 1 if the number is positive, 0 (zero) if the number is zero, or -1 if the number is negative.	SIGN(<number>)t	🔗
ROUNDUP Function (DAX)	Rounds a number up, away from 0 (zero).	ROUNDUP(<number>, <num_digits>)	🔗
ROUND Function (DAX)	Rounds a number to the specified number of digits.	ROUND(<number>, <num_digits>)	🔗
ROUNDDOWN Function (DAX)	Rounds a number down, toward zero.	ROUNDDOWN(<number>, <num_digits>)	🔗
RAND Function (DAX)	Returns a random number greater than or equal to 0 and less than 1, evenly distributed. The number that is returned changes each time the cell containing this function is recalculated.	RAND()	🔗
RANDBETWEEN Function (DAX)	Returns a random number in the range between two numbers you specify.	RANDBETWEEN(<bottom>, <top>)	🔗
RADIANS Function (DAX)	Converts degrees to radians.	RADIANS(angle)	🔗
QUOTIENT Function (DAX)	Performs division and returns only the integer portion of the division result. Use this function when you want to discard the remainder of division.	QUOTIENT(<numerator>, <denominator>)	🔗
PRODUCTX Function (DAX)	Returns the product of an expression evaluated for each row in a table.	PRODUCTX(<table>, <expression>)	🔗
PRODUCT Function (DAX)	Returns the product of the numbers in a column.	PRODUCT(<column>)	🔗
POWER Function (DAX)	Returns the result of a number raised to a power.	POWER(<number>, <power>)	🔗
PI Function (DAX)	Returns the value of Pi, 3.14159265358979, accurate to 15 digits.	PI()	🔗
ODD Function (DAX)	Returns number rounded up to the nearest odd integer.	ODD(number)	🔗
MROUND Function (DAX)	Returns a number rounded to the desired multiple.	MROUND(<number>, <multiple>)	🔗
LOG Function (DAX)	Returns the logarithm of a number to the base you specify.	LOG(<number>, <base>)	🔗
LOG10 Function (DAX)	Returns the base-10 logarithm of a number.	LOG10(<number>)	🔗
LN Function (DAX)	Returns the natural logarithm of a number. Natural logarithms are based on the constant e (2.71828182845904).	LN(<number>)	🔗
LCM Function (DAX)	Returns the least common multiple of integers. The least common multiple is the smallest positive integer that is a multiple of all integer arguments number1, number2, and so on. Use LCM to add fractions with different denominators.	LCM(number1, [number2], ...)	🔗
ISO.CEILING Function (DAX)	Rounds a number up, to the nearest integer or to the nearest multiple of significance.	ISO.CEILING(<number>[, <significance>])	🔗
INT Function (DAX)	Rounds a number down to the nearest integer.	INT(<number>)	🔗
GCD Function (DAX)	Returns the greatest common divisor of two or more integers. The greatest common divisor is the largest integer that divides both number1 and number2 without a remainder.	GCD(number1, [number2], ...)	🔗
FLOOR Function (DAX)	Rounds a number down, toward zero, to the nearest multiple of significance.	FLOOR(<number>, <significance>)	🔗
FACT Function (DAX)	Returns the factorial of a number, equal to the series $1*2*3*...*$, ending in the given number.	FACT(<number>)	🔗
EXP Function (DAX)	Returns e raised to the power of a given number. The constant e equals 2.71828182845904, the base of the natural logarithm.	EXP(<number>)	🔗
EVEN Function (DAX)	Returns number rounded up to the nearest even integer. You can use this function for processing items that come in twos. For example, a packing crate accepts rows of one or two items. The crate is full when the number of items, rounded up to the nearest two, matches the crate's capacity.	EVEN(number)	🔗
DIVIDE Function (DAX)	Performs division and returns alternate result or BLANK() on division by 0.	DIVIDE(<numerator>, <denominator> [, <alternateresult>])	🔗
DEGREES Function (DAX)	Converts radians into degrees.	DEGREES(angle)	🔗
CURRENCY Function (DAX)	Evaluates the argument and returns the result as currency data type.	CURRENCY(<value>)	🔗
COSH Function (DAX)	Returns the hyperbolic cosine of a number.	COSH(number)	🔗
COS Function (DAX)	Returns the cosine of the given angle.	COS(number)	🔗
COMBIN Function (DAX)	Returns the number of combinations for a given number of items. Use COMBIN to determine the total possible number of groups for a given number of items.	COMBIN(number, number_chosen)	🔗
COMBINA Function (DAX)	Returns the number of combinations (with repetitions) for a given number of items.	COMBINA(number, number_chosen)	🔗
CEILING Function (DAX)	Rounds a number up, to the nearest integer or to the nearest multiple of significance.	CEILING(<number>, <significance>)	🔗
ATANH Function (DAX)	Returns the inverse hyperbolic tangent of a number. Number must be between -1 and 1 (excluding -1 and 1). The inverse hyperbolic tangent is the value whose hyperbolic tangent is number, so ATANH(TANH(number)) equals number.	ATANH(number)	🔗
ATAN Function (DAX)	Returns the arctangent, or inverse tangent, of a number. The arctangent is the angle whose tangent is number. The returned angle is given in radians in the range -pi/2 to pi/2.	ATAN(number)	🔗
ASINH Function (DAX)	Returns the inverse hyperbolic sine of a number. The inverse hyperbolic sine is the value whose hyperbolic sine is number, so ASINH(SINH(number)) equals number.	ASINH(number)	🔗
ASIN Function (DAX)	Returns the arcsine, or inverse sine, of a number. The arcsine is the angle whose sine is number. The returned angle is given in radians in the range -pi/2 to pi/2.	ASIN(number)	🔗
ACOSH Function (DAX)	Returns the inverse hyperbolic cosine of a number. The number must be greater than or equal to 1. The inverse hyperbolic cosine is the value whose hyperbolic cosine is number, so ACOSH(COSH(-number)) equals number.	ACOSH(number)	🔗
ACOS Function (DAX)	Returns the arccosine, or inverse cosine, of a number. The arccosine is the angle whose cosine is number. The returned angle is given in radians in the range 0 (zero) to pi.	ACOS(number)	🔗
ABS Function (DAX)	Returns the absolute value of a number.	ABS(<number>)	🔗

PARENT AND CHILD FUNCTIONS

Function	Description	Syntax	Link
PATHLENGTH Function (DAX)	Returns the number of parents to the specified item in a given PATH result, including self.	PATHLENGTH(<path>)	
PATHITEMREVERSE Function (DAX)	Returns the item at the specified position from a string resulting from evaluation of a PATH function. Positions are counted backwards from right to left.	PATHITEMREVERSE(<path>, <position> [, <type>])	
PATHITEM Function (DAX)	Returns the item at the specified position from a string resulting from evaluation of a PATH function. Positions are counted from left to right.	PATHITEM(<path>, <position> [, <type>])	
PATH Function (DAX)	Returns a delimited text string with the identifiers of all the parents of the current identifier, starting with the oldest and continuing until current.	PATH(<ID_columnName>, <parent_columnName>)	
PATHCONTAINS Function (DAX)	Returns TRUE if the specified item exists within the specified path.	PATHCONTAINS(<path>, <item>)	

STATISTICAL FUNCTIONS

Function	Description	Syntax	Link
XNPV Function (DAX)	Returns the present value for a schedule of cash flows that is not necessarily periodic.	XNPV(<table>, <values>, <dates>, <rate>)	🔗
XIRR Function (DAX)	Returns the internal rate of return for a schedule of cash flows that is not necessarily periodic.	XIRR(<table>, <values>, <dates>, [guess])	🔗
VARX.S Function (DAX)	and n is the population size	VARX.S(InternetSales_USD, InternetSales_USD[UnitPrice_USD] - (InternetSales_USD[DiscountAmount_USD]/InternetSales_USD[OrderQuantity]))	🔗
VARX.S Function (DAX)	Returns the variance of a sample population.	VARX.S(<table>, <expression>)	🔗
VARX.P Function (DAX)	and n is the population size	VARX.P(InternetSales_USD, InternetSales_USD[UnitPrice_USD] - (InternetSales_USD[DiscountAmount_USD]/InternetSales_USD[OrderQuantity]))	🔗
VARX.P Function (DAX)	Returns the variance of the entire population.	VARX.P(<table>, <expression>)	🔗
VAR.S Function (DAX)	and n is the population size	VAR.S(InternetSales_USD[SalesAmount_USD])	🔗
VAR.S Function (DAX)	Returns the variance of a sample population.	VAR.S(<columnName>)	🔗
VAR.P Function (DAX)	and n is the population size	VAR.P(InternetSales_USD[SalesAmount_USD])	🔗
VAR.P Function (DAX)	Returns the variance of the entire population.	VAR.P(<columnName>)	🔗
TOPN Function (DAX)	Returns the top N rows of the specified table.	TOPN(<n_value>, <table>, <orderBy_expression>, [<order>[, <orderBy_expression>, [<order>]]-])	🔗
T.INV Function (DAX)	Returns the left-tailed inverse of the Student's t-distribution.	T.INV(Probability,Deg_freedom)	🔗
T.INV.2T Function (DAX)	Returns the two-tailed inverse of the Student's t-distribution.	T.INV.2T(Probability,Deg_freedom)	🔗
T.DIST.RT Function (DAX)	Returns the right-tailed Student's t-distribution.	T.DIST.RT(X,Deg_freedom)	🔗
T.DIST Function (DAX)	Returns the Student's left-tailed t-distribution.	T.DIST(X,Deg_freedom,Cumulative)	🔗
T.DIST.2T Function (DAX)	Returns the two-tailed Student's t-distribution.	T.DIST.2T(X,Deg_freedom)	🔗
TANH Function (DAX)	Returns the hyperbolic tangent of a number.	TANH(number)	🔗
TAN Function (DAX)	Returns the tangent of the given angle.	TAN(number)	🔗
SUMMARIZE Function (DAX)	Returns a summary table for the requested totals over a set of groups.	SUMMARIZE(<table>, <group-By_columnName>[, <groupBy_columnName>]...[, <name>, <expression>]...)	🔗
STDEVX.S Function (DAX)	and n is the population size	STDEVX.S(RELATEDTABLE(InternetSales_USD, InternetSales_USD[UnitPrice_USD] - (InternetSales_USD[DiscountAmount_USD]/InternetSales_USD[OrderQuantity])))	🔗
STDEVX.S Function (DAX)	Returns the standard deviation of a sample population.	STDEVX.S(<table>, <expression>)	🔗
STDEVX.P Function (DAX)	and n is the population size	STDEVX.P(RELATEDTABLE(InternetSales_USD, InternetSales_USD[UnitPrice_USD] - (InternetSales_USD[DiscountAmount_USD]/InternetSales_USD[OrderQuantity])))	🔗
STDEVX.P Function (DAX)	Returns the standard deviation of the entire population.	STDEVX.P(<table>, <expression>)	🔗
STDEV.S Function (DAX)	and n is the population size	STDEV.S(InternetSales_USD[SalesAmount_USD])	🔗
STDEV.S Function (DAX)	Returns the standard deviation of a sample population.	STDEV.S(<ColumnName>)	🔗
STDEV.P Function (DAX)	and n is the population size	STDEV.P(InternetSales_USD[SalesAmount_USD])	🔗
STDEV.P Function (DAX)	Returns the standard deviation of the entire population.	STDEV.P(<ColumnName>)	🔗
SQRTPI Function (DAX)	Returns the square root of (number * pi).	SQRTPI(number)	🔗
SINH Function (DAX)	Returns the hyperbolic sine of a number.	SINH(number)	🔗
SIN Function (DAX)	Returns the sine of the given angle.	SIN(number)	🔗
SELECTCOLUMNS Function (DAX)	Adds calculated columns to the given table or table expression.	SELECTCOLUMNS(<table>, <name>, <scalar_expression>[, <name>, <scalar_expression>]...)	🔗
SAMPLE Function (DAX)	Returns a sample of N rows from the specified table.	SAMPLE(<n_value>, <table>, <orderBy_expression>, [<order>[, <orderBy_expression>, [<order>]]-])	🔗
ROW Function (DAX)	Returns a table with a single row containing values that result from the expressions given to each column.	ROW(<name>, <expression>[], <name>, <expression>]...)	🔗
RANKX Function (DAX)	Returns the ranking of a number in a list of numbers for each row in the table argument.	RANKX(<table>, <expression>[, <value>[, <order>[, <ties>]]])	🔗
RANK.EQ Function (DAX)	Returns the ranking of a number in a list of numbers.	RANK.EQ(<value>, <columnName>[, <order>])	🔗
POISSON.DIST Function (DAX)	Returns the Poisson distribution. A common application of the Poisson distribution is predicting the number of events over a specific time, such as the number of cars arriving at a toll plaza in 1 minute.	POISSON.DIST(x,mean,cumulative)	🔗
PERCENTILEX.INC Function (DAX)	Returns the percentile number of an expression evaluated for each row in a table.	PERCENTILEX.INC(<table>, <expression>)	
PERCENTILEX.EXC Function (DAX)	Returns the percentile number of an expression evaluated for each row in a table.	PERCENTILEX.EXC(<table>, <expression>, k)	🔗
PERCENTILE.INC Function (DAX)	Returns the k-th percentile of values in a range, where k is in the range 0..1, inclusive.	PERCENTILE.INC(<column>, <k>)	🔗
PERCENTILE.EXC Function (DAX)	Returns the k-th percentile of values in a range, where k is in the range 0..1, exclusive.	PERCENTILE.EXC(<column>, <k>)	🔗
NORM.S.INV (DAX)	Returns the inverse of the standard normal cumulative distribution. The distribution has a mean of zero and a standard deviation of one.	NORM.S.INV(Probability)	🔗
NORM.S.DIST Function (DAX)	Returns the standard normal distribution (has a mean of zero and a standard deviation of one).	NORM.S.DIST(Z, Cumulative)	🔗
NORM.INV Function (DAX)	The inverse of the normal cumulative distribution for the specified mean and standard deviation.	NORM.INV(Probability, Mean, Standard_dev)	🔗
NORM.DIST Function (DAX)	Returns the normal distribution for the specified mean and standard deviation.	NORM.DIST(X, Mean, Standard_dev, Cumulative)	🔗
MINX Function (DAX)	Returns the smallest numeric value that results from evaluating an expression for each row of a table.	MINX(<table>, <expression>)	🔗
MIN Function (DAX)	Returns the smallest numeric value in a column, or between two scalar expressions. Ignores logical values and text.	MIN(<column>)	🔗
MINA Function (DAX)	Returns the smallest value in a column, including any logical values and numbers represented as text.	MINA(<column>)	🔗
MEDIANX Function (DAX)	Returns the median number of an expression evaluated for each row in a table.	MEDIANX(<table>, <expression>)	🔗
MEDIAN Function (DAX)	Returns the median of numbers in a column.	MEDIAN(<column>)	🔗
MAXX Function (DAX)	Evaluates an expression for each row of a table and returns the largest numeric value.	MAXX(<table>,<expression>)	🔗
MAX Function (DAX)	Returns the largest numeric value in a column, or between two scalar expressions.	MAX(<column>)	🔗
MAXA Function (DAX)	Returns the largest value in a column. Logical values and blanks are counted.	MAXA(<column>)	🔗
GEOMEANX Function (DAX)	Returns the geometric mean of an expression evaluated for each row in a table.	GEOMEANX(<table>, <expression>)	🔗
GEOMEAN Function (DAX)	Returns the geometric mean of the numbers in a column.	GEOMEAN(<column>)	🔗
GENERATE Function (DAX)	Returns a table with the Cartesian product between each row in table1 and the table that results from evaluating table2 in the context of the current row from table1.	GENERATE(<table1>, <table2>)	🔗
GENERATEALL Function (DAX)	Returns a table with the Cartesian product between each row in table1 and the table that results from evaluating table2 in the context of the current row from table1.	GENERATEALL(<table1>, <table2>)	🔗
EXPON.DIST Function (DAX)	Returns the exponential distribution. Use EXPON.DIST to model the time between events, such as how long an automated bank teller takes to deliver cash. For example, you can use EXPON.DIST to determine the probability that the process takes at most 1 minute.	EXPON.DIST(x,lambda,cumulative)	🔗
DISTINCTCOUNT Function (DAX)	The DISTINCTCOUNT function counts the number of distinct values in a column.	DISTINCTCOUNT(<column>)	🔗
DATABASE Function	Provides a mechanism for declaring an inline set of data values.	DATABASE (ColumnName1, DataType1, ColumnName2, DataType2..., {Value1, Value2..., {ValueN, ValueN+1...}}...)	🔗
CROSSJOIN Function (DAX)	Returns a table that contains the Cartesian product of all rows from all tables in the arguments. The columns in the new table are all the columns in all the argument tables.	CROSSJOIN(<table>, <table>[], <table>]..)	🔗
COUNTX Function (DAX)	Counts the number of rows that contain a number or an expression that evaluates to a number, when evaluating an expression over a table.	COUNTX(<table>,<expression>)	🔗
COUNTROWS Function (DAX)	The COUNTROWS function counts the number of rows in the specified table, or in a table defined by an expression.	COUNTROWS(<table>)	🔗
COUNT Function (DAX)	The COUNT function counts the number of cells in a column that contain numbers.	COUNT(<column>)	🔗
COUNTBLANK Function (DAX)	Counts the number of blank cells in a column.	COUNTBLANK(<column>)	🔗
COUNTA Function (DAX)	The COUNTA function counts nonblank results when evaluating the result of an expression over a table. That is, it works just like the COUNTA function, but is used to iterate through the rows in a table and count rows where the specified expressions results in a nonblank result.	COUNTA(<table>,<expression>)	🔗
COUNTA Function (DAX)	The COUNTA function counts the number of cells in a column that are not empty. It counts not just rows that contain numeric values, but also rows that contain nonblank values, including text, dates, and logical values.	COUNTA(<column>)	🔗
CONFIDENCE.T Function (DAX)	Returns the confidence interval for a population mean, using a Student's t distribution.	CONFIDENCE.T(alpha,standard_dev,size)	🔗
CONFIDENCE.NORM Function (DAX)	The confidence interval is a range of values. Your sample mean, x, is at the center of this range and the range is $x \pm \text{CONFIDENCE.NORM}$. For example, if x is the sample mean of delivery times for products ordered through the mail, $x \pm \text{CONFIDENCE.NORM}$ is a range of population means. For any population mean, μ_0 , in this range, the probability of obtaining a sample mean further from μ_0 than x is greater than alpha		
CHISQ.INV.RT Function (DAX)	Returns the inverse of the right-tailed probability of the chi-squared distribution.	CHISQ.INV.RT(probability,deg_freedom)	🔗
CHISQ.INV Function (DAX)	Returns the inverse of the left-tailed probability of the chi-squared distribution.	CHISQ.INV(probability,deg_freedom)	🔗
BETA.INV Function (DAX)	Returns the inverse of the beta cumulative probability density function (BETA.DIST).	BETA.INV(probability,alpha,beta,[A],[B])	🔗
BETA.DIST Function (DAX)	Returns the beta distribution. The beta distribution is commonly used to study variation in the percentage of something across samples, such as the fraction of the day people spend watching television.	BETA.DIST(x,alpha,beta,cumulative,[A],[B])	🔗
AVERAGEG Function (DAX)	Calculates the average (arithmetic mean) of a set of expressions evaluated over a table.	AVERAGEG(<table>,<expression>)	🔗
AVERAGE Function (DAX)	Returns the average (arithmetic mean) of all the numbers in a column.	AVERAGE(<column>)	🔗
AVERAGEA Function (DAX)	Returns the average (arithmetic mean) of the values in a column. Handles text and non-numeric values.	AVERAGEA(<column>)	🔗
ADDCOLUMNS Function (DAX)	Adds calculated columns to the given table or table expression.	ADDCOLUMNS(<table>, <name>, <expression>[, <name>, <expression>]...)	🔗

TEXT FUNCTIONS

Function	Description	Syntax	Link
VALUE Function (DAX)	Converts a text string that represents a number to a number.	VALUE(<text>)	🔗
UPPER Function (DAX)	Converts a text string to all uppercase letters.	UPPER (<text>)	🔗
UNICHAR Function (DAX)	Returns the Unicode character referenced by the numeric value.	UNICHAR(number)	🔗
TRIM Function (DAX)	Removes all spaces from text except for single spaces between words.	TRIM(<text>)	🔗
SUBSTITUTE Function (DAX)	Replaces existing text with new text in a text string.	SUBSTITUTE(<text>, <old_text>, <new_text>, <instance_num>)	🔗
SEARCH Function (DAX)	Returns the number of the character at which a specific character or text string is first found, reading left to right. Search is case-insensitive and accent sensitive.	SEARCH(<find_text>, <within_text>[, <start_num>][, <NotFoundValue>])	🔗
RIGHT Function (DAX)	RIGHT returns the last character or characters in a text string, based on the number of characters you specify.	RIGHT(<text>, <num_chars>)	🔗
REPT Function (DAX)	Repeats text a given number of times. Use REPT to fill a cell with a number of instances of a text string.	REPT(<text>, <num_times>)	🔗
REPLACE Function (DAX)	REPLACE replaces part of a text string, based on the number of characters you specify, with a different text string.	REPLACE(<old_text>, <start_num>, <num_chars>, <new_text>)	🔗
MID Function (DAX)	Returns a string of characters from the middle of a text string, given a starting position and length.	MID(<text>, <start_num>, <num_chars>)	🔗
LOWER Function (DAX)	Converts all letters in a text string to lowercase.	LOWER(<text>)	🔗
LEN Function (DAX)	Returns the number of characters in a text string.	LEN(<text>)	🔗
LEFT Function (DAX)	Returns the specified number of characters from the start of a text string.	LEFT(<text>, <num_chars>)	🔗
FORMAT Function (DAX)	Converts a value to text according to the specified format.	FORMAT(<value>, <format_string>)	🔗
FIXED Function (DAX)	Rounds a number to the specified number of decimals and returns the result as text. You can specify that the result be returned with or without commas.	FIXED(<number>, <decimals>, <no_commas>)	🔗
FIND Function (DAX)	Returns the starting position of one text string within another text string. FIND is case-sensitive.	FIND(<find_text>, <within_text>[, <start_num>][, <NotFoundValue>])	🔗
EXACT Function (DAX)	Compares two text strings and returns TRUE if they are exactly the same, FALSE otherwise. EXACT is case-sensitive but ignores formatting differences. You can use EXACT to test text being entered into a document.	EXACT(<text1>,<text2>)	🔗
CONCATENATEX Function (DAX)	Concatenates the result of an expression evaluated for each row in a table.	CONCATENATEX(<table>, <expression>, [delimiter])	🔗
CONCATENATE Function (DAX)	Joins two text strings into one text string.	CONCATENATE(<text1>, <text2>)	🔗
CODE Function (DAX)	Returns a numeric code for the first character in a text string. The returned code corresponds to the character set used by your computer.	CODE(text)	🔗
BLANK Function (DAX)	Returns a blank.	BLANK()	🔗

OTHER FUNCTIONS

Function	Description	Syntax	Link
UNION Function (DAX)	Creates a union (join) table from a pair of tables.	UNION(<table_expression1>, <table_expression2> [,<table_expression>]...)	
TREATAS Function (DAX)	Applies the result of a table expression as filters to columns from an unrelated table.	TREATAS(table_expression, <column>[, <column>[, <column>[,...]]])	
SUMMARIZECOLUMNS Function (DAX)	Returns a summary table over a set of groups.	SUMMARIZECOLUMNS(<groupBy_columnName> [, <groupBy_columnName>]..., [<filterTable>]...[, <name>, <expression>]...)	
NATURALLEFTOUTERJOIN Function (DAX)	Performs an inner join of a table with another table. The tables are joined on common columns (by name) in the two tables. If the two tables have no common column names, an error is returned.	NATURALLEFTOUTERJOIN(<leftJoinTable>, <rightJoinTable>)	
NATURALINNERJOIN Function (DAX)	Performs an inner join of a table with another table. The tables are joined on common columns (by name) in the two tables. If the two tables have no common column names, an error is returned.	NATURALINNERJOIN(<leftJoinTable>, <rightJoinTable>)	
ISEMPTY Function (DAX)	Checks if a table is empty.	ISEMPTY(<table_expression>)	
INTERSECT Function (DAX)	Returns the row intersection of two tables, retaining duplicates.	INTERSECT(<table_expression1>, <table_expression2>)	
GROUPBY Function (DAX)	The GROUPBY function is similar to the SUMMARIZE function. However, GROUPBY does not do an implicit CALCULATE for any extension columns that it adds. GROUPBY permits a new function, CURRENTGROUP(), to be used inside aggregation functions in the extension columns that it adds. GROUPBY attempts to reuse the data that has been grouped		
GENERATESERIES Function (DAX)	Returns a single column table containing the values of an arithmetic series, that is, a sequence of values in which each differs from the preceding by a constant quantity. The name of the column returned is Value.	GENERATESERIES(<startValue>, <endValue>[, <incrementValue>])	
EXCEPT Function (DAX)	Returns the rows of one table which do not appear in another table.	EXCEPT(<table_expression1>, <table_expression2>)	
ERROR Function (DAX)	Raises an error with an error message.	ERROR(<text>)	
DATATABLE Function (DAX)	Provides a mechanism for declaring an inline set of data values.	DATATABLE (ColumnName1, DataType1, ColumnName2, DataType2..., {{Value1, Value2...}, {ValueN, ValueN+1...}...})	

In [1]:

```
1 t = (1,2,3,4,5)
```

In [2]:

```
1 type(t)
```

Out[2]:

tuple

In [3]:

```
1 t1 = ("sudh", 345, 34.08, 45+8j, True)
```

In [4]:

```
1 t1
```

Out[4]:

('sudh', 345, 34.08, (45+8j), True)

In [5]:

```
1 l1 = ["sudh", 345, 34.08, 45+8j, True]
```

In [6]:

```
1 l1
```

Out[6]:

['sudh', 345, 34.08, (45+8j), True]

In [7]:

```
1 type(l1)
```

Out[7]:

list

In [8]:

```
1 t2 = ()
```

In [9]:

```
1 type(t2)
```

Out[9]:

tuple

In [11]:

```
1 l1
```

Out[11]:

```
['sudh', 345, 34.08, (45+8j), True]
```

In [13]:

```
1 l1[0:2]
```

Out[13]:

```
['sudh', 345]
```

In [14]:

```
1 t1[0:2]
```

Out[14]:

```
('sudh', 345)
```

In [15]:

```
1 t1
```

Out[15]:

```
('sudh', 345, 34.08, (45+8j), True)
```

In [16]:

```
1 t1[::-1]
```

Out[16]:

```
(True, (45+8j), 34.08, 345, 'sudh')
```

In [17]:

```
1 t1[-1]
```

Out[17]:

```
True
```

In [19]:

```
1 t1[::-2]
```

Out[19]:

```
('sudh', 34.08, True)
```

In [20]:

```
1 l1
```

Out[20]:

```
['sudh', 345, 34.08, (45+8j), True]
```

In [21]:

```
1 l = [1,2,3,4,5]
```

In [22]:

```
1 l1[0] ='kumar'
```

In [23]:

```
1 l1
```

Out[23]:

```
['kumar', 345, 34.08, (45+8j), True]
```

In [24]:

```
1 t1[0]
```

Out[24]:

```
'sudh'
```

In [25]:

```
1 t1
```

Out[25]:

```
('sudh', 345, 34.08, (45+8j), True)
```

In [26]:

```
1 t1[0] ='kumar'
```

-

TypeError

Traceback (most recent call last)

t)

C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19140/3903255441.py in <module>

----> 1 t1[0] ='kumar'

TypeError: 'tuple' object does not support item assignment

In [27]:

```
1 t1
```

Out[27]:

```
('sudh', 345, 34.08, (45+8j), True)
```

In [28]:

```
1 t2 = (34,45,3,4)
```

In [30]:

```
1 t1+t2
```

Out[30]:

```
('sudh', 345, 34.08, (45+8j), True, 34, 45, 3, 4)
```

In [31]:

```
1 t1*2
```

Out[31]:

```
('sudh', 345, 34.08, (45+8j), True, 'sudh', 345, 34.08, (45+8j), True)
```

In [33]:

```
1 l1+l
```

Out[33]:

```
['kumar', 345, 34.08, (45+8j), True, 1, 2, 3, 4, 5]
```

In [34]:

```
1 t1.count(1)
```

Out[34]:

```
1
```

In [35]:

```
1 t1.index('sudh')
```

Out[35]:

```
0
```

In [36]:

```
1 t = (12,3,4,(3,4,5,7), 'sudh')
```

In [37]:

```
1 t
```

Out[37]:

```
(12, 3, 4, (3, 4, 5, 7), 'sudh')
```

In [38]:

```
1 t1 = ([3,4,54,5],('sdsf', 343.35), "raksha")
```

In [41]:

```
1 t1[0][2] = 'sudh'
```

In [42]:

```
1 t1
```

Out[42]:

```
([3, 4, 'sudh', 5], ('sdsf', 343.35), 'raksha')
```

In [43]:

```
1 t1 = (3,4,5)
```

In [44]:

```
1 l = list(t1)
```

In [45]:

```
1 l
```

Out[45]:

```
[3, 4, 5]
```

In [46]:

```
1 l = [1,2,3,4,5,5,6,7,7,7,8,8,9,0,2,1,3,4,8]
```

In [47]:

```
1 s= set(l)
```

In [48]:

```
1 s
```

Out[48]:

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [49]:

```
1 s = {}
```

In [50]:

```
1 type(s)
```

Out[50]:

dict

In [51]:

```
1 s1 = {2,3,4}
```

In [52]:

```
1 type(s1)
```

Out[52]:

set

In [53]:

```
1 s2 = {2,3,4,1,4,6,7,8,9,2}
```

In [54]:

```
1 s2
```

Out[54]:

{1, 2, 3, 4, 6, 7, 8, 9}

In [56]:

```
1 s1
```

Out[56]:

{2, 3, 4}

In [57]:

```
1 s2
```

Out[57]:

{1, 2, 3, 4, 6, 7, 8, 9}

In [58]:

```
1 s2[0]
```

-
TypeError

Traceback (most recent call last)

C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19140/2344841457.py in <module>
----> 1 s2[0]

TypeError: 'set' object is not subscriptable

In [59]:

```
1 list(s2)
```

Out[59]:

```
[1, 2, 3, 4, 6, 7, 8, 9]
```

In [60]:

```
1 s2
```

Out[60]:

```
{1, 2, 3, 4, 6, 7, 8, 9}
```

In [61]:

```
1 s2.add('sudh')
```

In [62]:

```
1 s2
```

Out[62]:

```
{1, 2, 3, 4, 6, 7, 8, 9, 'sudh'}
```

In [63]:

```
1 s2.add(345)
```

In [64]:

```
1 s2
```

Out[64]:

```
{1, 2, 3, 345, 4, 6, 7, 8, 9, 'sudh'}
```

In [65]:

```
1 s2.add([1,2,3])
```

-
TypeError
t)

Traceback (most recent call last)

C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19140/2085173000.py in <module>
----> 1 s2.add([1,2,3])

TypeError: unhashable type: 'list'

In [66]:

```
1 s = {[3,2,4,2], 3,4,5}
```

-
TypeError
t)

Traceback (most recent call last)

C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19140/66843725.py in <module>
----> 1 s = {[3,2,4,2], 3,4,5}

TypeError: unhashable type: 'list'

In [67]:

```
1 s = {(3,2,4,2), 3,4,5}
```

In [68]:

```
1 s
```

Out[68]:

{(3, 2, 4, 2), 3, 4, 5}

In [75]:

```
1 s = {(3,2,4,5),(3,2,4,5), 3,4,5,3}
```

In [76]:

```
1 s
```

Out[76]:

{(3, 2, 4, 5), 3, 4, 5}

In [77]:

```
1 s.remove(4)
```

In [78]:

```
1 s
```

Out[78]:

```
{(3, 2, 4, 5), 3, 5}
```

In [79]:

```
1 s.discard(5)
```

In [80]:

```
1 s
```

Out[80]:

```
{(3, 2, 4, 5), 3}
```

In [81]:

```
1 s.remove(45)
```

```
-----
-
KeyError Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19140/2010124880.py in <module>
----> 1 s.remove(45)

KeyError: 45
```

In [82]:

```
1 s.discard(45)
```

In [83]:

```
1 s
```

Out[83]:

```
{(3, 2, 4, 5), 3}
```

In [84]:

```
1 s = {"sudh" , 'Sudh'}
```

In [85]:

```
1 s
```

Out[85]:

```
{'Sudh', 'sudh'}
```

In [86]:

```
1 s = {4,5,6,7, "sudh", "sudh", 3,4,5,2,7}
```

In [87]:

```
1 s
```

Out[87]:

```
{2, 3, 4, 5, 6, 7, 'sudh'}
```

In [88]:

```
1 l = [3,4,5,6]
```

In [90]:

```
1 set(l)
```

Out[90]:

```
{3, 4, 5, 6}
```

In [91]:

```
1 d = {}
```

In [92]:

```
1 type(d)
```

Out[92]:

```
dict
```

In [94]:

```
1 d = {4:"sudh"}
```

In [95]:

```
1 d
```

Out[95]:

```
{4: 'sudh'}
```

In [101]:

```
1 d1= { "key1": 1234, "key2": "sudh", 45 : [3,4,5,6]}
```

In [102]:

```
1 d1
```

Out[102]:

```
{'key1': 1234, 'key2': 'sudh', 45: [3, 4, 5, 6]}
```

In [103]:

```
1 l = [3,4,5,6]
```

In [104]:

```
1 d1
```

Out[104]:

```
{'key1': 1234, 'key2': 'sudh', 45: [3, 4, 5, 6]}
```

In [105]:

```
1 l[0]
```

Out[105]:

```
3
```

In [106]:

```
1 d1['key1']
```

Out[106]:

```
1234
```

In [107]:

```
1 d1[45]
```

Out[107]:

```
[3, 4, 5, 6]
```

In [111]:

```
1 d = {3: ["ksfasfh", "ddsf" , 4,5,3,3] }
```

In [112]:

```
1 d = {3 : (3,4,5)}
```

In [113]:

```
1 d = {"key" : {4,3,3}}
```

In [114]:

```
1 d1 = {"key1" : [2,3,4,5] , "key2" : "sudh", "key1" : 45}
```

In [115]:

```
1 d1["key1"]
```

Out[115]:

45

In [116]:

```
1 d1 = {"key1" : [2,3,4,5] , "key2" : "sudh", "kumar" : 45}
```

In [117]:

```
1 d1["key1"]
```

Out[117]:

[2, 3, 4, 5]

In [119]:

```
1 d = {'name' : "sudh", "mo_no": 2342, "mail_id": "sudh@gmail.com", 'key1' : [4,3,5,2]}
```

In [120]:

```
1 d
```

Out[120]:

```
{'name': 'sudh',
'mo_no': 2342,
'mail_id': 'sudh@gmail.com',
'key1': [4, 3, 5, 2],
'key2': (3, 4, 5),
'key3': {4, 5, 6},
'key4': {1: 4, 2: 4}}
```

In [121]:

```
1 d['key3']
```

Out[121]:

{4, 5, 6}

In [122]:

```
1 type(d['key3'])
```

Out[122]:

set

In [124]:

```
1 d["key4"][2]
```

Out[124]:

4

In [125]:

```
1 d.keys()
```

Out[125]:

dict_keys(['name', 'mo_no', 'mail_id', 'key1', 'key2', 'key3', 'key4'])

In [126]:

```
1 d.values()
```

Out[126]:

dict_values(['sudh', 2342, 'sudh@gmail.com', [4, 3, 5, 2], (3, 4, 5), {4, 5, 6}, {1: 4, 2: 4}])

In [127]:

```
1 d.items()
```

Out[127]:

dict_items([('name', 'sudh'), ('mo_no', 2342), ('mail_id', 'sudh@gmail.co m'), ('key1', [4, 3, 5, 2]), ('key2', (3, 4, 5)), ('key3', {4, 5, 6}), ('key4', {1: 4, 2: 4}))

In [128]:

```
1 d = {'key1' : 'sudh', "key2" : [1,2,3,4]}
```

In [129]:

```
1 d
```

Out[129]:

{'key1': 'sudh', 'key2': [1, 2, 3, 4]}

In [133]:

```
1 d["key3"] = "kumar"
```

In [134]:

```
1 d
```

Out[134]:

```
{'key1': 'sudh', 'key2': [1, 2, 3, 4], 'key3': 'kumar'}
```

In [135]:

```
1 d[4]= [1,2,3,4,5]
```

In [136]:

```
1 d
```

Out[136]:

```
{'key1': 'sudh', 'key2': [1, 2, 3, 4], 'key3': 'kumar', 4: [1, 2, 3, 4, 5]}
```

In [137]:

```
1 d['key1']="fsdsfsds"
```

In [138]:

```
1 d
```

Out[138]:

```
{'key1': 'fsdsfsds', 'key2': [1, 2, 3, 4], 'key3': 'kumar', 4: [1, 2, 3, 4, 5]}
```

In [139]:

```
1 del d['key1']
```

In [140]:

```
1 d
```

Out[140]:

```
{'key2': [1, 2, 3, 4], 'key3': 'kumar', 4: [1, 2, 3, 4, 5]}
```

In [141]:

```
1 del d
```

In [142]:

```
1 d
```

```
-  
NameError Traceback (most recent call last)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19140/3161387801.py in <module>  
----> 1 d
```

NameError: name 'd' is not defined

In [145]:

```
1 d1 = {'key1': 'sudh', 'key2': [3, 4, 5]}
```

In [146]:

```
1 d1
```

Out[146]:

```
{'key1': 'sudh', 'key2': [3, 4, 5]}
```

In [147]:

```
1 d1[[1, 2, 3]] = "inuer"
```

```
-  
TypeError Traceback (most recent call last)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19140/409832875.py in <module>  
----> 1 d1[[1, 2, 3]] = "inuer"
```

TypeError: unhashable type: 'list'

In [149]:

```
1 d1 = {'key1': 'sudh', 'key2': [3, 4, 5]}
```

In [152]:

```
1 d1[(1, 2, 3)] = "fsdfsds"
```

In [153]:

```
1 d1
```

Out[153]:

```
{'key1': 'sudh', 'key2': [3, 4, 5], (1, 2, 3): 'fsdfsds'}
```

In [154]:

```
1 d1.get("key1")
```

Out[154]:

```
'sudh'
```

In [155]:

```
1 d1 = {"key1" : 'Raksha', "key2" : 'mahajan'}
```

In [156]:

```
1 d2 = {"key3": 345, "key4" : [1,2,3,4]}
```

In [157]:

```
1 d1.update(d2)
```

In [158]:

```
1 d1
```

Out[158]:

```
{'key1': 'Raksha', 'key2': 'mahajan', 'key3': 345, 'key4': [1, 2, 3, 4]}
```

In [159]:

```
1 d2
```

Out[159]:

```
{'key3': 345, 'key4': [1, 2, 3, 4]}
```

In [160]:

```
1 d1+d2
```

```
-----  
-  
TypeError                                     Traceback (most recent call last  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19140\833095442.py in <modu  
le>  
----> 1 d1+d2
```

```
TypeError: unsupported operand type(s) for +: 'dict' and 'dict'
```

In []:

```
1
```


In [43]:

```
1 s = 'this is basic python class'
```

In []:

```
1
```

In [44]:

```
1 len(s)
```

Out[44]:

26

In [45]:

```
1 count = 0
2 for i in s:
3     #count = count+1;
4     count+=1
5 print(count)
6
```

26

In [46]:

```
1 range(len(s))
```

Out[46]:

range(0, 26)

In [47]:

```
1 s[::-1]
```

Out[47]:

'ssalc nohtyp cisab si siht'

In [6]:

```
1 for i in range(len(s)-1,-1,-1):
2     print(i, end = ' ')
```

25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

In [4]:

```
1 for i in range(len(s)-1,-1,-1):
2     print(s[i], end= " ")
3
```

s s a l c n o h t y p c i s a b s i s i h t

In [2]:

```
1 s = 'this is basic python class'
2 i = len(s)-1
3 while i>=0:
4     print(s[i], end = "")
5     i = i-1
```

ssalc nohtyp cisab si siht

In [7]:

```
1 len(s)
```

Out[7]:

26

In [8]:

```
1 s = 'ineuron'
2 v = 'AaEeIiOoUu'
```

In [24]:

```
1 s='ineuron'
2 for i in s :
3     if i in ('a','e','i','o','u'):
4         print('it is vowel', i)
5     else :
6         print('not a vowel')
```

it is vowel i
not a vowel
it is vowel e
it is vowel u
not a vowel
it is vowel o
not a vowel

In [17]:

```
1 for i in s :  
2     if i == 'a' or i =='e' or i =='i' or i == 'o' or i == 'u':  
3         print('its a vowel')  
4     else:  
5         print('not a vowel')
```

```
its a vowel  
its a vowel  
its a vowel  
its a vowel  
not a vowel
```

In [18]:

```
1 s = 'ineuron'  
2 v = 'AaEeIiOoUu'  
3  
4 for i in s :  
5     if i in v:  
6         print('its a vowel', i)  
7     else:  
8         print('not vowel')
```

```
its a vowel i  
not vowel  
its a vowel e  
its a vowel u  
not vowel  
its a vowel o  
not vowel
```

In [19]:

```
1 s in 'Sudh'
```

Out[19]:

False

In [21]:

```
1 's' in 'sudh'
```

Out[21]:

True

In [27]:

```
1 s = 'shikhar'
```

In [28]:

```
1 s
```

Out[28]:

```
'shikhar'
```

In [31]:

```
1 s = 'eye'
2 for i in s :
3     temp = print(i)
4
```

```
['e']
['y']
['e']
```

In [36]:

```
1 s = 'raksha'
2 for i in s :
3     temp
```

```
-
```

TypeError Traceback (most recent call last)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_10504/594548326.py in <module>
 1 s = 'raksha'
 2 for i in s :
----> 3 temp = i+1
 4 print(temp)

TypeError: can only concatenate str (not "int") to str

In [54]:

```
1 s = input()
2 v = s[::-1]
3 if s == v :
4     print('its a palindrome')
5 else :
6     print('not a palindrome')
```

```
eye
its a palindrome
```

In [57]:

```
1 s = input()
2 if s == s[::-1]:
3     print('its a palindrome')
4 else:
5     print('not a palindrome')
```

eye
its a palindrome

In [64]:

```
1 s = 'eye'
2 for i in range(len(s)):
3     print(len(s)-1-i)
```

2
1
0

In [70]:

```
1 s = 'eyeee'
2 for i in range(len(s)):
3     if s[i] == s[len(s)-1-i]:
4         print(s, 'is a palindrome')
5         break
6     else:
7         print('not a palindrome')
```

eyeee is a palindrome

In [81]:

```
1 s = input()
2 for i in range(len(s)):
3     if s[i]!= s[len(s)-1-i]:
4         print(s, 'not a palindrome')
5         break
6     else:
7         print('palindrome')
```

eye
palindrome

In [85]:

```
1 s = input()
2 f = ""
3 i = len(s)-1
4
5 while i>=0:
6     f = f+s[i]
7     i=i-1
8 if(s==f):
9     print('its a palindrome')
10 else :
11     print('not a palindrome')
```

eye
its a palindrome

In [103]:

```
1 s = 'eye'
2 k = 0
3 for i in range(len(s)):
4     if s[i] == s[len(s)-1-i]:
5         continue
6     else:
7         print('not palindrome')
8         k=1
9         break
10 if k == 0:
11     print('palindrome')
```

palindrome

In [115]:

```
1 d = {'India' : 'In',
2      "canada" : 'CA',
3      "China" : 'CH',
4      "united state" : 'US'
5 }
```

In [116]:

```
1 d.keys()
```

Out[116]:

```
dict_keys(['India', 'canada', 'China', 'united state'])
```

In [131]:

```
1 for i in d:  
2     if i == 'India':  
3         print('its available')  
4         break  
5     else:  
6         print('not available')
```

its available

In [125]:

```
1 for i in d:  
2     print(i)
```

India
canada
China
united state

In [132]:

```
1 'India' in d
```

Out[132]:

True

In [137]:

```
1 l1=[]  
2 l2=[]  
3 for i in d:  
4     if len(i) >5:  
5         l2.append(i)  
6     else:  
7         l1.append(i)  
8 print(l1)  
9 print(l2)
```

['India', 'China']
['canada', 'united state']

In [139]:

```
1 d
```

Out[139]:

{'India': 'In', 'canada': 'CA', 'China': 'CH', 'united state': 'US'}

In [140]:

```
1 d.items()
```

Out[140]:

```
dict_items([('India', 'In'), ('canada', 'CA'), ('China', 'CH'), ('united state', 'US')])
```

In [146]:

```
1 for i in d:  
2     print(i,d[i])
```

```
India In  
canada CA  
China CH  
united state US
```

In [33]:

```
1 d1 = {"ineuron": {  
2                 "a" : 14,  
3                 "b" : 10,  
4                 "c" : 4  
5             },  
6             "course" : {  
7                 'd': 45,  
8                 "e" : 34,  
9                 "f":1  
10            }  
11         }  
12     }
```

In [34]:

```
1 for i in d1:  
2     print(i)
```

```
ineuron  
course
```

In [11]:

```
1 len(d1)
```

Out[11]:

```
2
```

In [28]:

```
1 for i in d1:  
2     print(i, d1[i])  
3
```

```
ineuron {'a': 14, 'b': 10, 'c': 4}  
course {'d': 45, 'e': 34, 'f': 1}
```

In [29]:

```
1 d1.values()
```

Out[29]:

```
dict_values([{'a': 14, 'b': 10, 'c': 4}, {'d': 45, 'e': 34, 'f': 1}])
```

In [38]:

```
1 res = []
2 for key, val in d1.items():
3     max = 0
4     for i in val.values():
5         if i > max:
6             max = i
7     res[key] = max
8 print(res)
```

```
{'ineuron': 14, 'course': 45}
```

In [55]:

```
1 max = 0
2 for i in d1.values():
3     for j in i.values():
4         if j > max:
5             max = j
6 print(max)
```

```
14
45
```

In [54]:

```
1 for i in d1.values():
2     print(i)
```

```
{'a': 14, 'b': 10, 'c': 4}
{'d': 45, 'e': 34, 'f': 1}
```

In [56]:

```
1 for i in d1.values():
2     for j in i.values():
3         print(j)
```

```
14
10
4
45
34
1
```

In [76]:

```
1 l =[]
2 for i in d1.values():
3     for j in i.values():
4         l.append(j)
5 print(l)
```

```
[14, 10, 4]
[14, 10, 4, 45, 34, 1]
```

In [35]:

```
1 for i in d1.values():
2     print(max(i.values()))
```

```
14
45
```

In [38]:

```
1 for i in d1.values():
2     print(max(i.values()))
```

```
14
45
```

In [39]:

```
1 max(25,10)
```

Out[39]:

```
25
```

In [4]:

```
1 d1 = {"ineuron": {
2             "a" : 14,
3             "b" : 10,
4             "c" : 4
5         },
6         "course" : {
7             'd': 45,
8             "e" : 34,
9             "f":1
10            },
11         "g" : 55,
12
13         "h" : [34,24,56,78],
14         "i" : (45,67,87),
15         "k" : "sudh"
16     }
```

In [5]:

```
1 l1 = []
2 max = 0
3 for k ,v in d1.items():
4     print(k,v)
5
```

```
ineuron {'a': 14, 'b': 10, 'c': 4}
course {'d': 45, 'e': 34, 'f': 1}
g 55
h [34, 24, 56, 78]
i (45, 67, 87)
k sudh
```

In [27]:

```
1
2 for v in d1.values():
3     print(v)
```

```
{'a': 14, 'b': 10, 'c': 4}
{'d': 45, 'e': 34, 'f': 1}
55
[34, 24, 56, 78]
(45, 67, 87)
sudh
```

In [8]:

```
1 l = [34, 24, 56, 78]
2 max(l)
```

```
-
```

TypeError Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_36516/569365669.py in <module>
 1 l = [34, 24, 56, 78]
----> 2 max(l)

TypeError: 'int' object is not callable

In [9]:

```
1 d1 = {"ineuron": {  
2     "a" : 14,  
3     "b" : 10,  
4     "c" : 4  
5 },  
6     "course" : {  
7         'd': 45,  
8         "e" : 34,  
9         "f":1  
10    },  
11    "g" : 55,  
12  
13    "h" : [34,24,56,78],  
14    "i" : (45,67,87),  
15    "k" : "sudh"  
16 }
```

In [10]:

```
1 l1 = []  
2 for i in d1.values():  
3     print(i)
```

```
{'a': 14, 'b': 10, 'c': 4}  
{'d': 45, 'e': 34, 'f': 1}  
55  
[34, 24, 56, 78]  
(45, 67, 87)  
sudh
```

In [11]:

```
1 for i in d1.values():
2     if type(i) == dict:
3         if i > max:
4             max = i
5             print(max)
6     if type(i) == list or type(i) == tuple:
7         if i > max:
8             max = i
9             print(max)
10    if type(i) == str:
11        pass
12    else:
13        max=i
14        print(max)
15
16 print(l1)
```

```
-----
-
TypeError                                 Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_36516\2127452035.py in <module>
      2 for i in d1.values():
      3     if type(i) == dict:
----> 4         if i > max:
      5             max = i
      6             print(max)

TypeError: '>' not supported between instances of 'dict' and 'int'
```

In []:

```
1
```

In [1]:

```
1 l = [1,2,3,4]
2 max (l)
```

Out[1]:

```
4
```

In [2]:

```
1
```

```
File "C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_34456\594370838.py", line 15
    "k" :
    ^
SyntaxError: unexpected EOF while parsing
```

In [3]:

```
1 d1 = {"ineuron": {  
2     "a" : 14,  
3     "b" : 10,  
4     "c" : 4  
5 },  
6     "course" : {  
7         'd': 45,  
8         "e" : 34,  
9         "f":1  
10    },  
11    "g" : 55,  
12  
13    "h" : [34,24,56,78],  
14    "i" : (45,67,87),  
15    "k" : "sudh"  
16 }
```

In [30]:

```
1 l1= []  
2 for i in d1.values():  
3     if type(i) == dict:  
4         l1.append(max(i.values()))  
5     if type(i) == list or type(i) == tuple:  
6         l1.append(max(i))  
7     if type(i) == str:  
8         pass  
9     if type(i) == int:  
10        l1.append(i)  
11 print(max(l1))
```

87

In [31]:

```
1 l1
```

Out[31]:

[14, 45, 55, 78, 87]

In [32]:

```
1 max(l1)
```

Out[32]:

87

In []:

```
1
```

In [2]:

```
1 a = 5
2 a/10
```

Out[2]:

0.5

In [4]:

```
1 a/0
2 print('this is my prog')
```

```
-----
-
ZeroDivisionError                                Traceback (most recent call last)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_13728/3450635350.py in <module>
      1 a/0
      2 print('this is my prog')

ZeroDivisionError: division by zero
```

In [6]:

```
1 a = 5
2 b = int(input())
3 a/b
```

```
0
-----
-
ZeroDivisionError                                Traceback (most recent call last)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_13728/4083926819.py in <module>
      1 a = 5
      2 b = int(input())
----> 3 a/b

ZeroDivisionError: division by zero
```

In [8]:

```
1 f = open('test.txt', 'w')
```

In [9]:

```
1 f.write('fsdsfdsfsa')
```

Out[9]:

10

In [10]:

```
1 f.close()
```

In [11]:

```
1 f = open('test.txt', 'r')
```

In [13]:

```
1 f.write('fsfasdf sdfasdf dsfasf')
```

```
-----
-
UnsupportedOperation                               Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_13728\1887384898.py in <module>
----> 1 f.write('fsfasdf sdfasdf dsfasf')

UnsupportedOperation: not writable
```

In [16]:

```
1 try :
2     f = open('test.txt', "r")
3     f.write('fsfsdsfa')
4
5 except :
6     print('there was a mistake')
7 print('this is my code')
```

```
there was a mistake
this is my code
```

In [24]:

```
1 l = [4,5,6,7,6,8,0]
2
3 try :
4     for i range(len(l)):
5         print(l)
6 except:
7     print('this is my code')
```

```
File "C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_13728\4068923912.py", line 4
    for i range(len(l)+1):
               ^
SyntaxError: invalid syntax
```

In [27]:

```
1 l = [4,5,6,7,6,8,0]
2 for i in range(len(l)+1):
3     print(l[i])
```

```
4
5
6
7
8
0

-----
-
IndexError                                     Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_13728/795944826.py in <module>
      1 l = [4,5,6,7,6,8,0]
      2 for i in range(len(l)+1):
----> 3     print(l[i])

IndexError: list index out of range
```

In [29]:

```
1 l = [4,5,6,7,6,8,0]
2 try :
3     for i in range(len(l)+1):
4         print(l[i])
5 except :
6     print('this is my code')
7 print('fdsfasdfasdffsadsafasd')
```

```
4
5
6
7
8
0
this is my code
fdsfasdfasdffsadsafasd
```

In [30]:

```
1 5/0
```

```
-  
ZeroDivisionError                                Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_13728\2874912419.py in <module>  
----> 1 5/0
```

```
ZeroDivisionError: division by zero
```

In [32]:

```
1 l = [4,5,6,7,6,8,0]  
2 try :  
3     for i in range(len(l)+1):  
4         print(l[i])  
5 except Exception as e:  
6     print( e)  
7 print('fdsfasdfasdffsadfadsafasd')
```

```
4  
5  
6  
7  
8  
0  
list index out of range  
fdsfasdfasdffsadfadsafasd
```

In [34]:

```
1 try :  
2     a = int(input())  
3     b = int(input())  
4  
5 except Exception as e:  
6     print(e)  
7  
8 print('my code is working even after error')
```

```
fasfa  
invalid literal for int() with base 10: 'fasfa'  
my code is working even after error
```

In [38]:

```
1 try :
2     f = open('test' , 'r')
3     f.write('this is my code with except handling')
4     print('this is my code after write ops')
5
6 except Exception as e :
7     print(e)
8 try:
9     l = [1,2,3,4]
10    for i in l:
11        print(i)
12 except :
13     print('this is my handler for loop')
```

[Errno 2] No such file or directory: 'test'

1
2
3
4

In [42]:

```
1 try :
2     d = {'key1' : 'sudh' , 'key2': [1,2,3,4,5], 'key3' : (4,5,6,7)}
3     d['key4'] : int(input())
4
5 except Exception as e:
6     print(e)
7
```

tew
invalid literal for int() with base 10: 'tew'

In [44]:

```
1 d = {'key1' : 'sudh' , 'key2': [1,2,3,4,5], 'key3' : (4,5,6,7)}
2 d['key4'] : int(input())
```

rew

```
-----
-
ValueError                                                 Traceback (most recent call las
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_13728/3949761922.py in <mod
ule>
    1 d = {'key1' : 'sudh' , 'key2': [1,2,3,4,5], 'key3' : (4,5,6,7)}
----> 2 d['key4'] : int(input())

ValueError: invalid literal for int() with base 10: 'rew'
```

In [48]:

```
1 try :  
2     d = {'key1' : 'sudh' , 'key2': [1,2,3,4,5], 'key3' : (4,5,6,7)}  
3     d['key4'] : int(input())  
4     f = open('test3.txt' , 'r')  
5 except ValueError as e:  
6     print(e)  
7
```

44

```
-----  
-  
FileNotFoundException          Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_13728/876450867.py in <module>  
    2     d = {'key1' : 'sudh' , 'key2': [1,2,3,4,5], 'key3' : (4,5,6,  
7)}  
    3     d['key4'] : int(input())  
----> 4     f = open('test3.txt' , 'r')  
    5 except ValueError as e:  
    6     print(e)
```

FileNotFoundException: [Errno 2] No such file or directory: 'test3.txt'

In [49]:

```
1 try :  
2     d = {'key1' : 'sudh' , 'key2': [1,2,3,4,5], 'key3' : (4,5,6,7)}  
3     d['key4'] : int(input())  
4     f = open('test3.txt' , 'r')  
5 except Exception as e:  
6     print(e)  
7  
8
```

44

[Errno 2] No such file or directory: 'test3.txt'

In [50]:

```
1 try :  
2     d = {'key1' : 'sudh' , 'key2': [1,2,3,4,5], 'key3' : (4,5,6,7)}  
3     d['key4'] : int(input())  
4     f = open('test3.txt' , 'r')  
5 except ValueError as e:  
6     print(e)  
7  
8 except FileNotFoundError as e:  
9     print(e)
```

66

[Errno 2] No such file or directory: 'test3.txt'

In [54]:

```
1 try :  
2     d = {'key1' : 'sudh' , 'key2': [1,2,3,4,5], 'key3' : (4,5,6,7)}  
3     d['key4'] : int(input())  
4     f = open('test3.txt' , 'r')  
5  
6 except Exception as ee:  
7     print('This is my exception class',ee)  
8 except ValueError as e:  
9     print(e)  
10  
11 except FileNotFoundError as e:  
12     print(e)
```

88

This is my exception class [Errno 2] No such file or directory: 'test3.txt'

In [57]:

```
1 try :  
2     d = {'key1' : 'sudh' , 'key2': [1,2,3,4,5], 'key3' : (4,5,6,7)}  
3     d['key4'] : int(input())  
4     f = open('test3.txt' , 'r')  
5  
6 except ValueError as e:  
7     print(e)  
8  
9 except FileNotFoundError as e:  
10    print('testing2',e)  
11  
12 except Exception as ee:  
13     print('This is my exception class',ee)  
14
```

43

testing2 [Errno 2] No such file or directory: 'test3.txt'

In [61]:

```
1 try :  
2     f = open('test1.txt' , 'w')  
3     f.write('this is my code in try')  
4  
5 except Exception as e:  
6     print('this will handle an error' ,e )  
7  
8 else:  
9     print('this will execute once my try block will be executed successfully')  
10    f.close()
```

this will execute once my try block will be executed successfully

In [64]:

```
1 try :  
2     f = open('test4.txt' , 'r')  
3 except Exception as e:  
4     print('fadfas',e)  
5 else:  
6     print('do this on successful execution of try block')  
7  
8 finally :  
9     print('do this for sure, This will execute all the time even if try fail or pass')  
10  
11
```

fadfas [Errno 2] No such file or directory: 'test4.txt'
do this for sure, This will execute all the time even if try fail

In [66]:

```
1 try :  
2     f = open('test4.txt' , 'w')  
3 except Exception as e:  
4     print('fadfas',e)  
5 else:  
6     print('do this on successful execution of try block')  
7  
8 finally :  
9     print('do this for sure, This will execute all the time even if try fail')  
10
```

do this on successful execution of try block
do this for sure, This will execute all the time even if try fail

In [67]:

```
1 try :
2     f = open('test4.txt' , 'w')
3 except Exception as e:
4     print('fadfas',e)
5 else:
6     print('do this on successful execution of try block')
7
8 finally :
9     print('do this for sure, This will execute all the time even if try fail')
10    f = open('test5.txt' , 'r')
```

do this on successful execution of try block
do this for sure, This will execute all the time even if try fail

```
-  
  
FileNotFoundException                                     Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_13728/493838449.py in <module>  
      8 finally :  
      9     print('do this for sure, This will execute all the time even if  
f try fail')  
---> 10    f = open('test5.txt' , 'r')  
  
FileNotFoundException: [Errno 2] No such file or directory: 'test5.txt'
```

In [68]:

```
1 try :
2     f = open('test4.txt' , 'w')
3 except Exception as e:
4     print('fadfas',e)
5 else:
6     print('do this on successful execution of try block')
7
8 finally :
9     print('do this for sure, This will execute all the time even if try fail')
10    try :
11        f = open('test5.txt' , 'r')
12    except Exception as e :
13        print(e)
```

do this on successful execution of try block
do this for sure, This will execute all the time even if try fail
[Errno 2] No such file or directory: 'test5.txt'

In [69]:

```
1 try :
2     f = open('test4.txt' , 'w')
3 except Exception as e:
4     print('fadfas',e)
5 else:
6     print('do this on successful execution of try block')
7
8 finally :
9     print('do this for sure, This will execute all the time even if try fail')
10    try :
11        f = open('test5.txt' , 'r')
12    except Exception as e :
13        print(e)
14
15 finally:
16     print('This will come to this block for sure')
```

```
do this on successful execution of try block
do this for sure, This will execute all the time even if try fail
[Errno 2] No such file or directory: 'test5.txt'
This will come to this block for sure
```

In [73]:

```
1 def askint():
2     a = int(input())
3     return a
4
5 askint()

fas
-----
-
ValueError                                Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_13728/2961965484.py in <module>
      3     return a
      4
----> 5 askint()

C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_13728/2961965484.py in askint()
      1 def askint():
----> 2     a = int(input())
      3     return a
      4
      5 askint()

ValueError: invalid literal for int() with base 10: 'fas'
```

In [74]:

```
1 def askint():
2     try :
3         a = int(input())
4         return a
5     except Exception as e:
6         print(e)
7
8 askint()
```

```
tsfas
invalid literal for int() with base 10: 'tsfas'
```

In [79]:

```
1 def askint():
2     try:
3         while True :
4             a = int(input('Enter the number:'))
5     except Exception as e:
6         print(e)
7 askint()
```

```
Enter the number:fdf
invalid literal for int() with base 10: 'fdf'
```

In [82]:

```
1 def askint():
2     flag = True
3     while flag :
4         try :
5             a = int(input('Enter the number:'))
6             return a
7             flag = 'false'
8         except Exception as e:
9             print(e)
10 askint()
```

```
Enter the number:te
invalid literal for int() with base 10: 'te'
Enter the number:tre
invalid literal for int() with base 10: 'tre'
Enter the number:ete
invalid literal for int() with base 10: 'ete'
Enter the number:ete
invalid literal for int() with base 10: 'ete'
Enter the number:5
```

Out[82]:

5

In [3]:

```
1 def integer():
2     while True:
3         '''This function asks for an integer unless provided otherwise'''
4         try:
5             x = int(input())
6             break
7         except Exception as e:
8             print('enter the input again :- ',e)
9
10 integer()
```

```
enter the input again :- invalid literal for int() with base 10: ''
5
```

In [5]:

```
1 def getInt():
2     while True:
3         try:
4             a = int(input())
5         except Exception as e:
6             print(e)
7             continue
8         if a < 0 :
9             print('enter a positive no')
10            continue
11        else:
12            break
13
14    return a
15
16 getInt()
```

```
f
invalid literal for int() with base 10: 'f'
f
invalid literal for int() with base 10: 'f'
f
invalid literal for int() with base 10: 'f'
-5
enter a positive no
5
```

In [8]:

```
1 def askint():
2     while True:
3         try:
4             a = int(input())
5             break
6         except Exception as e:
7             print('Enter a number only' , e)
8             continue
9 askint()
```

```
ff
Enter a number only invalid literal for int() with base 10: 'ff'
ff
Enter a number only invalid literal for int() with base 10: 'ff'
ff
Enter a number only invalid literal for int() with base 10: 'ff'
d5
Enter a number only invalid literal for int() with base 10: 'd5'
5
```

In [9]:

```
1 a = 6/10
```

In [10]:

```
1 5/0
```

```
-----
-
ZeroDivisionError                                 Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_9160\2874912419.py in <module>
     1 5/0
-----> 1 5/0

ZeroDivisionError: division by zero
```

In [14]:

```
1 def test(a):
2     if a < 0:
3         raise Exception('you have entered a negative value' , a)
4     return a
```

In [15]:

```
1 test(-4)
```

```
-  
Exception Traceback (most recent call last)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_9160/2430745927.py in <module>  
----> 1 test(-4)  
  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_9160/3035122755.py in test(a)  
    1 def test(a):  
    2     if a < 0:  
----> 3         raise Exception('you have entered a negative value' , a)  
    4     return a  
  
Exception: ('you have entered a negative value', -4)
```

In [24]:

```
1 # using existing excep class and override it  
2 def test(a):  
3     if a < 0:  
4         raise ValueError('you have entered a negative value' , a)  
5     return a  
6
```

In [30]:

```
1 test(-4)
```

```
-  
ValueError Traceback (most recent call last)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_9160/2430745927.py in <module>  
----> 1 test(-4)  
  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_9160/626507129.py in test(a)  
    2 def test(a):  
    3     if a < 0:  
----> 4         raise ValueError('you have entered a negative value' , a)  
    5     return a  
  
ValueError: ('you have entered a negative value', -4)
```

In [31]:

```
1 try :  
2     a = int(input())  
3     test(a)  
4 except Exception as e:  
5     print('calling my raise exception',e)
```

-4

calling my raise exception ('you have entered a negative value', -4)

In []:

```
1
```

In [2]:

```
1 l = ['name','emailid','phoneno','address']
2 for i in l:
3     print(i + " sudh ")
```

name sudh
emailid sudh
phoneno sudh
address sudh

In [4]:

```
1 s = 'ineuron'
2 for i in s:
3     print(i)
```

i
n
e
u
r
o
n

In [5]:

```
1 l
```

Out[5]:

['name', 'emailid', 'phoneno', 'address']

In [7]:

```
1 for i in l :
2     print(i)
3
4 else:
5     print('if for loop is going to complete itself then only it will come to else')
```

name
emailid
phoneno
address
if for loop is going to complete itself then only it will come to else

In [8]:

```
1 for i in l:
2     if i == 'emailid':
3         break
4     print(i)
5 else:
6     print('check this statemet')
```

name

In [10]:

```
1 s = 'Rakshanda'
2 for i in s:
3     if i =='k':
4         break
5 else:
6     print('dont execute this unless and untill it is not printing my name')
```

In [11]:

```
1 a = 1
2 while a<6:
3     print(a)
4     a= a+1
```

```
1
2
3
4
5
```

In [12]:

```
1 a = 1
2 while a<5:
3     print(a)
4     if a == 4 :
5         break
6     a=a+1
7
```

```
1
2
3
4
```

In []:

```
1 a = 1
2 #while a < 5 :
3     print(a)
4     if a == 3:
5         continue
6     #a = a + 1
```

In [2]:

```
1 a = 1
2 while a < 5 :
3     print(a)
4     a = a + 1
5     if a == 3:
6         continue
7     #a = a + 1
```

```
1
2
3
4
```

In [1]:

```
1 while a < 5 :
2     pass
```

In [1]:

```
1 #a = 1
2 #while a < 5:
3     break
```

```
File "C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_26224/2373984773.py", line 3
    break
^
IndentationError: unexpected indent
```

In [2]:

```
1 range(6)
```

Out[2]:

```
range(0, 6)
```

In [3]:

```
1 list(range(6))
```

Out[3]:

```
[0, 1, 2, 3, 4, 5]
```

In [4]:

```
1 list(range(0,7))
```

Out[4]:

```
[0, 1, 2, 3, 4, 5, 6]
```

In [5]:

```
1 list(range(4,10))
```

Out[5]:

```
[4, 5, 6, 7, 8, 9]
```

In [6]:

```
1 list(range(3,20,2))
```

Out[6]:

```
[3, 5, 7, 9, 11, 13, 15, 17, 19]
```

In [8]:

```
1 list(range(3,10,-1))
```

Out[8]:

```
[]
```

In [10]:

```
1 list(range(10, 6, -1))
```

Out[10]:

```
[10, 9, 8, 7]
```

In [11]:

```
1 list(range(10 , -5 , -2 ))
```

Out[11]:

```
[10, 8, 6, 4, 2, 0, -2, -4]
```

In [12]:

```
1 for i in range(7):  
2     print(i)
```

```
0  
1  
2  
3  
4  
5  
6
```

In [17]:

```
1 n = 5
2 for i in range(0,n):
3     for j in range(0,i+1):
4         print("*", end = ' ')
5     print("\r")
```

```
*
```



```
**
```



```
***
```



```
****
```



```
*****
```

In [20]:

```
1 l = [1,2 ,3 ,45,67,8]
2 for i in l:
3     print(i)
4
5
```

```
1
2
3
45
67
8
```

In [24]:

```
1 t=(1,2,3,4,5,6,7)
2 for i in t:
3     print (i)
```

```
1
2
3
4
5
6
7
```

In [25]:

```
1 len(t)
```

Out[25]:

```
7
```

In [26]:

```
1 range(len(t))
```

Out[26]:

```
range(0, 7)
```

In [27]:

```
1 t=(1,2,3,4,5,6,7)
2 for i in range(len(t)):
3     print(i , t[i])
4
5
```

```
0 1
1 2
2 3
3 4
4 5
5 6
6 7
```

In [37]:

```
1 range(len(t),0,-1)
```

Out[37]:

```
range(7, 0, -1)
```

In [34]:

```
1 t
```

Out[34]:

```
(1, 2, 3, 4, 5, 6, 7)
```

In [36]:

```
1 t=(1,2,3,4,5,6,7)
2 for i in range(len(t),0,-1):
3     print(i)
4
5
```

```
7
6
5
4
3
2
1
```

In [39]:

```
1 list(range(len(t), 0, -1))
```

Out[39]:

```
[7, 6, 5, 4, 3, 2, 1]
```

In [40]:

```
1 t=(1,2,3,4,5,6,7)
2 for i in range(len(t),0,-1):
3     print(t[i])
```

```
- IndexError                                     Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_26224/3993085116.py in <module>
      1 t=(1,2,3,4,5,6,7)
      2 for i in range(len(t),0,-1):
----> 3     print(t[i])

IndexError: tuple index out of range
```

In [43]:

```
1 t = (3,4,4,5,6,7,7)
2 for i in range(len(t)-1, -1, -1):
3     print(t[i])
```

```
7
7
6
5
4
4
3
```

In [44]:

```
1 t = (3,4,4,5,6,7,7)
2 for i in range(len(t)-1, , -1):
3     print(t[i])
```

```
File "C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_26224/1135905079.py", line 2
    for i in range(len(t)-1, , -1):
                           ^
SyntaxError: invalid syntax
```

In [45]:

```
1 d = {'a' : 'fsdfs' , 'b': 'sdfsa' , "c" : [1,2,3,4] , "d" : (4,5,6,7) , "e" : "sudh"
```

In [46]:

```
1 for i in d:  
2     print(i)
```

a
b
c
d
e

In [47]:

```
1 d['d']
```

Out[47]:

(4, 5, 6, 7)

In [49]:

```
1 for i in d:  
2     print(i, d[i])
```

a fsdfs
b sdfsa
c [1, 2, 3, 4]
d (4, 5, 6, 7)
e sudh

In [50]:

```
1 d.items()
```

Out[50]:

dict_items([('a', 'fsdfs'), ('b', 'sdfsa'), ('c', [1, 2, 3, 4]), ('d', (4, 5, 6, 7)), ('e', 'sudh')])

In [51]:

```
1 for i in d.items():  
2     print(i)
```

('a', 'fsdfs')
('b', 'sdfsa')
('c', [1, 2, 3, 4])
('d', (4, 5, 6, 7))
('e', 'sudh')

In [52]:

```
1 s = {3, 546, 56, 4, 5, 8, 9, 3, 3, 4, 2, 2, 1, 6, 7, 8, 9}
```

In [53]:

```
1 s
```

Out[53]:

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 56, 546}
```

In [54]:

```
1 for i in s :  
2     print(i)
```

```
1  
546  
3  
4  
5  
2  
6  
8  
9  
7  
56
```

In [55]:

```
1 l = [1,2,3,4,5,6,7]
```

In [56]:

```
1 l
```

Out[56]:

```
[1, 2, 3, 4, 5, 6, 7]
```

In [58]:

```
1 range(len(l))
```

Out[58]:

```
range(0, 7)
```

In [59]:

```
1 range(len(l),0,-1)
```

Out[59]:

```
range(7, 0, -1)
```

In [60]:

```
1 list[range(len(t),0,-1)]
```

Out[60]:

```
list[range(7, 0, -1)]
```

In [61]:

```
1 list(range(len(t), 0, -1))
```

Out[61]:

```
[7, 6, 5, 4, 3, 2, 1]
```

In [62]:

```
1 list(range(len(t),-1,-1))
```

Out[62]:

```
[7, 6, 5, 4, 3, 2, 1, 0]
```

In [63]:

```
1 s = {3,546,56,4,5,8,9,3,3,4,2,2,1,6,7,8,9}
```

In [64]:

```
1 len(s)
```

Out[64]:

```
11
```

In [65]:

```
1 i = len(s)-1
2 while i >=0:
3     print(s[i],end = " ")
```

```
-----
-
TypeError                                     Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_26224\752235568.py in <module>
      1 i = len(s)-1
      2 while i >=0:
----> 3     print(s[i],end = " ")
```

TypeError: 'set' object is not subscriptable

In [66]:

```
1 s
```

Out[66]:

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 56, 546}
```

In [68]:

```
1
```

```
{1, 546, 3, 4, 5, 2, 6, 8, 9, 7, 56}  
{1, 546, 3, 4, 5, 2, 6, 8, 9, 7, 56}  
{1, 546, 3, 4, 5, 2, 6, 8, 9, 7, 56}  
{1, 546, 3, 4, 5, 2, 6, 8, 9, 7, 56}  
{1, 546, 3, 4, 5, 2, 6, 8, 9, 7, 56}  
{1, 546, 3, 4, 5, 2, 6, 8, 9, 7, 56}  
{1, 546, 3, 4, 5, 2, 6, 8, 9, 7, 56}  
{1, 546, 3, 4, 5, 2, 6, 8, 9, 7, 56}  
{1, 546, 3, 4, 5, 2, 6, 8, 9, 7, 56}  
{1, 546, 3, 4, 5, 2, 6, 8, 9, 7, 56}
```

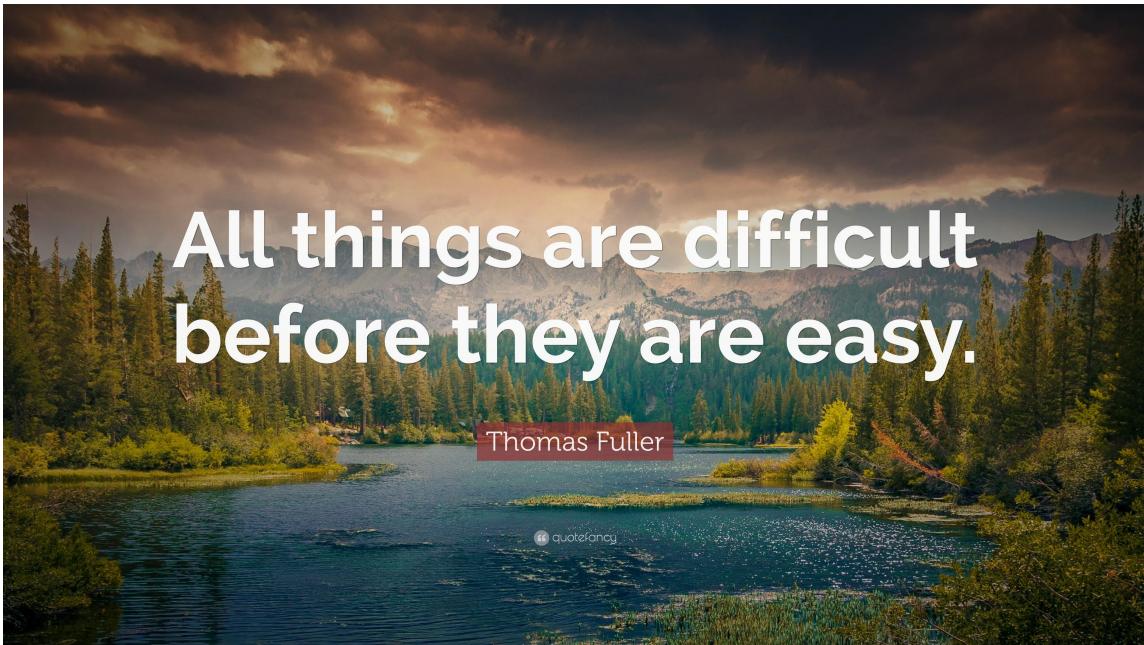
In []:

```
1
```

In [4]:

```
1 #10 . write a fun which will be able to read a image file and show it to you.
2 def fun10(filename):
3     '''This function will be able to read a image file and show it to you'''
4     import PIL
5     from PIL import Image
6     img = Image.open(filename, 'r')
7     return display(img)
8 file_name = input('Please enter the png file name along with .png extention: ')
9 fun10(file_name)
```

Please enter the png file name along with .png extention: Life_image.png



In [10]:

```
1 #12. write a func which can move a file from one directory to another directory
2 def fun12(src_path , dst_path):
3     '''This function will move a file from one directory to another directory'''
4     import shutil
5     shutil.move(src_path,dst_path)
6
7 src_path = input('Please enter the source directory path')
8 dst_path = input('Please enter the destination directory path')
9
10 #source path : D:\Python_Basics\Source\Subfolder
11 #Destination path : D:\Python_Basics\Dest
12
13 fun12(src_path,dst_path)
```

Please enter the source directory pathD:\Python_Basics\Source\Subfolder
Please enter the destination directory pathD:\Python_Basics\Dest

In [11]:

```
1 # 13 write a func which will be able to shutdown your system.
2 #def fun13():
3     '''This function will shutdown your system'''
4     import os
5     os.system("shutdown /s /t 1")
6
7 fun13()
8
```

```
File "C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_27484/3440336366.py", line 2
    '''This function will shutdown your system'''
    ^
IndentationError: unexpected indent
```

In [27]:

```
1 # importing all the required modules
2 import PyPDF2
3
4 # creating a pdf reader object
5 reader = PyPDF2.PdfReader('iCRM-Dec2022.pdf')
6
7 # print the number of pages in pdf file
8 print(len(reader.pages))
9
10 # print the text of the first page
11 print(reader.pages[0].extract_text())
```

1
12/5/22, 02:10 PM iCRM

ONEOTT iNTERTAINMENT LTD.

INCENTRE 49/50 MIDC, 12th Road , Andheri (East) , Mumbai -400093
GSTIN : 27AADCP6815A2Z0

RETAIL INVOICE

HSN/SAC
code

Description
of Goods

Qty

In [33]:

```
1 #17. write a func to read word file.
2 def fun17(filename):
3     '''This function will read a complete word file'''
4     f = open(filename+'.doc' , 'r')
5     print(f.read())
6
7 filename  = input('Please enter the word file name without extention: ')
8
9 # iCRM-converted_Feb2023
10
11 fun17(filename)
```

File "C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_8296/903973651.py",
line 3

```
    '''This function will read a complete word file'''
```

IndentationError: unexpected indent

In [30]:

```
1 import docx
2 def getText(filename):
3     doc = docx.Document(filename)
4     fullText = []
5     for para in doc.paragraphs:
6         fullText.append(para.text)
7     return '\n'.join(fullText)
8 print(getText('iCRM-converted_Feb2023.docx'))
```

2/3/23, 05:14 PM iCRM ONEOTT iNTERTAINMENT LTD. INCENTRE 49/50 MI
DC, 12th Road , Andheri (East) , Mumbai -400093 GSTIN : 27AADCP6815A2Z0
RETAIL INVOICE Scan this QR code to avail online payment options :RuPay
Debit Card BHIM UPI UPI QR code Comments Date Declaration: This is
computer generated invoice, No signature required
IRNNo: https://customer.onebroadband.in/QuotePrint.aspx?Id=_EP_CCtPPznTJLc|&t=_EP_VMxWJfHSw3w|&isThrowEx=_EP_bu326010zRE 1/1

In [32]:

```
1 doc = docx.Document('iCRM-converted_Feb2023.docx')
2 allText = []
3 for docpara in doc.paragraphs:
4     allText.append(docpara.text)
5 print(allText)
```

['2/3/23, 05:14 PM\\tiCRM', '', 'ONEOTT iNTERTAINMENT LTD.', '', 'INCENTRE
49/50 MIDC, 12th Road , Andheri (East) , Mumbai -400093 GSTIN : 27AADCP68
15A2Z0', '', 'RETAIL INVOICE', '', '', 'Scan this QR code to avail online
payment options :RuPay Debit Card', 'BHIM UPI', 'UPI QR code', '', '', '',
'Comments', '', '', 'Date', 'Declaration:', '', '', 'This is computer gene
rated invoice, No signature required', '\nIRNNo:', '', '', '', '', 'http
s://customer.onebroadband.in/QuotePrint.aspx?Id=_EP_CCtPPznTJLc|&t=_EP_VMx
WJfHSw3w|&isThrowEx=_EP_bu326010zRE|\\t1/1']

In [34]:

```
1 # 18. write a func which can help you to filter only word file from dict  
2
```

In [38]:

```
1 #19. write a func by which you can print an ip address of your system.  
2 def fun19():  
3     '''This function will print the IP address of your system'''  
4  
5     import socket  
6     name = socket.gethostname()  
7     IP_address = socket.gethostbyname(name)  
8     print(f"system name is {name} and IP address is {IP_address}")  
9  
10 fun19()
```

system name is DESKTOP-HGGE0RL and IP address is 192.168.1.8

In [41]:

```
1 #20. write a func by which you will be able to append two pdf files.  
2 from PyPDF2 import PdfMerger  
3  
4 pdfs = ['file1.pdf', 'file2.pdf']  
5  
6 merger = PdfMerger()  
7  
8 for pdf in pdfs:  
9     merger.append(pdf)  
10  
11 merger.write("result.pdf")  
12 merger.close()
```

In [53]:

```
1 #18. write a func which can help you to filter only word file from dict  
2  
3 import os  
4 def fun18a(dict_path = os.getcwd()):  
5     files = os.listdir()  
6  
7     for i in files:  
8         file_details = os.path.splitext(i)  
9         #print(file_details)  
10        if file_details[1] == '.docx':  
11            print(file_details[0], file_details[1])
```

In [54]:

```
1 fun18a()
```

iCRM-converted_Feb2023 .docx

In [60]:

```
1 # IMAGE
2 import cv2
3 def fun10a(img_path):
4     img = cv2.imread(img_path)
5     cv2.imshow('Image:', img)
6     cv2.waitKey(0)
7     cv2.destroyAllWindows()
8
9 fun10a('Life_image.png')
```

In [5]:

```
1 #11. write a func which can read video file and play for you.
2 import cv2
3 def fun11(video_path):
4     video = cv2.VideoCapture(video_path)
5
6     while(video.isOpened()):
7         ret,frame = video.read()
8         if ret == True:
9             cv2.imshow('Frame', frame)
10            if cv2.waitKey(25) & 0xFF ==ord('x'):
11                break
12            else:
13                break
14        video.release()
15        cv2.destroyAllWindows()
16 fun11('sample-5s.mp4')
```

In []:

```
1
```

In []:

```
1
```

In [1]:

```
1 len('sudh')
```

Out[1]:

4

In [2]:

```
1 print("sdasd")
```

sdasd

In [4]:

```
1 def test():
2     pass
```

In [16]:

```
1 def test1():
2     print('this is my first fun')
```

In [17]:

```
1 test1()
```

this is my first fun

In [18]:

```
1 a = test1()
```

this is my first fun

In [19]:

```
1 a + "sudh"
```

```
-----
-
TypeError                                 Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_38816/2234722170.py in <module>
      1 a + "sudh"
-----
```

TypeError: unsupported operand type(s) for +: 'NoneType' and 'str'

In [20]:

```
1 type(test1())
```

this is my first fun

Out[20]:

NoneType

In [21]:

```
1 str(a)
```

Out[21]:

'None'

In [22]:

```
1 def test2():
2     return "this is my first fun"
```

In [23]:

```
1 type(test2())
```

Out[23]:

str

In [25]:

```
1 test2()+' sudh'
```

Out[25]:

'this is my first fun sudh'

In [26]:

```
1 def test3():
2     return 342
```

In [27]:

```
1 type(test3())
```

Out[27]:

int

In [28]:

```
1 # print always type to return nonetype
```

In [29]:

```
1 def test4():
2     return 4,3,'sudh',[1,2,3,4]
```

In [30]:

```
1 test4()
```

Out[30]:

(4, 3, 'sudh', [1, 2, 3, 4])

In [31]:

```
1 type(test4())
```

Out[31]:

tuple

In [32]:

```
1 b = test4()
```

In [33]:

```
1 b
```

Out[33]:

(4, 3, 'sudh', [1, 2, 3, 4])

In [34]:

```
1 b[1]
```

Out[34]:

3

In [35]:

```
1 b[3]
```

Out[35]:

[1, 2, 3, 4]

In [36]:

```
1 test4()
```

Out[36]:

(4, 3, 'sudh', [1, 2, 3, 4])

In [38]:

```
1 a = 1
2 b=5
3 c='sudh'
4 d=[34,33,12,53]
```

In [39]:

```
1 a,b,c,d=1, 'sudh' , [1,3,5],5
```

In [46]:

```
1 x,y,u,v ,d = test4()
```

```
-----
-
ValueError                                Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_38816\440853695.py in <module>
     1 x,y,u,v ,d = test4()
ValueError: not enough values to unpack (expected 5, got 4)
```

In [47]:

```
1 x,y,u,v = test4()
```

In [42]:

```
1 u
```

Out[42]:

'sudh'

In [43]:

```
1 v
```

Out[43]:

[1, 2, 3, 4]

In [44]:

```
1 x
```

Out[44]:

4

In [45]:

```
1 y
```

Out[45]:

3

In [49]:

```
1 def test5():
2     a = 6*7/6
3     return a
```

In [50]:

```
1 test5()
```

Out[50]:

7.0

In [51]:

```
1 len('fsds')
```

Out[51]:

4

In [52]:

```
1 l = [3,4,5,6,7,7,'sudh',[1,2,3,4,56,78]]
```

In [53]:

```
1 def test6(a):
2     n = []
3     if type(a) == list:
4         for i in a:
5             if type(i) == int:
6                 n.append(i)
7     return n
8
9
```

In [54]:

```
1 test6([3,4,5,6,7,7,'sudh',[1,2,3,4,56,78]])
```

Out[54]:

[3, 4, 5, 6, 7, 7]

In [56]:

```
1 type(test6([3,4,5,6,7,7,'sudh',[1,2,3,4,56,78])))
```

Out[56]:

list

In [83]:

```
1 def test7(a):
2     if type(a) == dict:
3         for i in a.keys():
4             print(i)
5
```

In [84]:

```
1 test7({"a" : 'raksha' , 'b' : 'raj', "c" : 'sudh'})
```

a
b
c

In [85]:

```
1 type(test7({"a" : 'raksha' , 'b' : 'raj', "c" : 'sudh'}))
```

a
b
c

Out[85]:

NoneType

In [86]:

```
1 def test7 (c) :
2     if type(c) == dict:
3         return c.keys()
4     else:
5         return ('u have not passed a dict')
```

In [87]:

```
1 test7({"a" : 'raksha' , 'b' : 'raj'})
```

Out[87]:

dict_keys(['a', 'b'])

In [88]:

```
1 s = {"a" : 'raksha' , 'b' : 'raj'}
2 for i in s.keys():
3     print(i)
```

a
b

In [90]:

```
1 def test8(l1,l2):
2     if type(l1) == list and type(l2) == list:
3         return l1+l2
```

In [92]:

```
1 test8([1,2,3], [2,3,4])
```

Out[92]:

[1, 2, 3, 2, 3, 4]

In [105]:

```
1 def test8(l1,l2):
2     if type(l1) == list and type(l2) == list:
3         return l1.extend(l2)
4     else:
5         return 'either of ur data is not a list'
```

In [106]:

```
1 test8([1,2,3], [2,3,4])
```

In [107]:

```
1 n = test8([1,2,3], [2,3,4])
```

In [111]:

```
1 a = [2,3,4,5]
2 b = [4,5,6,7,8666]
3
4 test8(a,b)
5 a
```

Out[111]:

[2, 3, 4, 5, 4, 5, 6, 7, 8666]

In [112]:

```
1 def test9(l1,l2):
2     if type(l1) == list and type(l2) == list:
3         l1.extend(l2)
4         return l1
5     else:
6         return 'either of ur data is not a list'
```

In [113]:

```
1 test9([1,2,3], [2,3,4])
```

Out[113]:

```
[1, 2, 3, 2, 3, 4]
```

In [114]:

```
1 n = 5
2 for i in range(0,n):
3     for j in range(0,i+1):
4         print("*", end = ' ')
5     print("\r")
```

```
*
**
***
****
*****
```

In [127]:

```
1 def test10(n):
2     """ this is a function which will help u to create a triangle with any no of row
3     for i in range(0,n):
4         for j in range(0,i+1):
5             print("*", end = ' ')
6         print("\r")
```

In [128]:

```
1 test10(5)
```

```
*
**
***
****
*****
```

In [132]:

```
1 help(test10)
```

Help on function test10 in module __main__:

```
test10(n)
    this is a function which will help u to create a triangle with any no
of rows
```

In []:

```
1
```

In [2]:

```
1 # 1 . you have to write a func which will take string and return a len of it without
2 def fun1(s):
3     '''This function returns the length of given string'''
4     if type(s) == str:
5         count = 0
6         for i in s:
7             count = count+1
8         return count
9 fun1('Rakshanda')
```

Out[2]:

9

In [15]:

```
1 # 2. write a func which will be able to print index of all the primitive element whi
2 def fun2(l):
3     '''This function will return the index of all the primitive element you have'''
4     if type(l) == list or type(l) == tuple or type(l) == str:
5         for i in range(len(l)):
6             print(f"Index of {l[i]} is {i} ")
7
8 #fun2('Testing my string')
9 #fun2((3,2,4,56,7))
10 fun2([1,2,3,4,56])
```

Index of 1 is 0
Index of 2 is 1
Index of 3 is 2
Index of 4 is 3
Index of 56 is 4

In [23]:

```
1 # 3. write a func which will take a nested dict and give me output as a list of all
2 # Level nesting it should be work.
3 def fun3(dg1):
4     '''This function will take input as dictionary and gives out as a list of all th
5     even in case of any number of levels of nesting it will work.'''
6     lst=[]
7     fun3b(dg1,lst)
8     return lst
9
10 def fun3a(dg2):
11     '''This function returns the values of the given dictionary.'''
12     return dg2.values()
13
14
15 def fun3b(dg3,lst):
16     '''This function inwraps the nested dictionary and returns list of values'''
17     if type(dg3) == dict:
18         a = fun3a(dg3)
19         for i in a:
20             if type(i) == dict:
21                 b = fun3b(i,lst)
22             else:
23                 lst.append(i)
24     return lst
25
26
27 fun3({'a':4, 'b':6, "c":{'d':7, 'e':8, 'f':{'g':5, 'h':8, 'i':{'j':4, 'k': 9}}}})
```

Out[23]:

```
[4, 6, 7, 8, 5, 8, 4, 9]
```

In [1]:

```
1 # 3. write a func which will take a nested dict and give me output as a list of all
2 # Level nesting it should be work.
3
4 value_list = []
5 def values(d):
6     for i in d:
7         if type(i) == dict:
8             values(d[i])
9         else:
10            value_list.append(d[i])
11
12
13 d = {'a':4, 'b':6, "c":{'d':7, 'e':8, 'f':{'g':5, 'h':8, 'i':{'j':4, 'k': 9}}}}
14 values(d)
```

Out[1]:

```
[4, 6, {"d": 7, 'e': 8, 'f': {"g": 5, 'h': 8, 'i': {"j': 4, 'k': 9}}}]
```

In [27]:

```
1 #4. write a fun which will take another func as an input and return me an output
2 def fun4a(n):
3     '''This function generates the n range of number'''
4     return list(range(n))
5
6 def fun4(fun):
7     '''This function takes another function as input and gives the list of numbers a
8     if type(fun) == list:
9         l=[]
10        for i in fun:
11            l.append(i+2)
12        return l
13 fun4(fun4a(5))
```

Out[27]:

```
[2, 3, 4, 5, 6]
```

In [30]:

```
1 # 5. write a function which will take multiple List as a input and give me concatnat
2 def fun5(*args):
3     '''This function takes n number of lists as input and gives the concatenation of
4     l = []
5     for i in args:
6         if type(i) == list:
7             l = l+i
8     return l
9
10
11 fun5([1,2,3,4],['a','b','c','d'],['sudh',23,45])
```

Out[30]:

```
[1, 2, 3, 4, 'a', 'b', 'c', 'd', 'sudh', 23, 45]
```

In [35]:

```
1 #6. write a function which will be able to take a list as an input return an index o
2 # index function but even if we have repetitive element it should return index
3 def fun6(lst1):
4     '''This function takes list as input and return an index of each element like a
5     index fuction but even if we have repetitive element if should return index'''
6     l = []
7     if type(lst1) == list:
8         for i in range(len(lst1)):
9             l.append(f"Index of {lst1[i]} is {i}")
10    return l
11 fun6([1,2,3,5,1,2])
12
```

Out[35]:

```
['Index of 1 is 0',
 'Index of 2 is 1',
 'Index of 3 is 2',
 'Index of 5 is 3',
 'Index of 1 is 4',
 'Index of 2 is 5']
```

In [39]:

```
1 # 7.write a function which will should return list of all the file name from a dire
2 def fun7(path1):
3     '''This function takes the directory path string as input and returns all the fi
4     from given directory'''
5     import os
6     return os.listdir(path1)
7 #fun7('D:\\Python_Basics')
8 fun7(os.getcwd())
```

Out[39]:

```
'.ipynb_checkpoints',
'Dict,set,tuple.ipynb',
'Dict_while_for_problems.ipynb',
'forelse,while,tupleoperation.ipynb',
'Functions_Part1.ipynb',
'Generator_file_iteratior_yeild.ipynb',
'google.txt',
'Ifelse,forloop.ipynb',
'kwargs,args,iterable,listComprehension.ipynb',
'Python_Basic_1.ipynb',
'String,Indexing,List.ipynb',
'test1.txt',
'Untitled.ipynb',
'Untitled1.ipynb',
'Untitled2.ipynb']
```

In [40]:

```
1 # 8. write a func which will be able to show your system configuration
2 def fun8():
3     '''This function will be able to show your system configuration'''
4     import platform as pl
5     return pl.uname()
6 fun8()
```

Out[40]:

```
uname_result(system='Windows', node='DESKTOP-HGGE0RL', release='10', version='10.0.22621', machine='AMD64')
```

In [42]:

```
1 #9. write a func which will be able to show date and time
2 def fun9():
3     '''This function will be able to show date and time'''
4     import subprocess as sb
5     from subprocess import time
6     a= time.localtime()
7     Date = str(a[0])+'-'+str(a[1])+'-' +str(a[2])
8     Time = str(a[3])+'-'+str(a[4])+'-' +str(a[5])
9
10    return f"The Date is {Date} and the Time is {Time}"
11 fun9()
```

Out[42]:

```
'The Date is 2023-4-29 and the Time is 12-24-19'
```

In []:

```
1 #10 . write a fun which will be able to read a image file and show it to you.
2 def fun10(filename):
3     '''This function will be able to read a image file and show it to you'''
4     import PIL
5     from PIL import Image
6     img = Image.open(filename, 'r')
7     return display(img)
8 file_name = input('Please enter the png file name along with .png extention: ')
9 fun10(file_name)
10 #Life_image.png
```

In [51]:

```
1 pip install opencv-python
```

Collecting opencv-python
Note: you may need to restart the kernel to use updated packages.

```
  Downloading opencv_python-4.7.0.72-cp37-abi3-win_amd64.whl (38.2 MB)
Requirement already satisfied: numpy>=1.17.0 in c:\users\rakshanda\anaconda3\lib\site-packages (from opencv-python) (1.22.4+vanilla)
```

```
Installing collected packages: opencv-python
Successfully installed opencv-python-4.7.0.72
```

In []:

```
1 import cv2
2 img = cv2.imread('win_image1.jpg', 1)
3 print(img)
4 cv2.imshow('image', img)
5 cv2.waitKey(0)
6 cv2.destroyAllWindows()
```

In []:

```
1
```

In [1]:

```
1 l = [1,2,3,4]
```

In [2]:

```
1 next(l)
```

```
-----
-
TypeError                                 Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_29880/2016234150.py in <module>
----> 1 next(l)
```

TypeError: 'list' object is not an iterator

In [3]:

```
1 l = iter(l)
```

In [4]:

```
1 next(l)
```

Out[4]:

1

In [5]:

```
1 next(l)
```

Out[5]:

2

In [6]:

```
1 next(l)
```

Out[6]:

3

In [8]:

```
1 next(1)
```

```
-  
StopIteration                                Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_29880/2016234150.py in <module>  
----> 1 next(1)
```

StopIteration:

In [9]:

```
1 a = 56
```

In [10]:

```
1 next(a)
```

```
-  
TypeError                                 Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_29880/1242322984.py in <module>  
----> 1 next(a)
```

TypeError: 'int' object is not an iterator

In [11]:

```
1 a = iter(a)
```

```
-  
TypeError                                 Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_29880/767013278.py in <module>  
----> 1 a = iter(a)
```

TypeError: 'int' object is not iterable

In [12]:

```
1 t = (5,6,7,8,8)
```

In [13]:

```
1 next(t)
```

```
-  
TypeError Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_29880/2139502855.py in <module>  
----> 1 next(t)
```

TypeError: 'tuple' object is not an iterator

In [14]:

```
1 t = iter(t)
```

In [15]:

```
1 next(t)
```

Out[15]:

5

In [16]:

```
1 r = range(6)
```

In [17]:

```
1 next(r)
```

```
-  
TypeError Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_29880/4134984757.py in <module>  
----> 1 next(r)
```

TypeError: 'range' object is not an iterator

In [18]:

```
1 r = iter(r)
```

In [19]:

```
1 next(r)
```

Out[19]:

0

In [20]:

```
1 next(r)
```

Out[20]:

1

In [21]:

```
1 range(45)
```

Out[21]:

range(0, 45)

In [22]:

```
1 range(0,45,3)
```

Out[22]:

range(0, 45, 3)

In [23]:

```
1 list(range(0,45,3))
```

Out[23]:

[0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42]

In [26]:

```
1 def gencube(n):
2     for i in range(n):
3         return i**3
```

In [27]:

```
1 gencube(6)
```

Out[27]:

0

In [29]:

```
1 def gencube(n):
2     for i in range(n):
3         print(i**3)
```

In [30]:

```
1 gencube(5)
```

```
0  
1  
8  
27  
64
```

In [34]:

```
1 def gencube(n):  
2     l1 = []  
3     for i in range(n):  
4         l1.append(i**3)  
5     return l1
```

In [35]:

```
1 gencube(5)
```

Out[35]:

```
[0, 1, 8, 27, 64]
```

In [37]:

```
1 for i in range(5):  
2     print(i)
```

```
0  
1  
2  
3  
4
```

In [39]:

```
1 def gencube(n):  
2     for i in range(n):  
3         yield i**3  
4
```

In [40]:

```
1 gencube(5)
```

Out[40]:

```
<generator object gencube at 0x00000233F864A510>
```

In [41]:

```
1 tuple(gencube(5))
```

Out[41]:

```
(0, 1, 8, 27, 64)
```

In [42]:

```
1 for i in gencube(5):  
2     print(i)
```

```
0  
1  
8  
27  
64
```

In [55]:

```
1 def fib(n):  
2     a = 1  
3     b = 1  
4     for i in range(n):  
5         yield a,i  
6         a,b = b ,a+b
```

In [56]:

```
1 fib(9)
```

Out[56]:

```
<generator object fib at 0x00000233F864AE40>
```

In [58]:

```
1 tuple(fib(10))
```

Out[58]:

```
((1, 0),  
(1, 1),  
(2, 2),  
(3, 3),  
(5, 4),  
(8, 5),  
(13, 6),  
(21, 7),  
(34, 8),  
(55, 9))
```

In [54]:

```
1 for i in fib(10):  
2     print(i)
```

```
(1, 0)  
(1, 1)  
(2, 2)  
(3, 3)  
(5, 4)  
(8, 5)  
(13, 6)  
(21, 7)  
(34, 8)  
(55, 9)
```

In [62]:

```
1 def fib1(n):  
2     l1=[]  
3     a =1  
4     b = 1  
5     for i in range(n):  
6         l1.append(a)  
7         a ,b = b ,a+b  
8     return l1
```

In [63]:

```
1 fib1(10)
```

Out[63]:

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

In [68]:

```
1 def fib1(n):  
2     l1=[]  
3     a =1  
4     b = 1  
5     for i in range(n):  
6         l1.append(a)  
7         a = b  
8         b= a+b  
9     return l1
```

In [69]:

```
1 fib1(10)
```

Out[69]:

```
[1, 1, 2, 4, 8, 16, 32, 64, 128, 256]
```

In [70]:

```
1 print('something in console')
```

something in console

In [71]:

```
1 # %ls
2 # pwd
3 f = open("test.txt", "w")
```

In [79]:

```
1 f.write("this is my first file operation ")
```

Out[79]:

32

In [81]:

```
1 f.close()
```

In [82]:

```
1 %%writefile test1.txt
2 this is a data i would like to store
3
```

Writing test1.txt

In [84]:

```
1 f = open('test1.txt')
```

In [85]:

```
1 f.read()
```

Out[85]:

'this is a data i would like to store\n'

In [86]:

```
1 f.write('dfas')
```

```
-  
UnsupportedOperation Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_29880/61034640.py in <module>  
e>  
----> 1 f.write('dfas')  
  
UnsupportedOperation: not writable
```

In [88]:

```
1 f.read() # cursor is it end
```

Out[88]:

''

In [89]:

```
1 # to reset the cursor  
2 f.seek(1)
```

Out[89]:

1

In [90]:

```
1 f.read()
```

Out[90]:

'his is a data i would like to store\n'

In [91]:

```
1 f.seek(0)
```

Out[91]:

0

In [92]:

```
1 f.read()
```

Out[92]:

'this is a data i would like to store\n'

In [95]:

```
1 # cursor position  
2 f.tell()
```

Out[95]:

38

In [96]:

```
1 f.seek(0)
```

Out[96]:

0

In [97]:

```
1 f.tell()
```

Out[97]:

0

In [107]:

```
1 f = open("test1.txt", "r+")
```

In [108]:

```
1 f.read()
```

Out[108]:

```
'this is a data i would like to store.\nfsdfa\nfsafas\nfasfasf\nfsafas\n  
sdfasdfas\nnsffasdffsdafsfaf\nthis is a data i would like to stor  
e.\n'
```

In [109]:

```
1 f.readline()
```

Out[109]:

'

In [110]:

```
1 f.tell()
```

Out[110]:

147

In [111]:

```
1 f.seek(0)
```

Out[111]:

0

In [114]:

```
1 # it will read data line by line
2 f.readline()
```

Out[114]:

'fsafas\n'

In [115]:

```
1 f = open('google.txt' , 'w')
```

In [119]:

```
1 f.write("""Google began as a search project by Larry Page and Sergey Brin when they
2
3 The search engine algorithm they developed was unique because it ranked pages not on
4
5 Page and Brin determined that links to a page were a sign of its online authority, a
```



Out[119]:

488

In [121]:

```
1 f.close()
```

In [125]:

```
1 f = open('google.txt' , "r+")
```

In [126]:

```
1 f.read()
```

Out[126]:

"Google began as a search project by Larry Page and Sergey Brin when they were Ph.D. students at Stanford University.\n\nThe search engine algorithm they developed was unique because it ranked pages not only based on their content, but on how many other webpages linked back to them.\n\nPage and Brin determined that links to a page were a sign of its online authority, and Google's algorithm therefore returned more useful results, helping propel Google to become the most-used search engine."

In [127]:

```
1 f.tell()
```

Out[127]:

492

In [128]:

```
1 f.seek(0)
```

Out[128]:

0

In [129]:

```
1 f.readline()
```

Out[129]:

'Google began as a search project by Larry Page and Sergey Brin when they were Ph.D. students at Stanford University.\n'

In [130]:

```
1 f.close()
```

In [133]:

```
1 f = open('google.txt', "r+")
2 for i in f:
3     print(i , end = " ")
```

Google began as a search project by Larry Page and Sergey Brin when they were Ph.D. students at Stanford University.

The search engine algorithm they developed was unique because it ranked pages not only based on their content, but on how many other webpages linked back to them.

Page and Brin determined that links to a page were a sign of its online authority, and Google's algorithm therefore returned more useful results, helping propel Google to become the most-used search engine.

In [134]:

```
1 f.write('writing new line')
```

Out[134]:

16

In [135]:

```
1 f.close()
```

In [152]:

```
1 f = open('google.txt' , "r+")
```

In [153]:

```
1 f.seek(0)
```

Out[153]:

0

In [154]:

```
1 f.write('adding new line.')
```

Out[154]:

16

In [155]:

```
1 f.close()
```

In [156]:

```
1 f = open('google.txt' , "r+")
```

In [157]:

```
1 f.read()
```

Out[157]:

"adding new line.a search project by Larry Page and Sergey Brin when they were Ph.D. students at Stanford University.\n\nThe search engine algorithm they developed was unique because it ranked pages not only based on their content, but on how many other webpages linked back to them.\n\nPage and Brin determined that links to a page were a sign of its online authority, and Google's algorithm therefore returned more useful results, helping propel Google to become the most-used search engine.writing new lineadding new line,adding new line."

In [172]:

```
1 f.seek(0)
```

Out[172]:

0

In [162]:

```
1 f = open('google.txt', "r+")
```

In [186]:

```
1 f.seek(0)
```

Out[186]:

0

In [182]:

```
1 f.readlines()
```

Out[182]:

```
['adding new line.a search project by Larry Page and Sergey Brin when they  
were Ph.D. students at Stanford University.\n',  
'\n',  
'The search engine algorithm they developed was unique because it ranked  
pages not only based on their content, but on how many other webpages link  
ed back to them.\n',  
'\n',  
"Page and Brin determined that links to a page were a sign of its online  
authority, and Google's algorithm therefore returned more useful results,  
helping propel Google to become the most-used search engine.writing new li  
neadding new line,adding new line."]
```

In [184]:

```
1 len(f.readlines())
```

Out[184]:

5

In [187]:

```
1 l = f.readlines()
```

In [205]:

```
1 l[4].split()
```

Out[205]:

```
['Page',
 'and',
 'Brin',
 'determined',
 'that',
 'links',
 'to',
 'a',
 'page',
 'were',
 'a',
 'sign',
 'of',
 'its',
 'online',
 'authority',
 'and',
 "Google's",
 'algorithm',
 'therefore',
 'returned',
 'more',
 'useful',
 'results',
 'helping',
 'propel',
 'Google',
 'to',
 'become',
 'the',
 'most-used',
 'search',
 'engine.writing',
 'new',
 'lineadding',
 'new',
 'line,adding',
 'new',
 'line.']}
```

In [191]:

```
1 l1 = []
2 for i in l[0].split():
3     l1.append(i[0])
```

In [192]:

```
1 l1
```

Out[192]:

```
['a',
 'n',
 'l',
 's',
 'p',
 'b',
 'L',
 'P',
 'a',
 'S',
 'B',
 'w',
 't',
 'w',
 'P',
 's',
 'a',
 'S',
 'U']
```

In [217]:

```
1
2 for i in l[4].split():
3     if i == "Google":
4         i.replace("Google", 'rakshanda')
```

In [218]:

```
1 l[4]
```

Out[218]:

"Page and Brin determined that links to a page were a sign of its online authority, and Google's algorithm therefore returned more useful results, helping propel Google to become the most-used search engine.writing new line
eadding new line,adding new line."

In [220]:

```
1 import re
2 def replace_word(file_name, old_word, new_word):
3     with open(file_name, 'r+') as f:
4         text = f.read()
5         f.seek(0)
6         f.write(re.sub(old_word, new_word, text))
7         f.truncate()
8 replace_word('google.txt', 'Google', 'rakshanda')
```

In [221]:

```
1 f = open('google.txt', "r+")
```

In [222]:

```
1 f.name
```

Out[222]:

```
'google.txt'
```

In [223]:

```
1 l = ["this is my line1" , "this is my line 2 " , "this is my line3" , "this is my li
```

In [224]:

```
1 f.seek(0)
```

Out[224]:

```
0
```

In [225]:

```
1 f.write('I can write line at current cursor')
```

Out[225]:

```
34
```

In [226]:

```
1 f.seek(0)
```

Out[226]:

```
0
```

In [227]:

```
1 f.writelines(l)
```

In [228]:

```
1 f.close()
```

In [231]:

```
1 f = open('test1.txt', "r+")
```

In [232]:

```
1 f.name
```

Out[232]:

```
'test1.txt'
```

In [234]:

```
1 f = open('google.txt', "r+")
```

In [235]:

```
1 f.fileno()
```

Out[235]:

```
7
```

In [236]:

```
1 ls
```

```
Volume in drive D is New Volume
Volume Serial Number is 1E8A-D698
```

```
Directory of D:\Python_Basics
```

```
29-04-2023  09:58    <DIR>          .
28-04-2023  19:42    <DIR>          .ipynb_checkpoints
27-04-2023  15:17          40,921 Dict,set,tuple.ipynb
28-04-2023  14:07          29,802 Dict_while_for_problems.ipynb
27-04-2023  20:56          23,791 forelse,while,tupleoperation.ipynb
28-04-2023  16:31          18,956 Functions_Part1.ipynb
29-04-2023  09:58          39,376 Generator_file_iteratior_yeild.ipynb
28-04-2023  21:36          546 google.txt
27-04-2023  17:56          13,584 Ifelse,forloop.ipynb
28-04-2023  19:36          35,749 kwargs,args,iterable,listComprehension.ipynb
27-04-2023  11:23          16,802 Python_Basic_1.ipynb
27-04-2023  15:23          57,409 String,Indexing,List.ipynb
28-04-2023  20:38          32 test.txt
28-04-2023  20:51          147 test1.txt
28-04-2023  19:33          35,749 Untitled.ipynb
29-04-2023  09:56          39,054 Untitled1.ipynb
               14 File(s)      351,918 bytes
               2 Dir(s)   326,900,137,984 bytes free
```

In [237]:

```
1 import os
```

In [238]:

```
1 os.remove('test.txt')
```

In [239]:

```
1 ls
```

```
Volume in drive D is New Volume
Volume Serial Number is 1E8A-D698
```

```
Directory of D:\Python_Basics
```

```
29-04-2023  09:59    <DIR>          .
28-04-2023  19:42    <DIR>          .ipynb_checkpoints
27-04-2023  15:17          40,921 Dict,set,tuple.ipynb
28-04-2023  14:07          29,802 Dict_while_for_problems.ipynb
27-04-2023  20:56          23,791 forelse,while,tupleoperation.ipynb
28-04-2023  16:31          18,956 Functions_Part1.ipynb
29-04-2023  09:58          39,376 Generator_file_iteratior_yeild.ipynb
28-04-2023  21:36          546 google.txt
27-04-2023  17:56          13,584 Ifelse,forloop.ipynb
28-04-2023  19:36          35,749 kwargs,args,iterable,listComprehension.ipynb
27-04-2023  11:23          16,802 Python_Basic_1.ipynb
27-04-2023  15:23          57,409 String,Indexing,List.ipynb
28-04-2023  20:51          147 test1.txt
28-04-2023  19:33          35,749 Untitled.ipynb
29-04-2023  09:56          39,054 Untitled1.ipynb
               13 File(s)      351,886 bytes
               2 Dir(s)   326,900,137,984 bytes free
```

In [240]:

```
1 os.getcwd()
```

Out[240]:

```
'D:\\Python_Basics'
```

In [241]:

```
1 pwd()
```

Out[241]:

```
'D:\\Python_Basics'
```

In [246]:

```
1 os.listdir()
2 os.system("ls *.txt")
```

```
File "C:\\Users\\RAKSHA~1\\AppData\\Local\\Temp\\ipykernel_29880/1466853418.py", line 2
    os.system("ls *.txt")
^
IndentationError: unexpected indent
```

In [247]:

```
1 for i in os.listdir():
2     if i.endswith(".txt"):
3         print(i)
```

google.txt
test1.txt

In [250]:

```
1 l = ['raksha', 'aksha' , 'sama']
2 for i in l:
3     if i.endswith('a'):
4         print(i)
```

raksha
aksha
sama

In [244]:

```
1 s = 'raksha'
2 s.endswith('a')
```

Out[244]:

True

In [249]:

```
1 import os
2 os.system("ls *.txt")
```

Out[249]:

1

In [251]:

```
1 os.getcwd()
```

Out[251]:

'D:\\Python_Basics'

In []:

```
1
```

In [1]:

```
1 a = 10
```

In [5]:

```
1 if a < 15 :  
2     print('fsdfsfs')
```

fsdfsfs

In [6]:

```
1 a<15
```

Out[6]:

True

In [7]:

```
1 if a < 15 :  
2     pass
```

In [8]:

```
1 if a < 15 :  
2     print("my name is raju")
```

my name is raju

In [15]:

```
1 if 24 < 15 :  
2     print('my name is sudh')
```

In [14]:

```
1 24 < 15
```

Out[14]:

False

In [16]:

```
1 if 10 <3 :  
2     print('10 is lesser than 3')  
3 else :  
4     print('it statement is wrong')
```

it statement is wrong

In [18]:

```
1 if 2 <3 :  
2     print('2 is lesser than 3')  
3 else :  
4     print('it statement is wrong')
```

10 is lesser than 3

In [24]:

```
1 income = int(input())  
2 if income < 50 :  
3     print('I will be able to buy phone')  
4 elif income < 70 :  
5     print("I will be able to buy car")  
6 elif income < 90 :  
7     print ('I will be able to rent a house')  
8 else :  
9     print('I wont be able to buy anything')
```

100

I wont be able to buy anything

In [27]:

```
1 total_price = int(input())  
2 if total_price > 20000:  
3     discount = total_price * 0.20  
4     print('Discount will be', discount)  
5 elif total_price <=7000:  
6     discount = total_price *.05  
7     print('discount will be', discount)  
8 else:  
9     print('wont be able to give discount')
```

4532

discount will be 226.6000000000002

In [28]:

```
1 'sudh' == 'sudh'
```

Out[28]:

True

In [29]:

```
1 a = 9  
2 if a==9 :  
3     print('valid')
```

valid

```
1 | nested if
```

In [30]:

```
1 | l = [1,2,3,4,5,6]
2 | for i in l:
3 |     print(i)
```

```
1
2
3
4
5
6
```

In [31]:

```
1 | s = 'sudh'
2 | for i in s:
3 |     print(i)
```

```
s
u
d
h
```

In [32]:

```
1 | t = (1,2,3,4,5,6,6)
```

In [34]:

```
1 | for i in t:
2 |     print(i)
```

```
1
2
3
4
5
6
6
```

In [39]:

```
1 | l = [1,4,5.6,4+7j, 'sudh', True]
2 | for i in l:
3 |     print(type(i), 'type of', i)
```

```
<class 'int'> type of 1
<class 'int'> type of 4
<class 'float'> type of 5.6
<class 'complex'> type of (4+7j)
<class 'str'> type of sudh
<class 'bool'> type of True
```

In [40]:

```
1 l = [1,4,5.6]
2 for i in l:
3     print(i+2)
```

```
3
6
7.6
```

In [50]:

```
1 l = [1,4,5.6]
2 l1=[]
3 for i in l:
4     l1.append(i+2);
5 print(l1)
```

```
[3, 6, 7.6]
```

In [49]:

```
1 print(l1)
```

```
[3, 6, 7.6]
```

In [51]:

```
1 l = [1,4,5.6]
2 l1=[]
3 for i in l:
4     i=i+2
5     l1.append(i);
6 print(l1)
```

```
[3, 6, 7.6]
```

In [59]:

```
1 l = [2,45,78,12,'sudh',6+7j,[56,67,78]]
2 l2=[]
3 for i in l:
4     if type(i) == int :
5         l2.append(i);
6 print(l2)
```

```
[2, 45, 78, 12]
```

In [69]:

```
1 l = [2,45,78,12,'sudh',6+7j,[56,67,78]]
2 l2=[]
3 for i in l:
4     if type(i) == list :
5         for j in i:
6             if type(j) ==int:
7
8                 l2.append(j);
9 print(l2)
```

[56, 67, 78]

In [61]:

```
1 l = [2,45,78,12,'sudh',6+7j,[56,67,78]]
2 l2=[]
3 for i in l:
4     if type(i) == complex :
5         l2.append(i);
6 print(l2)
```

[(6+7j)]

In [87]:

```
1 l = [2,45,78,12,'sudh',6+7j,[56,67,78]]
2 l2=[]
3 for i in l:
4     l2.append(i)
5     print(l2.index(i),'index value is', i)
6
7
```

0 index value is 2
1 index value is 45
2 index value is 78
3 index value is 12
4 index value is sudh
5 index value is (6+7j)
6 index value is [56, 67, 78]

In [105]:

```
1 l = [2,45,78,12,'sudh','kumar',6+7j,[56,67,78]]
2 l2=[]
3 for i in l:
4     if type(i) == str :
5         for j in i:
6             l2.append(j)
7 print(l2)
8
```

['s', 'u', 'd', 'h', 'k', 'u', 'm', 'a', 'r']

In [94]:

```
1 l = [2,45,78,12,'sudh',6+7j,[56,67,78]]
2 l2=[]
3 for i in l:
4     if type(i)==int:
5         i=i**2
6         l2.append(i)
7 print(l2)
```

[4, 2025, 6084, 144]

In [97]:

```
1 l = [2,45,78,12,'sudh',6+7j,[56,67,78]]
2 for i in range(len(l)):
3     print('index', i , 'for an element' , l[i])
```

index 0 for an element 2
index 1 for an element 45
index 2 for an element 78
index 3 for an element 12
index 4 for an element sudh
index 5 for an element (6+7j)
index 6 for an element [56, 67, 78]

In [99]:

```
1 for i in enumerate(l):
2     print(i)
```

(0, 2)
(1, 45)
(2, 78)
(3, 12)
(4, 'sudh')
(5, (6+7j))
(6, [56, 67, 78])

In [101]:

```
1 for i,j in enumerate(l):
2     print(i,j)
```

0 2
1 45
2 78
3 12
4 sudh
5 (6+7j)
6 [56, 67, 78]

In [106]:

```
1 l = [2,45,78,12,'sudh','kumar',6+7j,[56,67,78]]  
2 for i in l:  
3     if type(i)== str:  
4         l1=[]  
5         for j in i:  
6             l1.append(j)  
7         print(l1)
```

```
['s', 'u', 'd', 'h']  
['k', 'u', 'm', 'a', 'r']
```

In [108]:

```
1 l = [2,45,78,12,'sudh','kumar',6+7j,[56,67,78]]  
2 l2=[]  
3 for i in l:  
4     if type(i)==int :  
5         l2.append(i**2)  
6 print(l2)
```

```
[4, 2025, 6084, 144]
```

In []:

```
1
```

In [47]:

```
1 class dict_parsing():
2
3     def __init__(self,a):
4         self.a = a
5
6     def get_keys(self):
7         if self.verify_dict() :
8             return list(self.a.keys())
9
10    def get_values(self):
11        if self.verify_dict() :
12            return list(self.a.values())
13
14    def verify_dict(self):
15        if type(self.a) !=dict:
16            raise Exception('input is not dictionary' , self.a)
17
18    def insertion(self,**kwargs):
19        for k ,v in kwargs.items():
20            self.a[k] = v
21
22    return self.a
```

In [48]:

```
1 d = dict_parsing({'a': 5, 'b': 6, 'c': 7, 'h': '67'})
```

In [50]:

```
1 d.insertion(name = 'sudh')
```

Out[50]:

```
{'a': 5, 'b': 6, 'c': 7, 'h': '67', 'name': 'sudh'}
```

In [52]:

```
1 d.insertion(t = 78)
```

Out[52]:

```
{'a': 5, 'b': 6, 'c': 7, 'h': '67', 'name': 'sudh', 't': 78}
```

In [54]:

```
1 d.insertion(h = 56, mail_id = 'raksha@gmail.com' , mob_No = 4532)
```

Out[54]:

```
{'a': 5,
'b': 6,
'c': 7,
'h': 56,
'name': 'sudh',
't': 78,
'mail_id': 'raksha@gmail.com',
'mob_No': 4532}
```

In [32]:

```
1 d.verify_dict()
```

In [29]:

```
1 d.get_keys()
```

In [33]:

```
1 d.get_values()
```

In [16]:

```
1 class dict_parsing():
2
3     def __init__(self,a):
4         if type(a)!= dict:
5             raise Exception('input is not dictionary' , self.a)
6         self.a = a
7
8
9
10    # def verify_dict(self):
11    #     if type(self.a) !=dict:
12    #         raise Exception('input is not dictionary' , self.a)
13
14
15    def get_keys(self):
16        for i in self.a.keys():
17            print(i)
18
19
20    def get_values(self):
21        if self.verify_dict() :
22            return list(self.a.values())
23
24
25    def insertion(self,**kwargs):
26
27        for k ,v in kwargs.items():
28            self.a[k] = v
29        return self.a
```

In [17]:

```
1 d = dict_parsing({'a': 5, 'b': 6, 'c': 7, 'h': '67'})
```

In [18]:

```
1 d.insertion()
```

Out[18]:

```
{'a': 5, 'b': 6, 'c': 7, 'h': '67'}
```

In [21]:

```
1 d.get_keys()
```

```
a  
b  
c  
h
```

In [2]:

```
1 class dict_fun():
2     def __init__(self,a):
3         self.a = a
4
5     def get_keys(self):
6         if self.verify_input():
7             return list(self.a.keys())
8
9     def get_values(self):
10        if self.verify_input():
11            return self.a.values()
12
13    def verify_input(self):
14        if type(self.a) !=dict:
15            raise Exception('input is not dictionary' , b)
```

In [3]:

```
1 d = dict_fun({'a': 5, 'b': 6, 'c': 7, 'h': '67'})
```

In [4]:

```
1 d.get_keys()
```

In [21]:

```
1 import mydict
```

In [22]:

```
1 mydict.Dict_function
```

Out[22]:

mydict.Dict_function

In [27]:

```
1 a = mydict.Dict_function({'a': 5, 'b': 6, 'c': 7, 'h': '67'})
```

In [28]:

```
1 a.get_keys()
```

Out[28]:

['a', 'b', 'c', 'h']

In [29]:

```
1 a.get_values()
```

Out[29]:

[5, 6, 7, '67']

In [30]:

```
1 from mydict import Dict_function
```

In [31]:

```
1 a = Dict_function({'a': 5, 'b': 6, 'c': 7, 'h': '67'})
```

In [32]:

```
1 a.get_values()
```

Out[32]:

```
[5, 6, 7, '67']
```

In []:

```
1
```

In [1]:

```
1 class xyz :  
2     def __init__(self, a,b,c):  
3         self.a = a  
4         self.b = b  
5         self.c = c  
6  
7     def test(self):  
8         print('this is my first print method of xyz class')  
9  
10    def test1(self):  
11        print('this is my test2 method of xyz class ')  
12  
13    def test2(self):  
14        print('this is my test3 method of xyz class')  
15  
16
```

In [6]:

```
1 p = xyz(1,2,3)
```

In [7]:

```
1 p.test1()
```

this is my test2 method of xyz class

In [5]:

```
1 class xyz1(xyz):  
2     pass
```

In [9]:

```
1 q = xyz1(1,2,3)
```

In [15]:

```
1 q.test1()
```

this is my test2 method of xyz class

In [16]:

```
1 q = xyz1()
```

```
-  
TypeError Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_2300\2225742902.py in <modu  
le>  
----> 1 q = xyz1()  
  
TypeError: __init__() missing 3 required positional arguments: 'a', 'b', a  
nd 'c'
```

In [17]:

```
1 class xyz1(xyz):  
2     def test(self):  
3         print('this is the test method available in xyz1')
```

In [18]:

```
1 g = xyz1(1,2,3)
```

In [19]:

```
1 g.test()
```

```
this is the test method available in xyz1
```

In [39]:

```
1 class xyz :
2     def __init__(self, a,b,c):
3         self.a = a
4         self.b = b
5         self.c = c
6
7     def test(self):
8         print('this is my first print method of xyz class')
9
10
11 class xyz1:
12     def __init__(self, p,q,r):
13         self.p = p
14         self.q = q
15         self.r = r
16
17     def test1(self):
18         print('this is a meth from class xyz1')
19
20 class child(xyz1,xyz):
21     pass
22
23 #class child(xyz1,xyz):
24 #    pass
25
26 # variable wise it will work for first argument
27 # method wise it will work for all the methods
```

In [40]:

```
1 n = child(4,5,6)
```

In [41]:

```
1 n.test()
```

this is my first print method of xyz class

In [42]:

```
1 n.test1()
```

this is a meth from class xyz1

In [51]:

```
1 # Multiple Inheritance
2 class xyz :
3     def __init__(self, a,b,c):
4         self.a = a
5         self.b = b
6         self.c = c
7
8     def test(self):
9         print('this is my first print method of xyz class')
10
11
12 class xyz1:
13     def __init__(self, p,q,r):
14         self.p = p
15         self.q = q
16         self.r = r
17
18     def test1(self):
19         print('this is a meth from class xyz1')
20
21 class child(xyz1,xyz):
22     def __init__(self,*args, **kwargs):
23         xyz.__init__(self,*args)
24         xyz1.__init__(self,**kwargs)
25
26
```

In [53]:

```
1 n = child(4,5,6,p=5, q=6, r =7)
```

In [54]:

```
1 n.a
```

Out[54]:

4

In [55]:

```
1 n.p
```

Out[55]:

5

In [56]:

```
1 n.test()
```

this is my first print method of xyz class

In [57]:

```
1 n.test1()
```

this is a meth from class xyz1

In [65]:

```
1 # Multilevel Inheritance
2 class xyz :
3     def __init__(self, a,b,c):
4         self.a = a
5         self.b = b
6         self.c = c
7
8     def test(self):
9         print('this is my first print method of xyz class')
10
11
12 class xyz1(xyz):
13
14     def test1(self):
15         print('this is a meth from class xyz1')
16
17 class xyz2(xyz1):
18     def test2(self):
19         print('this is meth from xlass xyz2')
20
21 class xyz3(xyz2):
22     def test3(self):
23         print('this is meth from xyz3')
```

In [66]:

```
1 v = xyz2(4,5,6)
```

In [67]:

```
1 v = xyz3(5,6,7)
```

In [68]:

```
1 v.test3()
```

this is meth from xyz3

In [132]:

```
1 class File:
2     def write(self,data):
3         f = open('inherit.txt', "w")
4         f.write(data)
5         f.close()
6
7     def read(self):
8         f = open('inherit.txt', 'r')
9         f.seek(0)
10        print(f.read())
11
12 class child(File):
13     def test(self):
14         pass
```

In [133]:

```
1 x = File()
```

In [135]:

```
1 x.write('Writing the file')
```

In [136]:

```
1 x.read()
```

Writing the file

In [94]:

```
1 x1 = child()
```

In [95]:

```
1 x1.read()
```

this is my first write method

In [119]:

```
1 import logging
2 logging.basicConfig(filename ='List.log',level=logging.DEBUG,format ="%(asctime)s,%(l
3
4 class File:
5     def __init__(self,filename):
6         logging.info('Creating file Instance variable')
7         self.file = filename
8
9     def read(self):
10        logging.info('Executing read method')
11        try:
12            with open(f'{self.file}.txt' , 'r') as f:
13                data = f.read()
14                return data
15        except FileNotFoundError as e:
16            logging.error('Error occured while opening the file')
17            logging.exception(f"Error is {s}")
18
19    def write(self,data):
20        logging.info('Executing write method')
21        try:
22            with open(f'{self.file}.txt' , "w") as f:
23                data = f.write(data)
24                f.close()
25
26        except FileNotFoundError as e:
27            logging.error('Something is going wrong')
28
29 class Child(File):
30     def test(self):
31         print('this is child class')
```

In [125]:

```
1 x = Child('inherit')
```

In [126]:

```
1 x.write('writing file')
```

In [127]:

```
1 x.read()
```

Out[127]:

'writing file'

In [121]:

```
1 Obj = File('inherit')
```

In [111]:

```
1 Obj.read()
```

Out[111]:

```
'writing file'
```

In [112]:

```
1 Obj.write('writing file')
```

In [140]:

```
1 class test:
2     def __init__(self,a,b,c):
3         self.a = a
4         self.b = b
5         self.c = c
6
7 class test1(test):
8     pass
9
10 u = test(4,5,6)
```

In [141]:

```
1 v = test1(3,4,5)
```

In [142]:

```
1 v.a
```

Out[142]:

```
3
```

In [173]:

```
1 # _a = protected =
2 class test:
3     def __init__(self):
4         self._a = 4
5
6 class test1(test):
7     def __init__(self) :
8         self._a = 7
9
10 u = test()
```

In [177]:

```
1 u._a
```

Out[177]:

```
4
```

In [175]:

```
1 v = test1()
```

In [178]:

```
1 v._a
```

Out[178]:

7

In [179]:

```
1 # __a = protected = u can use inside classes and inside subclasses but outside that
2 class test:
3     def __init__(self):
4         self.__a = 4
5
6 class test1(test):
7     def __init__(self) :
8         self.__a = 7
9
10 u = test()
```

In [180]:

```
1 u.__a
```

```
-----
-
AttributeError                                     Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_2300\3701111835.py in <module>
      1 u.__a
```

AttributeError: 'test' object has no attribute '__a'

In [182]:

```
1 v =test1()
2 v.__a
```

```
-----
-
AttributeError                                     Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_2300\4219025645.py in <module>
      1 v =test1()
      2 v.__a
```

AttributeError: 'test1' object has no attribute '__a'

In [195]:

```
1 # protected : inside class and inside subclass(child classes )
2 # private : Just within the class (not child class)
3 # public : can access from any classes anywhere
4 # encapsulation : others not able access any data without user permission
5 class test:
6     def __init__(self,a,b,c):
7         self._a = a
8         self.__b = b
9         self.c = c
10
11 class test1(test):
12     pass
13
14 v = test(4,5,6)
```

In [196]:

```
1 v._a
```

Out[196]:

4

In [191]:

```
1 # If I want to access private variable then use
2 v.__b
3
```

```
-
```

AttributeError Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_2300\4134146159.py in <module>
 1 # If I want to access private variable then use
----> 2 v.__b

AttributeError: 'test' object has no attribute '__b'

In [193]:

```
1 # If I want to access private variable then use
2 v._test__b
```

Out[193]:

5

In [197]:

```
1 v.c
```

Out[197]:

6

In [199]:

```
1 u = test1(3,4,5)
```

In [200]:

```
1 u.c
```

Out[200]:

5

In [204]:

```
1 u._a
```

Out[204]:

3

In [209]:

```
1 u._test1__b
```

-
AttributeError Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_2300/4083783106.py in <module>
----> 1 u._test1__b

AttributeError: 'test1' object has no attribute '_test1__b'

In [210]:

```
1 u._test__b
```

Out[210]:

4

In []:

```
1 class bonouscalculater:  
2
```

In [1]:

```
1 def test(a,b,c,d,e):  
2     return a,b,c,d,e
```

In [2]:

```
1 test(34,56,78,3,2)
```

Out[2]:

(34, 56, 78, 3, 2)

In [3]:

```
1 def test1(*args):  
2     return args
```

In [4]:

```
1 test1(23,45,33)
```

Out[4]:

(23, 45, 33)

In [6]:

```
1 test1(45,34.3,'sudh',45, [12,3,4,5],(2,3,4), 7+6j)
```

Out[6]:

(45, 34.3, 'sudh', 45, [12, 3, 4, 5], (2, 3, 4), (7+6j))

In [7]:

```
1 def test2(*sudh):  
2     return sudh
```

In [8]:

```
1 test2('sfsaf', 332,345)
```

Out[8]:

('sfsaf', 332, 345)

In [9]:

```
1 def test3(*sudh, a):  
2     return sudh,a
```

In [12]:

```
1 test3(34,22,44,7,a=675)
```

Out[12]:

```
((34, 22, 44, 7), 675)
```

In [13]:

```
1 def test4(*args , a,b,c,d):  
2     return args, a,b,c,d
```

In [14]:

```
1 test4(34,23,44,55, a='sudh' , b= '45', c = 45.3, d = 'raksha')
```

Out[14]:

```
((34, 23, 44, 55), 'sudh', '45', 45.3, 'raksha')
```

In [15]:

```
1 def test3(a, *args):  
2     return args , a
```

In [16]:

```
1 test3('fsds' , 45,33,33)
```

Out[16]:

```
((45, 33, 33), 'fsds')
```

In [18]:

```
1 def test5(a, *sudh, b,c):  
2     return a,sudh,b,c
```

In [19]:

```
1 test5('fsds' , [23,4,5,6,2] , 45,43,34,55,2, b=6,c=8)
```

Out[19]:

```
('fsds', ([23, 4, 5, 6, 2], 45, 43, 34, 55, 2), 6, 8)
```

In [20]:

```
1 def test6(*args):  
2     for i in args:  
3         if type(i) == list:  
4             return i
```

In [21]:

```
1 test6([12,3,2,3], 'sudh', 4, 3)
```

Out[21]:

```
[12, 3, 2, 3]
```

In [22]:

```
1 test6([12,3,2,3], 'sudh', 4, 3, [2,3,4],[5,6,7])
```

Out[22]:

```
[12, 3, 2, 3]
```

In [23]:

```
1 def test6(*args):
2     for i in args:
3         if type(i) == list:
4             print(i)
```

In [24]:

```
1 test6([12,3,2,3], 'sudh', 4, 3, [2,3,4],[5,6,7])
```

```
[12, 3, 2, 3]
[2, 3, 4]
[5, 6, 7]
```

In [25]:

```
1 l = []
2 def test6(*args):
3     for i in args:
4         if type(i) == list:
5             l.append(i)
6     return l
```

In [26]:

```
1 test6([12,3,2,3], 'sudh', 4, 3, [2,3,4],[5,6,7])
```

Out[26]:

```
[[12, 3, 2, 3], [2, 3, 4], [5, 6, 7]]
```

In [27]:

```
1 d = {'a' : [2,3,4,5,6], 6: [4,56,5,33,31]}
```

In [28]:

```
1 d["a"]
```

Out[28]:

```
[2, 3, 4, 5, 6]
```

In [1]:

```
1 def test7(**kwargs):  
2     return kwargs
```

In [2]:

```
1 test7(353,53,33,4,5)
```

```
-----  
-  
TypeError                                     Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_12584\1533826139.py in <mod  
ule>  
----> 1 test7(353,53,33,4,5)
```

TypeError: test7() takes 0 positional arguments but 5 were given

In [32]:

```
1 test7(a=45,b=56)
```

Out[32]:

```
{'a': 45, 'b': 56}
```

In [34]:

```
1 def test8(**sudh):  
2     return sudh
```

In [35]:

```
1 test8(b=787,c=90,f=[3,45,2,4,4], l ='raksha')
```

Out[35]:

```
{'b': 787, 'c': 90, 'f': [3, 45, 2, 4, 4], 'l': 'raksha'}
```

In [36]:

```
1 def test9(**sudh):  
2     return sudh
```

In [38]:

```
1 test9(name = 'sudh', age=31 , phone_num= 9234562, adr='fsdasfa',mail_id = 'sudh@gmai
```

Out[38]:

```
{'name': 'sudh',
'age': 31,
'phone_num': 9234562,
'adr': 'fsdasfa',
'mail_id': 'sudh@gmail.com'}
```

In [39]:

```
1 def test10(a,**sudh):
2     return sudh,a
```

In [41]:

```
1 test10(45, a= 34,b=65,c=77,d= 88)
```

```
-----
-
TypeError                                 Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_32896/1156300966.py in <module>
----> 1 test10(45, a= 34,b=65,c=77,d= 88)

TypeError: test10() got multiple values for argument 'a'
```

In [43]:

```
1 test10(45, t= 34, b=65, c=77, d= 88)
```

Out[43]:

```
({'t': 34, 'b': 65, 'c': 77, 'd': 88}, 45)
```

In [44]:

```
1 def test11(a,**sudh,*args)
```

```
File "C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_32896/266792643.py", line 1
    def test11(a,**sudh,*args)
          ^
SyntaxError: invalid syntax
```

In [45]:

```
1 def test12(a, *args, **kwargs):
2     return a, args,kwargs
```

In [46]:

```
1 test12(2,3,4,12,34,56,77,90, g=78,h='raj')
```

Out[46]:

(2, (3, 4, 12, 34, 56, 77, 90), {'g': 78, 'h': 'raj'})

In [47]:

```
1 def test13(a,b):
2     return a*b
```

In [48]:

```
1 test13(4,5)
```

Out[48]:

20

In [57]:

```
1 a = lambda a,b : (a*b, a+b)
```

In [58]:

```
1 a(3,4)
```

Out[58]:

(12, 7)

In [59]:

```
1 a = lambda a,b : a*b
```

In [60]:

```
1 a(27,7)
```

Out[60]:

189

In [61]:

```
1 a = lambda *a : a
```

In [62]:

```
1 a(56,7,5,[4,5,6,7], 'raksha')
```

Out[62]:

(56, 7, 5, [4, 5, 6, 7], 'raksha')

In [66]:

```
1 v = lambda *i : i**2      # tuple multiplication not possible
```

In [70]:

```
1 x = lambda x : [ print(i) for i in x]
```

In [71]:

```
1 x('raksha')
```

```
r  
a  
k  
s  
h  
a
```

Out[71]:

```
[None, None, None, None, None, None]
```

In [72]:

```
1 x = lambda x : [ i for i in x]
```

In [73]:

```
1 x([4,5,6])
```

Out[73]:

```
[4, 5, 6]
```

In [74]:

```
1 [ i for i in l]
```

Out[74]:

```
[[12, 3, 2, 3], [2, 3, 4], [5, 6, 7]]
```

In [75]:

```
1 l
```

Out[75]:

```
[[12, 3, 2, 3], [2, 3, 4], [5, 6, 7]]
```

In [76]:

```
1 a = lambda **kwargs : kwargs
```

In [77]:

```
1 a(a='raksha', b = 'raj' , c ='cg')
```

Out[77]:

```
{'a': 'raksha', 'b': 'raj', 'c': 'cg'}
```

In [81]:

```
1 a = 10
2 def test16(c,d):
3     a = 5
4     return c*d
```

In [82]:

```
1 test16(a,50)
```

Out[82]:

```
500
```

In [85]:

```
1 a = 10
2 def test16(c,d):
3     c= 5
4     return c*d
```

In [87]:

```
1 test16(a,50)
```

Out[87]:

```
250
```

In [93]:

```
1 c = 10
2 def test16(c,d):
3     c= 5
4     return c*d
```

In [95]:

```
1 test16(10,50)
```

Out[95]:

```
250
```

In [96]:

```
1 c
```

Out[96]:

10

In [99]:

```
1 l = [1,2,3,4,5,4,88,99]
2 l1 = []
3 for i in l:
4     l1.append(i+2)
```

In [100]:

```
1 l1
```

Out[100]:

[3, 4, 5, 6, 7, 6, 90, 101]

In [103]:

```
1 def test17(l):
2     l1=[]
3     for i in l:
4         l1.append(i+2)
5     return l1
```

In [104]:

```
1 test17([1,2,3,4,5,6,7])
```

Out[104]:

[3, 4, 5, 6, 7, 8, 9]

In [105]:

```
1 x = lambda l : [ i+2 for i in l]
```

In [106]:

```
1 x(l1)
```

Out[106]:

[5, 6, 7, 8, 9, 8, 92, 103]

In [107]:

```
1 x = lambda l : [ (i+2,i**2) for i in l ]
```

In [109]:

```
1 x(l1)
```

Out[109]:

```
[(5, 9), (6, 16), (7, 25), (8, 36), (9, 49), (8, 36), (92, 8100), (103, 10201)]
```

In [121]:

```
1 x = lambda l : [ (i+i,i**2) for i in l if i< 4 if i>1]
```

In [122]:

```
1 x(l)
```

Out[122]:

```
[(4, 4), (6, 9)]
```

In [123]:

```
1 l1 = []
2 for i in l:
3     if i <4 :
4         l1.append((i**2, i+i))
```

In [124]:

```
1 l1
```

Out[124]:

```
[(1, 2), (4, 4), (9, 6)]
```

In [125]:

```
1 d = {1 : 1, 2 :4 , 3 :9}
```

In [127]:

```
1 x = lambda l : { i: i**2 for i in range(l) }
```

In [129]:

```
1 x(10)
```

Out[129]:

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

In [135]:

```
1 def test21(l):
2     d={}
3     for i in range(l):
4         d[i] = i**2
5     return d
6
```

In [136]:

```
1 test21(10)
```

Out[136]:

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

In [137]:

```
1 d = {}
2 for i in range(10):
3     d[i] = i**2
```

In [138]:

```
1 d
```

Out[138]:

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

In [139]:

```
1 ( i for i in range(10))
```

Out[139]:

```
<generator object <genexpr> at 0x0000021909D85A50>
```

In [140]:

```
1 tuple(( i for i in range(10)))
```

Out[140]:

```
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

In [142]:

```
1 a = 56
2 for i in a :
3     print(i)
```

```
-----
-
TypeError                                 Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_32896\4248146362.py in <module>
      1 a = 56
----> 2 for i in a :
      3     print(i)

TypeError: 'int' object is not iterable
```

In [145]:

```
1 s = 'sudh'
2 for i in a :
3     print(i)
```

```
s
u
d
h
```

In [146]:

```
1 next(s)
```

```
-----
-
TypeError                                 Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_32896\1977886155.py in <module>
      1 next(s)

TypeError: 'str' object is not an iterator
```

In [147]:

```
1 b = iter(s)
```

In [148]:

```
1 b
```

Out[148]:

```
<str_iterator at 0x21909d8b400>
```

In [149]:

```
1 next(b)
```

Out[149]:

```
's'
```

In [150]:

```
1 next(b)
```

Out[150]:

```
'u'
```

In [151]:

```
1 next(b)
```

Out[151]:

```
'd'
```

In [152]:

```
1 s = 'fsdsfsds'
2 for i in s :
3     print(i)
```

```
f
s
d
s
f
s
d
s
```

In [153]:

```
1 s = 'sud'
2 d = iter(s)
```

In [154]:

```
1 next(d)
```

Out[154]:

```
's'
```

In [155]:

```
1 next(d)
```

Out[155]:

```
'u'
```

In [156]:

```
1 next(d)
```

Out[156]:

```
'd'
```

In [157]:

```
1 next(d)
```

```
-
StopIteration
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_32896/3131910724.py in <module>
----> 1 next(d)
```

Traceback (most recent call last)

StopIteration:

In [158]:

```
1 l = [3,4,5,6]
```

In [159]:

```
1 next(l)
```

```
-
TypeError
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_32896/2016234150.py in <module>
----> 1 next(l)
```

Traceback (most recent call last)

TypeError: 'list' object is not an iterator

In [3]:

```
1 l = [3,4,5,6]
2
```

In [9]:

```
1 l
```

Out[9]:

```
<list_iterator at 0x1427b092790>
```

In [10]:

```
1 l = iter(1)
```

In [11]:

```
1 next(l)
```

Out[11]:

4

In [12]:

```
1 next(l)
```

Out[12]:

5

In [13]:

```
1 next(l)
```

Out[13]:

6

In []:

```
1
```

In []:

```
1 print('fsds')
```

In []:

```
1 import logging
```

In []:

```
1 logging.basicConfig(filename = 'test.log')
```

```
1 debug  
2 info  
3 warning  
4 error  
5 critical
```

In []:

```
1 logging.info('this is my info log')  
2 logging.warning('this is my warning')  
3 logging.error('this is my error log')
```

In []:

```
1 logging.shutdown()
```

In []:

```
1 # to reflect info  
2 logging.basicConfig(filename = 'test.log', level = logging.INFO)
```

In []:

```
1 logging.info('this is my info log')  
2 logging.warning('this is my warning')  
3 logging.error('this is my error log')
```

In []:

```
1 logging.shutdown()
```

In [1]:

```
1 import logging  
2 logging.basicConfig(filename = 'test3.log' , level = logging.INFO , format ='%(asctime
```

In [2]:

```
1 logging.info('this is my info log')
2 logging.warning('this is my warning log')
3 logging.debug('this is my debug log')
4 logging.error('this is my error log')
```

```
1 Priority
2 ERROR
3 WARNING
4 INFO
5 DEBUG
```

In [1]:

```
1 import logging
2 logging.basicConfig(filename = 'test4.log' , level = logging.DEBUG , format ='%(asctime)s - %(name)s - %(levelname)s - %(message)s')
```

In [2]:

```
1 def dividebyzero(a,b):
2     logging.info('This is a start of my code and I am trying to enter %s and %s', a, b)
3     try:
4         div = a/b
5         logging.info('executed successfully')
6     except Exception as e:
7         logging.error('error has happened')
8         logging.exception('Exception occurred ' + str(e))
9
```

In [3]:

```
1 dividebyzero(5,0)
```

In [4]:

```
1 import Testing2
```

In [5]:

```
1 Testing2.test(5,6)
```

Out[5]:

11

In [8]:

```
1 # dir()
```

In [9]:

```
1
```

Out[9]:

```
<_io.TextIOWrapper name='mymodule.py' mode='w' encoding='cp1252'>
```

In []:

```
1
```

In [1]:

```
1 l = [1,2,3,4,5]
```

In [3]:

```
1 l1 = []
2 for i in l:
3     l1.append(i+5)
4 print(l1)
```

[6, 7, 8, 9, 10]

In [5]:

```
1 l = [i+5 for i in l ]
```

In [18]:

```
1 l = [1,2,3,4]
```

In [15]:

```
1 x = lambda l: [i**2 for i in l]
```

In [19]:

```
1 x(l)
```

Out[19]:

[1, 4, 9, 16]

In [20]:

```
1 [i**2 for i in l]
```

Out[20]:

[1, 4, 9, 16]

In [22]:

```
1 def test(a):
2     return a**2
```

In [23]:

```
1 test(5)
```

Out[23]:

In [29]:

```
1 x = map(test, [1,2,3,4,5] )
```

In [30]:

```
1 list(x)
```

Out[30]:

```
[1, 4, 9, 16, 25]
```

In [31]:

```
1 list(map(test, [1,2,3,4,5] ))
```

Out[31]:

```
[1, 4, 9, 16, 25]
```

In [32]:

```
1 def test1(a):
2     return a*345+67
```

In [34]:

```
1 x = map(test, 1)
```

In [35]:

```
1 list(x)
```

Out[35]:

```
[1, 4, 9, 16]
```

In [36]:

```
1 l1 = ['34', '45' , '56', '78']
```

In [37]:

```
1 def test2(a):
2     return int(a)
```

In [38]:

```
1 x = map(test2, l1)
```

In [39]:

```
1 list(x)
```

Out[39]:

```
[34, 45, 56, 78]
```

In [42]:

```
1 type(list(x))
```

Out[42]:

list

In [44]:

```
1 list(map(lambda x : int(x), l1))
```

Out[44]:

[34, 45, 56, 78]

In [49]:

```
1 list(map(lambda x : x+5, l1))
```

Out[49]:

[6, 7, 8, 9]

In [50]:

```
1 l = [4,5,6,7,78,8,99]
```

In [51]:

```
1 def test1(a):
2     return a%2 == 0
```

In [52]:

```
1 list(map(test1, l))
```

Out[52]:

[True, False, True, False, True, True, False]

In [55]:

```
1 x = lambda l : [i for i in l if i%2 == 0]
```

In [56]:

```
1 x(l)
```

Out[56]:

[4, 6, 78, 8]

In [57]:

```
1 def test2(a):
2     if a%2 == 0:
3         return a
```

In [59]:

```
1 x = filter(test2, l)
```

In [60]:

```
1 list(x)
```

Out[60]:

```
[4, 6, 78, 8]
```

In [61]:

```
1 # map means mapping every element
2 # filter means only those elements which is true
```

In [69]:

```
1 list(filter(lambda x : x%2 == 0 , l))
```

Out[69]:

```
[4, 6, 78, 8]
```

In [71]:

```
1 list(filter(lambda x : x+2 , l))
```

Out[71]:

```
[4, 5, 6, 7, 78, 8, 99]
```

In [72]:

```
1 list(filter(lambda x : x%3 == 0 , l))
```

Out[72]:

```
[6, 78, 99]
```

In [73]:

```
1 # Reduce
```

In [74]:

```
1 from functools import reduce
```

In [75]:

```
1 l = [3,4,5,6,7,8,9,2]
```

In [76]:

```
1 sum(l)
```

Out[76]:

44

In [78]:

```
1 x = lambda a,b : a*b
```

In [79]:

```
1 x(5,6)
```

Out[79]:

30

In [80]:

```
1 def test5(a,b):
2     return a*b
```

In [81]:

```
1 reduce(test5, 1)
```

Out[81]:

362880

In [83]:

```
1 def test5(a,b):
2     return a/b
```

In [84]:

```
1 reduce(test5, 1)
```

Out[84]:

2.48015873015873e-05

In [85]:

```
1 def test5(a,b,c):
2     return a+b+c
```

In [86]:

```
1 reduce(test5, 1)
```

-

TypeError

Traceback (most recent call last)

```
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_9876/3013230881.py in <module>
    ----> 1 reduce(test5, 1)
```

TypeError: test5() missing 1 required positional argument: 'c'

In [87]:

```
1 l = [3]
```

In [88]:

```
1 reduce(test5, 1)
```

Out[88]:

3

In [89]:

```
1 l = [3,4,5,6,7,8,9,2]
2 reduce(lambda x,y : x+y, 1)
```

Out[89]:

44

In [95]:

```
1 l = [1,2,3,4,5]
2 l1 = [4,5,6,7,8]
3 l2 = ['sudh' , 'raksha' , 'raj']
```

In [96]:

```
1 list(zip(l,l1,l2))
```

Out[96]:

```
[(1, 4, 'sudh'), (2, 5, 'raksha'), (3, 6, 'raj')]
```

In [104]:

```
1 x = enumerate([1,2,3,4,5])
```

In [106]:

```
1 print(x)
```

```
<enumerate object at 0x000001E38B34D2C0>
```

In []:

```
1
```

In [1]:

```
1 import os
```

In [3]:

```
1 def test(a,b):  
2     return a+b
```

In [5]:

```
1 open('Testing2.py', 'w')
```

Out[5]:

```
<_io.TextIOWrapper name='Testing2.py' mode='w' encoding='cp1252'>
```

In [1]:

```
1 import Testing2
```

In [2]:

```
1 Testing2.test(6,7)
```

Out[2]:

```
13
```

In [3]:

```
1 Testing2.div(20,5)
```

Out[3]:

```
4.0
```

In [5]:

```
1 Testing2.mul(21,6)
```

Out[5]:

```
126
```

In [1]:

```
1 from Testing2 import *
```

In [2]:

```
1 div(30,5)
```

Out[2]:

```
6.0
```

In [3]:

```
1 mul(6,7)
```

Out[3]:

42

In [4]:

```
1 open('mymodule.py' , 'w')
```

Out[4]:

```
<_io.TextIOWrapper name='mymodule.py' mode='w' encoding='cp1252'>
```

In [1]:

```
1 import mymodule
```

In [5]:

```
1 mymodule.get_course()
```

Out[5]:

```
['data science', 'blockchain', 'drone', 'robotics', 'cloud']
```

In [6]:

```
1 mymodule.greetings()
```

Out[6]:

```
'greeting from ineuron'
```

In [7]:

```
1 mymodule.key_find()
```

Out[7]:

```
dict_keys(['name', 'course', 'greeting'])
```

In [8]:

```
1 mymodule.value_find()
```

Out[8]:

```
dict_values(['ineuron', ['data science', 'blockchain', 'drone', 'robotics', 'cloud'], 'greeting from ineuron'])
```

In [2]:

```
1 mymodule.Item_find()
```

Out[2]:

```
dict_items([('name', 'ineuron'), ('course', ['data science', 'blockchain', 'drone', 'robotics', 'cloud']), ('greeting', 'greeting from ineuron')])
```

In [3]:

```
1 # Packages are collection of modules
```

In []:

```
1
```

In [1]:

```
1 class car:  
2     pass
```

In [6]:

```
1 # init is a constucter  
2 # constucter is a entity which will pass data to a class  
3 # self always behaves as a pointer  
4 # and self is pointing to the class car  
5  
6  
7 class car:  
8     def __init__(self, brand_name, fueltype, body_type):  
9         self.brand_name = brand_name  
10        self.fueltype = fueltype  
11        self.body_type = body_type  
12  
13    def desc_car(self):  
14        print(self.brand_name, self.fueltype, self.body_type)
```

In [7]:

```
1 innova.body_type
```

Out[7]:

```
'suv'
```

In [8]:

```
1 innova = car('toyota', 'petrol', 'suv')  
2 nexon = car('tata', 'petrol', 'suv')  
3 fortuner= car('toyota', 'desiel', 'suv')
```

In [10]:

```
1 innova.desc_car()
```

```
toyota petrol suv
```

In [9]:

```
1 innova
```

Out[9]:

```
<__main__.car at 0x272760dfd90>
```

In [26]:

```
1 innova.desc_car()
```

```
toyota petrol suv
```

In [27]:

```
1 innova.brand_name
```

Out[27]:

```
'toyota'
```

In [28]:

```
1 n exon.desc_car()
```

```
tata petrol suv
```

In [31]:

```
1 fortuner.desc_car()
```

```
toyota desiel suv
```

In [32]:

```
1 a = 10
```

In [33]:

```
1 print(type(a))
```

```
<class 'int'>
```

In [34]:

```
1 s = 'sudh'
```

In [35]:

```
1 print(type(s))
```

```
<class 'str'>
```

In [13]:

```
1 class car:  
2  
3     def test(self):  
4         print('this is my first method')
```

In [14]:

```
1 x = car()
```

In [15]:

```
1 x.test()
```

this is my first method

In [16]:

```
1 print(type(x))
```

<class '__main__.car'>

In [57]:

```
1 class list_parser:
2     def parser(self,a):
3         if type(a) == list:
4             for i in a :
5                 print(i, end = " ")
6
7     def reverse_list(self,a):
8         if type(a) == list:
9             for i in range(len(a),-1,-1):
10                print(i,end = " ")
11
```

In [58]:

```
1 x = list_parser()
```

In [59]:

```
1 x.parser([1,2,3,4,5])
```

1 2 3 4 5

In [60]:

```
1 x.reverse_list([1,2,3,4,5,6])
```

6 5 4 3 2 1 0

In [61]:

```
1 l = [1,2,3,4]
2 for i in range(len(l),-1,-1):
3     print(i, end = ' ')
```

4 3 2 1 0

In [62]:

```
1 class list_parser:
2     def parser(self,a):
3         if type(a) == list:
4             for i in a :
5                 print(i)
6
7     def reverse_list(self,a):
8         if type(a) == list:
9             return a[::-1]
10
```

In [63]:

```
1 x= list_parser()
```

In [64]:

```
1 x.reverse_list([1,2,3,4,5])
```

Out[64]:

[5, 4, 3, 2, 1]

In [82]:

```
1 class list_parser:
2     def __init__(self,a):
3         self.a = a
4
5     def parser(self):
6         if type(self.a) == list:
7             for i in self.a :
8                 print(i, end = " ")
9
10    def reverse_list(self):
11        if type(self.a) == list:
12            return self.a[::-1]
13
```

In [83]:

```
1 x = list_parser([1,2,3,4,5])
```

In [84]:

```
1 x.reverse_list()
```

Out[84]:

[5, 4, 3, 2, 1]

In [85]:

```
1 x.parser()
```

1 2 3 4 5

```
1 # Create a class for dict parsing
2 # 1 . give all the keys
3 2. give all the values
4 3. throw an exception in case input is not dict
5 4. to take user input and then parse key and value for dict
6 5. to insert new key value parse into dict
```

In [6]:

```
1 class dict_fun():
2     def __init__(self,a):
3         self.a = a
4
5     def get_keys(self):
6         if self.verify_input():
7             return list(self.a.keys())
8
9     def get_values(self):
10        if self.verify_input():
11            return self.a.values()
12
13    def verify_input(self):
14        if type(self.a) !=dict:
15            raise Exception('input is not dictionary' , b)
16
17    def user_input(self):
18        b= eval(input())
19        self.a.update(b)
20        print(self.a)
21        print(self.a.keys())
22
23    def add_new_dict(self,b):
24        if type(b) == dict:
25            self.a.update(b)
26        return self.a
27
28    def insertion(self,k,v):
29        self.a[k] = v
30        return self.a
31
32    def user_insertion(self):
33        k = input()
34        v = input()
35        self.a[k] = v
36        return self.a
```

In [7]:

```
1 x = dict_fun({'a':5, 'b':6 , 'c':7})
```

In [8]:

```
1 x.get_keys()
```

In [9]:

```
1 x.user_insertion()
```

5

6

Out[9]:

```
{'a': 5, 'b': 6, 'c': 7, '5': '6'}
```

In [10]:

```
1 x.get_keys()
```

In [11]:

```
1 x.get_values()
```

In [12]:

```
1 x.verify_input([1,2,3])
```

```
-----
-
TypeError                                     Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_20384\2802977484.py in <mod
ule>
----> 1 x.verify_input([1,2,3])

TypeError: verify_input() takes 1 positional argument but 2 were given
```

In [13]:

```
1 x.user_input()
```

Traceback (most recent call last):

```
  File "C:\Users\RAKSHANDA\anaconda3\lib\site-packages\IPython\core\interactiveshell.py", line 3444, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
```

```
  File "C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_20384/1838609356.py", line 1, in <module>
    x.user_input()
```

```
  File "C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_20384/72099168.py", line 18, in user_input
    b= eval(input())
```

```
File "<string>", line unknown
```

```
^
```

SyntaxError: unexpected EOF while parsing

In [14]:

```
1 x.add_new_dict({"e":5, "f" : 6})
```

Out[14]:

```
{'a': 5, 'b': 6, 'c': 7, '5': '6', 'e': 5, 'f': 6}
```

In [15]:

```
1 x.insertion('h' , '67')
```

Out[15]:

```
{'a': 5, 'b': 6, 'c': 7, '5': '6', 'e': 5, 'f': 6, 'h': '67'}
```

In [16]:

```
1 class dict_fun():
2     def __init__(self,a):
3         self.a = a
4
5     def get_keys(self):
6         if self.verify_input():
7             return list(self.a.keys())
8
9     def get_values(self):
10        if self.verify_input():
11            return self.a.values()
12
13    def verify_input(self):
14        if type(self.a) !=dict:
15            raise Exception('input is not dictionary' , self.a)
16        return 1
```

In [17]:

```
1 d = dict_fun({'a': 5, 'b': 6, 'c': 7, 'h': '67'})
```

In [18]:

```
1 d.get_keys()
```

Out[18]:

```
['a', 'b', 'c', 'h']
```

In [38]:

```
1 # Final_Program
2
3 class Dict_function :
4     def __init__(self,a):
5         self.a = a
6
7     def verify_dict(self):
8         if type(self.a) != dict:
9             raise Exception(self.a , 'is not a dictionary')
10    return 1
11
12
13     def get_keys(self):
14         if self.verify_dict():
15             return list(self.a.keys())
16
17     def get_values(self):
18         if self.verify_dict():
19             return list(self.a.values())
20
21
22     def user_input(self):
23         b = eval(input())
24         self.a.update(b)
25         return self.a,
26
27     def insertion(self,**kwargs):
28         for k , v in kwargs.items():
29             self.a[k] = v
30         return self.a
31
32
```

In [30]:

```
1 x= Dict_function({'a': 5, 'b': 6, 'c': 7, 'h': '67'})
```

In [31]:

```
1 x.get_keys()
```

Out[31]:

```
['a', 'b', 'c', 'h']
```

In [32]:

```
1 x.get_values()
```

Out[32]:

```
[5, 6, 7, '67']
```

In [34]:

```
1 x.user_input()  
  
{'g' : 45}
```

Out[34]:

```
({'a': 5, 'b': 6, 'c': 7, 'h': '67', 'g': 45},)
```

In [35]:

```
1 x.insertion(name = 'raksha', g = 35)
```

Out[35]:

```
{'a': 5, 'b': 6, 'c': 7, 'h': '67', 'g': 35, 'name': 'raksha'}
```

In [39]:

```
1 open('mydict.py', 'w')
```

Out[39]:

```
<_io.TextIOWrapper name='mydict.py' mode='w' encoding='cp1252'>
```

In []:

```
1
```

In [1]:

```
1 class xyz :  
2     def __init__(self, a,b,c):  
3         self.a = a  
4         self.b = b  
5         self.c = c  
6  
7     def test(self):  
8         print('this is my first print method of xyz class')  
9  
10    def test1(self):  
11        print('this is my test2 method of xyz class ')  
12  
13    def test2(self):  
14        print('this is my test3 method of xyz class')  
15  
16
```

In [6]:

```
1 p = xyz(1,2,3)
```

In [7]:

```
1 p.test1()
```

this is my test2 method of xyz class

In [5]:

```
1 class xyz1(xyz):  
2     pass
```

In [9]:

```
1 q = xyz1(1,2,3)
```

In [15]:

```
1 q.test1()
```

this is my test2 method of xyz class

In [16]:

```
1 q = xyz1()
```

```
-  
TypeError Traceback (most recent call last)  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_2300\2225742902.py in <modu  
le>  
----> 1 q = xyz1()  
  
TypeError: __init__() missing 3 required positional arguments: 'a', 'b', a  
nd 'c'
```

In [17]:

```
1 class xyz1(xyz):  
2     def test(self):  
3         print('this is the test method available in xyz1')
```

In [18]:

```
1 g = xyz1(1,2,3)
```

In [19]:

```
1 g.test()
```

```
this is the test method available in xyz1
```

In [39]:

```
1 class xyz :  
2     def __init__(self, a,b,c):  
3         self.a = a  
4         self.b = b  
5         self.c = c  
6  
7     def test(self):  
8         print('this is my first print method of xyz class')  
9  
10  
11 class xyz1:  
12     def __init__(self, p,q,r):  
13         self.p = p  
14         self.q = q  
15         self.r = r  
16  
17     def test1(self):  
18         print('this is a meth from class xyz1')  
19  
20 class child(xyz1,xyz):  
21     pass  
22  
23 #class child(xyz1,xyz):  
24 #    pass  
25  
26 # variable wise it will work for first argument  
27 # method wise it will work for all the methods
```

In [40]:

```
1 n = child(4,5,6)
```

In [41]:

```
1 n.test()
```

this is my first print method of xyz class

In [42]:

```
1 n.test1()
```

this is a meth from class xyz1

In [51]:

```
1 # Multiple Inheritance
2 class xyz :
3     def __init__(self, a,b,c):
4         self.a = a
5         self.b = b
6         self.c = c
7
8     def test(self):
9         print('this is my first print method of xyz class')
10
11
12 class xyz1:
13     def __init__(self, p,q,r):
14         self.p = p
15         self.q = q
16         self.r = r
17
18     def test1(self):
19         print('this is a meth from class xyz1')
20
21 class child(xyz1,xyz):
22     def __init__(self,*args, **kwargs):
23         xyz.__init__(self,*args)
24         xyz1.__init__(self,**kwargs)
25
26
```

In [53]:

```
1 n = child(4,5,6,p=5, q=6, r =7)
```

In [54]:

```
1 n.a
```

Out[54]:

4

In [55]:

```
1 n.p
```

Out[55]:

5

In [56]:

```
1 n.test()
```

this is my first print method of xyz class

In [57]:

```
1 n.test1()
```

this is a meth from class xyz1

In [65]:

```
1 # Multilevel Inheritance
2 class xyz :
3     def __init__(self, a,b,c):
4         self.a = a
5         self.b = b
6         self.c = c
7
8     def test(self):
9         print('this is my first print method of xyz class')
10
11
12 class xyz1(xyz):
13
14     def test1(self):
15         print('this is a meth from class xyz1')
16
17 class xyz2(xyz1):
18     def test2(self):
19         print('this is meth from xlass xyz2')
20
21 class xyz3(xyz2):
22     def test3(self):
23         print('this is meth from xyz3')
```

In [66]:

```
1 v = xyz2(4,5,6)
```

In [67]:

```
1 v = xyz3(5,6,7)
```

In [68]:

```
1 v.test3()
```

this is meth from xyz3

In [132]:

```
1 class File:
2     def write(self,data):
3         f = open('inherit.txt', "w")
4         f.write(data)
5         f.close()
6
7     def read(self):
8         f = open('inherit.txt', 'r')
9         f.seek(0)
10        print(f.read())
11
12 class child(File):
13     def test(self):
14         pass
```

In [133]:

```
1 x = File()
```

In [135]:

```
1 x.write('Writing the file')
```

In [136]:

```
1 x.read()
```

Writing the file

In [94]:

```
1 x1 = child()
```

In [95]:

```
1 x1.read()
```

this is my first write method

In [119]:

```
1 import logging
2 logging.basicConfig(filename ='List.log',level=logging.DEBUG,format ="%(asctime)s,%(l
3
4 class File:
5     def __init__(self,filename):
6         logging.info('Creating file Instance variable')
7         self.file = filename
8
9     def read(self):
10        logging.info('Executing read method')
11        try:
12            with open(f'{self.file}.txt' , 'r') as f:
13                data = f.read()
14                return data
15        except FileNotFoundError as e:
16            logging.error('Error occured while opening the file')
17            logging.exception(f"Error is {s}")
18
19    def write(self,data):
20        logging.info('Executing write method')
21        try:
22            with open(f'{self.file}.txt' , "w") as f:
23                data = f.write(data)
24                f.close()
25
26        except FileNotFoundError as e:
27            logging.error('Something is going wrong')
28
29 class Child(File):
30     def test(self):
31         print('this is child class')
```

In [125]:

```
1 x = Child('inherit')
```

In [126]:

```
1 x.write('writing file')
```

In [127]:

```
1 x.read()
```

Out[127]:

'writing file'

In [121]:

```
1 Obj = File('inherit')
```

In [111]:

```
1 Obj.read()
```

Out[111]:

```
'writing file'
```

In [112]:

```
1 Obj.write('writing file')
```

In []:

```
1
```

In [6]:

```
1 # one single entity but behaviour is change in diff diff situatiion  
2 # eg . sometime concat sometimes addition
```

In [7]:

```
1 def test(a,b):  
2     return a+ b
```

In [8]:

```
1 test(3,4)  
2
```

Out[8]:

7

In [9]:

```
1 test('sudh' , 'kumar')
```

Out[9]:

'sudhkumar'

In [11]:

```
1 class insta:  
2     def share_stories(self):  
3         print('this will share my insta stories')  
4  
5 class facebook:  
6     def share_stories(self):  
7         print('this will share my fb stories')
```

In [13]:

```
1 def sharestories(app):  
2     app.share_stories()
```

In [14]:

```
1 i = insta()  
2 f = facebook()
```

In [15]:

```
1 sharestories(i)
```

this will share my insta stories

In [16]:

```
1 sharestories(f)
```

this will share my fb stories

In [19]:

```
1 class social_media:
2     def share_stories(self):
3         print('share a stories')
4     def upload_pic(self):
5         print('upload a pic')
6
7 class facebook(social_media):
8     def share_stories(self):
9         print('tjis is a fun for facebook shareig story')
10
11 class insta(social_media):
12     def share_stories(self):
13         print('this will print story from insta')
```

In [20]:

```
1 f = facebook()
2 i = insta()
```

In [21]:

```
1 f.share_stories()
```

tjis is a fun for facebook shareig story

In [22]:

```
1 i.share_stories()
```

this will print story from insta

In [26]:

```
1 class test(Exception):
2     def __init__(self, msg , myvalue):
3         self.msg = msg
4         self.myvalue = myvalue
5
```

In [30]:

```
1 try:
2     raise(test('this is my own exception class' , "fsfds"))
3     5/0
4 except test as t:
5     print(t)
```

('this is my own exception class', 'fsfds')

In [32]:

```
1 try:  
2     raise(test('this is my own exception class', 5 ))  
3 except test as t:  
4     print(t)
```

```
('this is my own exception class', 5)
```

In [36]:

```
1 # multiple inheritance and static variable eg  
2  
3 class Batch_number :  
4     batchnumber = ""  
5     def batch(self):  
6         print('self.batchnumber')  
7  
8 class course_name :  
9     cname = ""  
10    def course(self):  
11        print(self.cname)  
12  
13 class Student(Batch_number, course_name):  
14     def deets(self):  
15         print('batch : ', self.batchnumber )  
16         print('course : ', self.cname)  
17  
18 s1 = Student()  
19 s1.batchnumber = "2"  
20 s1.cname = 'FSDS'  
21 s1.deets()
```

```
batch : 2  
course : FSDS
```

In [38]:

```
1 class Batch_number :
2     batchnumber = ""
3
4     def __init__(self,a,b,c):
5         self.a = a
6         self.b = b
7         self.c = c
8
9     def batch(self):
10        print('self.batchnumber')
11
12 class course_name :
13     cname = ""
14     def course(self):
15         print(self.cname)
16
17 class Student(Batch_number, course_name):
18     def deets(self):
19         print('batch : ', self.batchnumber )
20         print('course : ', self.cname)
21
22 s1 = Student(4,5,6)
23 s1.batchnumber = "2"
24 s1.cname = 'FSDS'
25 s1.deets()
```

batch : 2
course : FSDS

In [40]:

```
1 bn = Batch_number(4,5,6)
```

In [41]:

```
1 bn.batchnumber
```

Out[41]:

''

In [42]:

```
1 bn.batchnumber = 6
```

In [43]:

```
1 bn.batchnumber
```

Out[43]:

6

In [44]:

```
1 bn.a
```

Out[44]:

```
4
```

In [46]:

```
1 bn1 = Batch_number(1,2,3)
```

In [47]:

```
1 bn1.a
```

Out[47]:

```
1
```

In [48]:

```
1 bn1.batchnumber
```

Out[48]:

```
..
```

In [49]:

```
1 Batch_number.batchnumber
```

Out[49]:

```
..
```

In [50]:

```
1 bn1.batchnumber = 'sudh'
```

In [53]:

```
1 bn1.batch()
```

```
self.batchnumber
```

In [54]:

```
1 bn1.batchnumber
```

Out[54]:

```
'sudh'
```

In [58]:

```
1 Batch_number.batchnumber = 'kumar'
```

In [59]:

```
1 bn1.batchnumber
```

Out[59]:

```
'sudh'
```

In [60]:

```
1 bn.batchnumber
```

Out[60]:

```
6
```

In [61]:

```
1 bn1.batch()
```

```
self.batchnumber
```

In [64]:

```
1 Batch_number.batch('sudh')
```

```
self.batchnumber
```

In [66]:

```
1 # Static method
2 class Batch_number :
3     batchnumber = ""
4
5     def __init__(self,a,b,c):
6         self.a = a
7         self.b = b
8         self.c = c
9
10    @staticmethod
11    def batch():
12        print(self.batchnumber)
13
14 class course_name :
15     cname = ""
16     def course(self):
17         print(self.cname)
18
19 class Student(Batch_number, course_name):
20     def deets(self):
21         print('batch : ', self.batchnumber )
22         print('course : ', self.cname)
23
24 s1 = Student(4,5,6)
25 s1.batchnumber = "2"
26 s1.cname = 'FSDS'
27 s1.deets()
```

batch : 2

course : FSDS

In [88]:

```
1 class Batch_number :
2     batchnumber = ""
3
4     def __init__(self,a,b,c):
5         self.a = a
6         self.b = b
7         self.c = c
8
9     #@staticmethod
10    def batch(self):
11        print(self.batchnumber)
12
```

In [89]:

```
1 bn = Batch_number(3,4,5)
```

In [90]:

```
1 bn1.batchnumber = 'sudh'
```

In [91]:

```
1 bn1.batchnumber
```

Out[91]:

```
'sudh'
```

In [94]:

```
1 bn.batch()
```

In [93]:

```
1 bn1.batch()
```

```
self.batchnumber
```

In [95]:

```
1 bn1 = Batch_number(3,4,5)
```

In [96]:

```
1 bn1.batch()
```

In [134]:

```
1 # STATIC METHOD
2 # YOU can access static method as well as static argument through class name
3 # but instance variable u can access only after creating object
4 class Batch_number :
5     batchnumber = "FSDS"
6
7     def __init__(self,a,b,c):
8         self.a = a
9         self.b = b
10        self.c = c
11
12        @staticmethod
13        def batch():
14            print('this is my static method')
15
16        def batch1(self):
17            print('this is class method', self.batchnumber)
```

In [135]:

```
1 bn = Batch_number(3,4,5)
```

In [136]:

```
1 #Batch_number.batchnumber = 'sudh'  
2 bn.batch()
```

this is my static method

In [137]:

```
1 Batch_number.batchnumber
```

Out[137]:

'FSDS'

In [138]:

```
1 bn.batch1()
```

this is class method FSDS

In [139]:

```
1 Batch_number.a
```

```
-----  
-  
AttributeError                                     Traceback (most recent call last  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19436/1348857264.py in <mod  
ule>  
----> 1 Batch_number.a
```

AttributeError: type object 'Batch_number' has no attribute 'a'

In [140]:

```
1 bn.a
```

Out[140]:

3

In [141]:

```
1 Batch_number.batch()
```

this is my static method

In [142]:

```
1 Batch_number.batch1()
```

```
-  
TypeError  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19436/1380430957.py in <mod  
ule>  
----> 1 Batch_number.batch1()  
  
TypeError: batch1() missing 1 required positional argument: 'self'
```

In [143]:

```
1 class Accountdetail:  
2     account_no = '3432423421334'  
3     def __init__(self,amount):  
4         self.amount = amount  
5  
6     def show_balance(self, deduction):  
7         self.amount = self.amount - deduction  
8         return self.amount , account_no
```

In [146]:

```
1 class division_by_5 (Exception):
2     def __init__(self,msg):
3         self.msg = msg
4
5     try :
6         a = int(input('Enter the first no: '))
7         b = int(input('Enter the second no : '))
8
9         if b == 5 :
10             raise division_by_5('division by 5 error')
11
12     else :
13         result = a / b
14
15 except ValueError as msg:
16     print(msg)
17
18 except ZeroDivisionError as msg:
19     print(msg)
20
```

```
Enter the first no: 3
Enter the second no : 5
```

```
-----
-
division_by_5                                     Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19436/548931067.py in <module>
     8
     9     if b == 5 :
--> 10         raise division_by_5('division by 5 error')
    11
    12     else :
```

division_by_5: division by 5 error

In [147]:

```
1 # Overloading and overriding
```

In [152]:

```
1 class test:
2     def fun(test):
3         print('this is my sample class')
4
5     def __str__(self):
6         return str('this is the function call at the time of object print')
```

In [153]:

```
1 t = test()
```

In [154]:

```
1 print(t)
```

this is the function call at the time of object print

In [156]:

```
1 t
```

Out[156]:

<__main__.test at 0x1fa3bef67f0>

In [163]:

```
1 class test1:
2     def fun(test):
3         print('this is my sample class')
```

In [167]:

```
1 t1 = test1()
```

In [168]:

```
1 print(t1)
```

<__main__.test1 object at 0x000001FA3C046610>

In [169]:

```
1 t1
```

Out[169]:

<__main__.test1 at 0x1fa3c046610>

In [170]:

```
1 # Overloading
2
3 class test:
4     def __init__(self,a,b,c):
5         self.a = a
6         self.b = b
7         self.c = c
```

In [172]:

```
1 t = test(4,5,6)
```

In [173]:

```
1 print(t)
```

```
<__main__.test object at 0x000001FA3BDA7520>
```

In [174]:

```
1
2 class test:
3     def __init__(self,a,b,c):
4         self.a = a
5         self.b = b
6         self.c = c
7
8     def __str__(self):
9         return 'fdsfsd'
```

In [175]:

```
1 t1 = test(4,5,6)
```

In [176]:

```
1 print(t1)
```

```
fdsfsd
```

In [177]:

```
1 # Overriding
2
3 class test:
4     def __init__(self,a,b,c):
5         self.a = a
6         self.b = b
7         self.c = c
8
9     def __str__(self):
10        return 'fdsfsd'
11
12     def class_fun(self):
13         print('this is a print from class_fun')
14
15 class test1(test):
16     def class_fun(self) :
17         print('fsfsafas overriding method from test1')
18
```

In [178]:

```
1 test1_obj = test1(1,2,3)
2 test_obj = test(4,5,6)
```

In [180]:

```
1 test1_obj.class_fun()
```

fsfsafas overriding method from test1

In [181]:

```
1 test_obj.class_fun()
```

this is a print from class_fun

In []:

```
1
```

In [6]:

```
1 # one single entity but behaviour is change in diff diff situatiion  
2 # eg . sometime concat sometimes addition
```

In [7]:

```
1 def test(a,b):  
2     return a+ b
```

In [8]:

```
1 test(3,4)  
2
```

Out[8]:

7

In [9]:

```
1 test('sudh' , 'kumar')
```

Out[9]:

'sudhkumar'

In [11]:

```
1 class insta:  
2     def share_stories(self):  
3         print('this will share my insta stories')  
4  
5 class facebook:  
6     def share_stories(self):  
7         print('this will share my fb stories')
```

In [13]:

```
1 def sharestories(app):  
2     app.share_stories()
```

In [14]:

```
1 i = insta()  
2 f = facebook()
```

In [15]:

```
1 sharestories(i)
```

this will share my insta stories

In [16]:

```
1 sharestories(f)
```

this will share my fb stories

In [19]:

```
1 class social_media:
2     def share_stories(self):
3         print('share a stories')
4     def upload_pic(self):
5         print('upload a pic')
6
7 class facebook(social_media):
8     def share_stories(self):
9         print('tjis is a fun for facebook shareig story')
10
11 class insta(social_media):
12     def share_stories(self):
13         print('this will print story from insta')
```

In [20]:

```
1 f = facebook()
2 i = insta()
```

In [21]:

```
1 f.share_stories()
```

tjis is a fun for facebook shareig story

In [22]:

```
1 i.share_stories()
```

this will print story from insta

In [26]:

```
1 class test(Exception):
2     def __init__(self, msg , myvalue):
3         self.msg = msg
4         self.myvalue = myvalue
5
```

In [30]:

```
1 try:
2     raise(test('this is my own exception class' , "fsfds"))
3     5/0
4 except test as t:
5     print(t)
```

('this is my own exception class', 'fsfds')

In [32]:

```
1 try:  
2     raise(test('this is my own exception class', 5 ))  
3 except test as t:  
4     print(t)
```

```
('this is my own exception class', 5)
```

In [36]:

```
1 # multiple inheritance and static variable eg  
2  
3 class Batch_number :  
4     batchnumber = ""  
5     def batch(self):  
6         print('self.batchnumber')  
7  
8 class course_name :  
9     cname = ""  
10    def course(self):  
11        print(self.cname)  
12  
13 class Student(Batch_number, course_name):  
14     def deets(self):  
15         print('batch : ', self.batchnumber )  
16         print('course : ', self.cname)  
17  
18 s1 = Student()  
19 s1.batchnumber = "2"  
20 s1.cname = 'FSDS'  
21 s1.deets()
```

```
batch : 2  
course : FSDS
```

In [38]:

```
1 class Batch_number :
2     batchnumber = ""
3
4     def __init__(self,a,b,c):
5         self.a = a
6         self.b = b
7         self.c = c
8
9     def batch(self):
10        print('self.batchnumber')
11
12 class course_name :
13     cname = ""
14     def course(self):
15         print(self.cname)
16
17 class Student(Batch_number, course_name):
18     def deets(self):
19         print('batch : ' , self.batchnumber )
20         print('course : ' , self.cname)
21
22 s1 = Student(4,5,6)
23 s1.batchnumber = "2"
24 s1.cname = 'FSDS'
25 s1.deets()
```

batch : 2
course : FSDS

In [40]:

```
1 bn = Batch_number(4,5,6)
```

In [41]:

```
1 bn.batchnumber
```

Out[41]:

''

In [42]:

```
1 bn.batchnumber = 6
```

In [43]:

```
1 bn.batchnumber
```

Out[43]:

6

In [44]:

```
1 bn.a
```

Out[44]:

```
4
```

In [46]:

```
1 bn1 = Batch_number(1,2,3)
```

In [47]:

```
1 bn1.a
```

Out[47]:

```
1
```

In [48]:

```
1 bn1.batchnumber
```

Out[48]:

```
..
```

In [49]:

```
1 Batch_number.batchnumber
```

Out[49]:

```
..
```

In [50]:

```
1 bn1.batchnumber = 'sudh'
```

In [53]:

```
1 bn1.batch()
```

```
self.batchnumber
```

In [54]:

```
1 bn1.batchnumber
```

Out[54]:

```
'sudh'
```

In [58]:

```
1 Batch_number.batchnumber = 'kumar'
```

In [59]:

```
1 bn1.batchnumber
```

Out[59]:

```
'sudh'
```

In [60]:

```
1 bn.batchnumber
```

Out[60]:

```
6
```

In [61]:

```
1 bn1.batch()
```

```
self.batchnumber
```

In [64]:

```
1 Batch_number.batch('sudh')
```

```
self.batchnumber
```

In [66]:

```
1 # Static method
2 class Batch_number :
3     batchnumber = ""
4
5     def __init__(self,a,b,c):
6         self.a = a
7         self.b = b
8         self.c = c
9
10    @staticmethod
11    def batch():
12        print(self.batchnumber)
13
14 class course_name :
15     cname = ""
16     def course(self):
17         print(self.cname)
18
19 class Student(Batch_number, course_name):
20     def deets(self):
21         print('batch : ', self.batchnumber )
22         print('course : ', self.cname)
23
24 s1 = Student(4,5,6)
25 s1.batchnumber = "2"
26 s1.cname = 'FSDS'
27 s1.deets()
```

```
batch : 2
course : FSDS
```

In [88]:

```
1 class Batch_number :
2     batchnumber = ""
3
4     def __init__(self,a,b,c):
5         self.a = a
6         self.b = b
7         self.c = c
8
9     #@staticmethod
10    def batch(self):
11        print(self.batchnumber)
12
```

In [89]:

```
1 bn = Batch_number(3,4,5)
```

In [90]:

```
1 bn1.batchnumber = 'sudh'
```

In [91]:

```
1 bn1.batchnumber
```

Out[91]:

```
'sudh'
```

In [94]:

```
1 bn.batch()
```

In [93]:

```
1 bn1.batch()
```

```
self.batchnumber
```

In [95]:

```
1 bn1 = Batch_number(3,4,5)
```

In [96]:

```
1 bn1.batch()
```

In [134]:

```
1 # STATIC METHOD
2 # YOU can access static method as well as static argument through class name
3 # but instance variable u can access only after creating object
4 class Batch_number :
5     batchnumber = "FSDS"
6
7     def __init__(self,a,b,c):
8         self.a = a
9         self.b = b
10        self.c = c
11
12        @staticmethod
13        def batch():
14            print('this is my static method')
15
16        def batch1(self):
17            print('this is class method', self.batchnumber)
```

In [135]:

```
1 bn = Batch_number(3,4,5)
```

In [136]:

```
1 #Batch_number.batchnumber = 'sudh'  
2 bn.batch()
```

this is my static method

In [137]:

```
1 Batch_number.batchnumber
```

Out[137]:

'FSDS'

In [138]:

```
1 bn.batch1()
```

this is class method FSDS

In [139]:

```
1 Batch_number.a
```

```
-----  
-  
AttributeError                                     Traceback (most recent call last  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19436/1348857264.py in <mod  
ule>  
----> 1 Batch_number.a
```

AttributeError: type object 'Batch_number' has no attribute 'a'

In [140]:

```
1 bn.a
```

Out[140]:

3

In [141]:

```
1 Batch_number.batch()
```

this is my static method

In [142]:

```
1 Batch_number.batch1()
```

```
-  
TypeError  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19436/1380430957.py in <mod  
ule>  
----> 1 Batch_number.batch1()  
  
TypeError: batch1() missing 1 required positional argument: 'self'
```

In [143]:

```
1 class Accountdetail:  
2     account_no = '3432423421334'  
3     def __init__(self,amount):  
4         self.amount = amount  
5  
6     def show_balance(self, deduction):  
7         self.amount = self.amount - deduction  
8         return self.amount , account_no
```

In [146]:

```
1 class division_by_5 (Exception):
2     def __init__(self,msg):
3         self.msg = msg
4
5     try :
6         a = int(input('Enter the first no: '))
7         b = int(input('Enter the second no : '))
8
9         if b == 5 :
10             raise division_by_5('division by 5 error')
11
12     else :
13         result = a / b
14
15 except ValueError as msg:
16     print(msg)
17
18 except ZeroDivisionError as msg:
19     print(msg)
20
```

```
Enter the first no: 3
Enter the second no : 5
```

```
-----
-
division_by_5                                     Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_19436/548931067.py in <module>
     8
     9     if b == 5 :
--> 10         raise division_by_5('division by 5 error')
    11
    12     else :
```

division_by_5: division by 5 error

In [147]:

```
1 # Overloading and overriding
```

In [152]:

```
1 class test:
2     def fun(test):
3         print('this is my sample class')
4
5     def __str__(self):
6         return str('this is the function call at the time of object print')
```

In [153]:

```
1 t = test()
```

In [154]:

```
1 print(t)
```

this is the function call at the time of object print

In [156]:

```
1 t
```

Out[156]:

<__main__.test at 0x1fa3bef67f0>

In [163]:

```
1 class test1:
2     def fun(test):
3         print('this is my sample class')
```

In [167]:

```
1 t1 = test1()
```

In [168]:

```
1 print(t1)
```

<__main__.test1 object at 0x000001FA3C046610>

In [169]:

```
1 t1
```

Out[169]:

<__main__.test1 at 0x1fa3c046610>

In [170]:

```
1 # Overloading
2
3 class test:
4     def __init__(self,a,b,c):
5         self.a = a
6         self.b = b
7         self.c = c
```

In [172]:

```
1 t = test(4,5,6)
```

In [173]:

```
1 print(t)
```

```
<__main__.test object at 0x000001FA3BDA7520>
```

In [174]:

```
1
2 class test:
3     def __init__(self,a,b,c):
4         self.a = a
5         self.b = b
6         self.c = c
7
8     def __str__(self):
9         return 'fdgsd'
```

In [175]:

```
1 t1 = test(4,5,6)
```

In [176]:

```
1 print(t1)
```

```
fdgsd
```

In [177]:

```
1 # Overriding
2
3 class test:
4     def __init__(self,a,b,c):
5         self.a = a
6         self.b = b
7         self.c = c
8
9     def __str__(self):
10        return 'fdgsd'
11
12     def class_fun(self):
13         print('this is a print from class_fun')
14
15 class test1(test):
16     def class_fun(self) :
17         print('fsfsafas overriding method from test1')
18
```

In [178]:

```
1 test1_obj = test1(1,2,3)
2 test_obj = test(4,5,6)
```

In [180]:

```
1 test1_obj.class_fun()
```

fsfsafas overriding method from test1

In [181]:

```
1 test_obj.class_fun()
```

this is a print from class_fun

In []:

```
1
```

In [3]:

```
1 class xyz :  
2     def __init__(self , a , b,c) :  
3         self.a = a  
4         self.b = b  
5         self.c = c  
6  
7     def test(self) :  
8         print("this is my first test metond of xyz class ")  
9  
10    def test1(self) :  
11        print("this is a test1 meth of xyz class ")  
12  
13    def test2(self) :  
14        print("this is my test2 meth of xyz class ")  
15  
16
```

In [5]:

```
1 p = xyz(1,2,3)
```

In [6]:

```
1 p.test1()
```

this is a test1 meth of xyz class

In []:

```
1
```

In [12]:

```
1 class xyz1(xyz) :  
2     pass
```

In [14]:

```
1 q = xyz1(3,4,5)
```

In [15]:

```
1 q.test1()
```

this is a test1 meth of xyz class

In [16]:

```
1 class xyz1(xyz) :  
2     def test(self) :  
3         print("this is a test meth aviable in xyz1")  
4  
5  
6
```

In [17]:

```
1 g = xyz1(4,5,6)
```

In [18]:

```
1 g.test()
```

this is a test meth aviable in xyz1

In [51]:

```
1 class xyz:  
2     def __init__(self, a,b,c) :  
3         self.a= a  
4         self.b= b  
5         self.c= c  
6  
7  
8     def test(self) :  
9         print("this is a meth of xyz class ")  
10  
11 class xyz1:  
12  
13     def __init__(self , p,q,v) :  
14         self.p = p  
15         self.v = v  
16         self.q = q  
17  
18     def test1(self) :  
19         print("this is a meth form class xyz1")  
20  
21 class child(xyz1,xyz):  
22     def __init__(self , *args ,**kwargs):  
23         xyz.__init__(self,*args)  
24         xyz1.__init__(self,**kwargs)  
25
```

In [52]:

```
1 n = child(4,5,6,p = 3,q = 4,v = 5)
```

In [53]:

```
1 n.p
```

Out[53]:

In [54]:

```
1 n.q
```

Out[54]:

4

In [55]:

```
1 n.a
```

Out[55]:

4

In [56]:

```
1 n.test1()
```

this is a meth form class xyz1

In [57]:

```
1 n.test()
```

this is a meth of xyz class

In [62]:

```
1 class xyz:
2     def __init__(self, a,b,c) :
3         self.a= a
4         self.b= b
5         self.c= c
6
7
8     def test(self) :
9         print("this is a meth of xyz class ")
10
11 class xyz1(xyz):
12
13     def test1(self) :
14         print("this is a meth form class xyz1")
15
16 class xyz2(xyz1) :
17     def test2(self) :
18         print("this is a meth form class 2")
19
20 class xyz3(xyz2):
21     def test3(self) :
22         print("this is my meth form class xyz3")
```

In [61]:

```
1 v = xyz2(5,6,7)
```

In [64]:

```
1 m = xyz3(4,5,6)
```

In []:

```
1 q1 - create a file class for reading data from respective file with a method name re
2 q2 - try to inherit read and write method form parent class to child class to perf
```

In []:

```
1 import logging
2
3 logging.basicConfig(filename="list.log",level=logging.DEBUG,format"%(asctime)s %(le
4
5
6 class File:
7     def __init__(self,filename):
8         logging.info("Creating file instance varibale")
9         self.file = filename
10
11     def read(self):
12         logging.info("executing read method")
13         try:
14             with open(f"{self.file}.txt",'r') as f:
15                 data = f.read()
16                 return data
17         except FileNotFoundError as e:
18             logging.error("Error occured while opening the file ")
19             logging.exception(f"Error is {e}")
20
21     def write(self,data):
22         logging.info("executing write method")
23         try:
24             with open(f"{self.file}.txt",'w') as f:
25                 data = f.write(data)
26         except FileNotFoundError as e:
27             logging.error("Something is going wrong")
28
29 obj = File("abc")
30
```

In [70]:

```
1 class readFile:
2     def __init__(self,fileObj, lines):
3         self.file = fileObj
4         self.lines = lines
5     def readFile(self):
6         file2=open(self.file, 'r')
7         line=file2.readline()
8         while(line!=""):
9             print(line)
10            line=file2.readline()
11
12     def writeFile(self):
13         content_write = open(self.file, 'w')
14         content_write.writelines(lines)
15         content_write.close()
16
17 class readFile1(readFile):
18     def readfile1(self):
19         print("inheritance of readFile class")
20
21 file1 = r'test1.txt'
22 lines = 'Inserted Lines .....'
23
24 read_write = readFile1(file1,lines)
25 read_write.writeFile()
26 read_write.readFile()
27
```

Inserted Lines

In []:

```
1 class fileopeartion:
2     def __init__(self,filepath):
3         self.filepath = filepath
4     def read(self):
5         try:
6             f = open(self.filepath, "r+")
7             print(f.read())
8         except Exception as e:
9             print(e)
10            f.close()
11     def write(self,writings=""):
12         try:
13             f = open(self.filepath, "a")
14         except Exception as e:
15             print(e)
16             f.write(writings)
17             f.close()
18 class child(fileopeartion):
19     pass
20
```

In []:

```
1 class file_read(object):
2     def __init__(self,filename) -> None:
3         self.file = filename
4
5 class read(file_read):
6     def read(self):
7         try :
8             open_file = open(self.file, "r")
9             self.read_file = open_file.read()
10            open_file.close()
11            return self.read_file
12        except Exception as e :
13            return e
14
15 class write(file_read) :
16     def write(self,*args):
17         try :
18             open_file = open(self.file, "w")
19             for ele in args:
20                 open_file.write(ele)
21             return True
22         except Exception as e :
23             return e
24
```

In []:

```
1 class filetask:
2     def __init__(self, filename,filetype):
3         self.filename=filename
4         self.filetype=filetype
5
6     def file_write(self):
7         f=open(self.filename, 'w')
8         f.write('this is myc class task')
9         f.close()
10
11    def file_read(self):
12        f=open(self.filename, 'r')
13        print(f.read())
14        f.close()
15
16 class child(ftable):
17     pass
18
19 test1=filetask('testingclass','.txt')
20 test2=child('testingchild','.txt')
21
```

In [74]:

```
1 class test :  
2     def __init__(self , a,b,c) :  
3         self.a = a  
4         self.b = b  
5         self.c = c  
6  
7 class test1(test) :  
8     pass  
9 u = test(4,5,6)
```

In [75]:

```
1 v = test1(3,4,5)
```

In [107]:

```
1 class test :  
2     def __init__(self ) :  
3         self.__a = 4  
4  
5  
6 class test1(test) :  
7     def __init__(self):  
8         self.__a = 7  
9  
10  
11 u = test()
```

In [108]:

```
1 u.__a
```

```
-----  
-  
AttributeError                                     Traceback (most recent call last)  
t)  
<ipython-input-108-45f5c7f66ea7> in <module>  
----> 1 u.__a
```

AttributeError: 'test' object has no attribute '__a'

In [109]:

```
1 v = test1()
```

In [110]:

```
1 v.__a
```

```
-  
AttributeError  
t)  
<ipython-input-110-7aed92cddc9e> in <module>  
----> 1 v.__a
```

Traceback (most recent call last)

```
AttributeError: 'test1' object has no attribute '__a'
```

In [120]:

```
1 class test:  
2     def __init__(self,a,b,c):  
3         self._a = a  
4         self._b= b  
5         self.c = c  
6  
7 class test1(test) :  
8     pass  
9  
10 v = test(4,5,6)
```

In [121]:

```
1 v.c
```

Out[121]:

4

In [133]:

```
1 u._test__b
```

Out[133]:

4

In [131]:

```
1 v._test__b
```

Out[131]:

5

In [123]:

```
1 v.c
```

Out[123]:

6

In [125]:

```
1 u = test1(3,4,7)
```

In [126]:

```
1 u.c
```

Out[126]:

7

In [128]:

```
1 u._a
```

Out[128]:

3

In [130]:

```
1 u._test1__b
```

```
-  
AttributeError Traceback (most recent call last)  
t)  
<ipython-input-130-28b27c9c3546> in <module>  
----> 1 u._test1__b  
  
AttributeError: 'test1' object has no attribute '_test1__b'
```

In [167]:

```
1 class bonousclaculator:  
2     def __init__(self, empid , emprating) :  
3         self.empid = empid  
4         self.emprating = emprating  
5         self._empmail = "fdsfsfssf@gmail.com"  
6         self.__bonusforratingA = "70%"  
7         self.__bonusforraringB = "60%"  
8         self.__bonusforratinC = "40%"  
9  
10    def bonuscalculator(self) :  
11        if self.emprating == "A":  
12            bonus = self.__bonusforratingA  
13            return bonus  
14        elif self.emprating == "B" :  
15            bonus = self.__bonusforraringB  
16            return bonus  
17        else :  
18            bonus = self.__bonusforratinC  
19            return bonus
```

In [168]:

```
1 emp1 = bonousclaculator(101 , "A")
2 emp2 = bonousclaculator(102 , "B")
3 emp3 = bonousclaculator(103, "C")
```

In [169]:

```
1 emp1.bonuscalculator()
```

Out[169]:

'70%'

In [170]:

```
1 emp2.bonuscalculator()
```

Out[170]:

'60%'

In [171]:

```
1 emp3.bonuscalculator()
```

Out[171]:

'40%'

In [172]:

```
1 emp1.empid = 104
```

In [173]:

```
1 emp1.empid
```

Out[173]:

104

In [174]:

```
1 emp1.emprating= "B"
```

In [175]:

```
1 emp1.bonuscalculator()
```

Out[175]:

'60%'

In [176]:

```
1 emp1.__bonusforraringB = "90%"
```

In [177]:

```
1 emp1.bonuscalculator()
```

Out[177]:

```
'60%'
```

In [178]:

```
1 emp2.emprating
```

Out[178]:

```
'B'
```

In [179]:

```
1 emp2.bonuscalculator()
```

Out[179]:

```
'60%'
```

In [180]:

```
1 emp2= "100%"
```

In [181]:

```
1 emp2.bonuscalculator()
```

```
-----
-
AttributeError                                Traceback (most recent call last)
t)
<ipython-input-181-ea154c101e92> in <module>
----> 1 emp2.bonuscalculator()

AttributeError: 'str' object has no attribute 'bonuscalculator'
```

In [182]:

```
1 emp2.empid = 34543543
```

-
AttributeError
t)

Traceback (most recent call las

t)
<ipython-input-182-e7ea127b34b4> in <module>
----> 1 emp2.empid = 34543543

AttributeError: 'str' object has no attribute 'empid'

In [183]:

```
1 emp2.empid
```

-
AttributeError
t)

Traceback (most recent call las

t)
<ipython-input-183-92058fc93785> in <module>
----> 1 emp2.empid

AttributeError: 'str' object has no attribute 'empid'

In [184]:

```
1 emp2.__bonusforraringB = "fsadfsafs"
```

-
AttributeError
t)

Traceback (most recent call las

t)
<ipython-input-184-ea21f58f68f8> in <module>
----> 1 emp2.__bonusforraringB = "fsadfsafs"

AttributeError: 'str' object has no attribute '__bonusforraringB'

In [161]:

```
1 emp2._bonousclaculator__bonusforraringB
```

Out[161]:

'60%'

In [165]:

```
1 emp2._bonousclaculator__bonusforraringB = "435456"
```

In [166]:

```
1 emp2._bonousclaculator__bonusforraringB
```

Out[166]:

```
'435456'
```

In [185]:

```
1 emp1._empmail
```

Out[185]:

```
'fdsfsfsssf@gmail.com'
```

In [186]:

```
1 emp1._empmail = "sudhanshu@gmail.com"
```

In [187]:

```
1 emp1._empmail
```

Out[187]:

```
'sudhanshu@gmail.com'
```

In [189]:

```
1 class xyz():
2     def __init__(self,a,b,c):
3         self.a=a
4         self.b=b
5         self.c=c
6     def test (self):
7         print ("machine")
8
9 class xyz1(xyz):
10    def __init__(self,p,q,r):
11        self.p=p
12        self.q=q
13        self.r=r
14    def test1(self):
15        print ("learning")
16 class xyz2(xyz1):
17    def test2(self):
18        print ("ineuron")
19
```

In [190]:

```
1 v = xyz2(3,4,5)
```

In []:

```
1 v.test
```

In [195]:

```
1 def checkPrime():
2     ''' Checks if a number is Prime. Keeps on asking
3         for an input in correct form, >1 & an integer
4         '''
5     while True:
6         try:
7             num=int(input('Enter an integer'))
8             if num>1:
9                 for i in range(2, num):
10
11                     if (num % i)==0:
12                         pss
13                         #print(num, ' is a not prime number')
14                         #break
15
16                     else:
17                         print(num, ' is a prime number')
18                         break
19
20             break
21
22
23         else:
24             print('Please enter a number >1')
25     except:
26         print('Please enter a valid input, an integer')
27 checkPrime()
28
```

```
Enter an integer4564
Please enter a valid input, an integer
Enter an integer456
Please enter a valid input, an integer
Enter an integer435
435  is a prime number
```

In [196]:

```
1 def largest_arr(*args):
2     lagest_list_iter=0
3     lagest_list=0
4     l=[]
5     for i in args:
6         if type(i)==list or type(i)==tuple or type(i)==set:
7             for j in i:
8                 l.append(j)
9                 lagest_list_iter=max(l)
10            return lagest_list_iter
11
12        else:
13            for i in args:
14                l.append(i)
15                lagest_list=max(l)
16            return lagest_list
17
```

In [197]:

```
1 a = 50
2 iter(a)
```

```
-----
-
TypeError                                     Traceback (most recent call last)
t)
<ipython-input-197-040dd2cda7dc> in <module>
      1 a = 50
----> 2 iter(a)

TypeError: 'int' object is not iterable
```

In [198]:

```
1 b = [4,5,6]
```

In [200]:

```
1 type(iter(b))
```

Out[200]:

list_iterator

In [204]:

```
1 if list in [list, tuple, set]:
2     print("do someting ")
```

do someting

In [208]:

```
1 import array as arr
```

In [223]:

```
1 a = arr.array('f', [1,2,3,4])
2 a
```

Out[223]:

```
array('f', [1.0, 2.0, 3.0, 4.0])
```

In [213]:

```
1 import numpy as np
2 np.array([1,2,3,4])
```

Out[213]:

```
array([1, 2, 3, 4])
```

In [214]:

```
1 np.asarray([3,5,67,67])
```

Out[214]:

```
array([ 3,  5, 67, 67])
```

In [215]:

```
1 np.asarray([34,6565,76,7,6])
```

Out[215]:

```
array([ 34, 6565,    76,      7,      6])
```

In [219]:

```
1 for i in a:
2     print(i)
```

```
1.0
2.0
3.0
4.0
```

In [224]:

```
1 class parent:
2     def __init__(self, p, q, r):
3         self.p = p
4         self._q = q
5         self.__r = r
6
7 class child(parent):
8     pass
9
10 c1 = child(4,5,6)
11
12 # How is this possible???
13 c1.z = 435
14 c1.z
```

Out[224]:

435

In [225]:

```
1 a = 10
```

In [226]:

```
1 c2 = child(4,5,6)
```

In [227]:

```
1 c2.z
```

```
-  
-  
AttributeError                                     Traceback (most recent call last  
t)  
<ipython-input-227-60f30b911812> in <module>  
----> 1 c2.z
```

AttributeError: 'child' object has no attribute 'z'

In [233]:

```
1 class bonuscalculator:
2     def __init__(self,empid, emprating):
3         self.empid = empid
4         self.emprating = emprating
5         self.__bonusforratingA = "70%"
6         self.__bonusforratingB = "60%"
7         self.__bonusforratingC = "40%"
8
9     def bonuscalc(self):
10        if self.emprating == "A":
11            bonus = self.__bonusforratingA
12            return bonus
13        elif self.emprating == "B":
14            bonus = self.__bonusforratingB
15            return bonus
16        else :
17            bonus = self.__bonusforratingC
18            return bonus ,self.empid
19
20 emp1 = bonuscalculator(101, "A")
21 emp2 = bonuscalculator(102, "B")
22 emp3 = bonuscalculator(103, "C")
23 emp1.bonuscalc()
24 emp2.bonuscalc()
25
26
```

Out[233]:

'60%'

In [234]:

```
1 emp3.bonuscalc()
```

Out[234]:

('40%', 103)

In [236]:

```
1 class bonuscalculator:
2     def __init__(self,empid,emprating):
3         self.empid = empid
4         self.emprating = emprating
5         self._empmail = "abcd@gmail.com"
6         self.__bonusforratingA = "70%"
7         self.__bonusforratingB = "60%"
8         self.__bonusforratingC = "40%"
9
10    def bonuscalculator(self):
11        if self.emprating == "A":
12            bonus = self.__bonusforratingA
13            return bonus ,self.empid
14        elif self.emprating == "B":
15            bonus = self.__bonusforratingB
16            return bonus,self.empid
17        else:
18            bonus = self.__bonusforratingC
19            return bonus,self.empid
20 emp1 = bonuscalculator(101,"A")
21 emp2 = bonuscalculator(102,"B")
22 emp3 = bonuscalculator(103,"C")
23 emp1.bonuscalculator()
24 emp2.bonuscalculator()
25
26
```

Out[236]:

```
('60%', 102)
```

In [243]:

```
1 class Xyz:
2     objCount = 0
3     def __init__(self, a, b, c):
4         self.a = a
5         self.b = b
6         self.c = c
7         Xyz.objCount += 1
8
9     def test(self):
10        print("Inside test method")
11
12    def test1(self):
13        print("Inside the test1 method")
14
15    def test2(self):
16        print("Inside the test2 method")
17
18 p = Xyz(1,2,3)
19 p1 = Xyz(2,3,4)
20 p1.objCount
```

Out[243]:

```
2
```

In [244]:

```
1 p2 = Xyz(2,3,4)
```

In [242]:

```
1 p1.objCount  
2
```

Out[242]:

3

In []:

```
1
```

In [5]:

```
1 import this
```

The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

In [6]:

```
1 print('this is my first program')
```

this is my first program

In [7]:

```
1 3/2
```

Out[7]:

1.5

In [9]:

```
1 1+2
```

Out[9]:

3

In [11]:

```
1 a = 10
```

In [12]:

```
1 a
```

Out[12]:

10

In [13]:

```
1 type(a)
```

Out[13]:

int

In [14]:

```
1 b = 45.6
2
```

In [15]:

```
1 type(b)
```

Out[15]:

float

In [16]:

```
1 c = 'sudh'
```

In [17]:

```
1 type(c)
2
```

Out[17]:

str

In [18]:

```
1 d = 5+6j
```

In [19]:

```
1 d
```

Out[19]:

(5+6j)

In [20]:

```
1 type(d)
```

Out[20]:

complex

In [21]:

```
1 n = 'ssssssdfaslkhfasldfjalsfjak'
```

In [22]:

```
1 g = True
```

In [23]:

```
1 type(g)
```

Out[23]:

bool

In [24]:

```
1 *a = 45
```

```
File "C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_25588\1810212927.py", line 1
    *a = 45
^
```

SyntaxError: starred assignment target must be in a list or tuple

In [25]:

```
1 _a = 45
```

In [26]:

```
1 _a
```

Out[26]:

45

In [27]:

```
1 a = 35
```

In [28]:

```
1 a
```

Out[28]:

35

In [29]:

```
1 a = 5353
2 b = 'sudh'
3 c= 6+7j
4 d = True
5 e =547.7
```

In [30]:

```
1 a
```

Out[30]:

5353

In [31]:

```
1 a1= 45
```

In [32]:

```
1 a+a1
```

Out[32]:

5398

In [33]:

```
1 a,b,c,d,e = 243, 'sudh', 4+6j, False, 43.56
```

In [34]:

```
1 a
```

Out[34]:

243

In [35]:

```
1 b
```

Out[35]:

'sudh'

In [36]:

```
1 c
```

Out[36]:

(4+6j)

In [37]:

```
1 d
```

Out[37]:

False

In [38]:

```
1 e
```

Out[38]:

43.56

In [39]:

```
1 type(c)
```

Out[39]:

complex

In [40]:

```
1 c
```

Out[40]:

(4+6j)

In [41]:

```
1 c.imag
```

Out[41]:

6.0

In [42]:

```
1 c.real
```

Out[42]:

4.0

In [43]:

```
1 | b
```

Out[43]:

```
'sudh'
```

In [44]:

```
1 | a = "sudh"
```

In [45]:

```
1 | a + 4
```

```
-----
-
TypeError                                 Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_25588\2703434432.py in <module>
      1 a + 4
-----
```

TypeError: can only concatenate str (not "int") to str

In [46]:

```
1 | a + '4'
```

Out[46]:

```
'sudh4'
```

In [47]:

```
1 | a + str(4)
```

Out[47]:

```
'sudh4'
```

In [48]:

```
1 | True + True
```

Out[48]:

```
2
```

In [49]:

```
1 | True - False
```

Out[49]:

```
1
```

In [50]:

```
1 1+ True
```

Out[50]:

2

In [51]:

```
1 input()
```

232

Out[51]:

'232'

In [56]:

```
1 a = input()
```

76

In [57]:

```
1 a
```

Out[57]:

'76'

In [58]:

```
1 b = 45
```

In [61]:

```
1 int(a)+(b)
```

Out[61]:

121

In [62]:

```
1 id(a)
```

Out[62]:

1536834808752

In [63]:

```
1 id(b)
```

Out[63]:

1536720465584

In []:

1

In [2]:

```
1 #!pip install mysql-connector-python
```

In [1]:

```
1 import mysql.connector as conn
```

In [2]:

```
1 mydb = conn.connect(host = "localhost" , user = 'root' , passwd ="Rakshanda@26")
```

In [3]:

```
1 mydb
```

Out[3]:

```
<mysql.connector.connection_cext.CMySQLConnection at 0x265355de400>
```

In [4]:

```
1 # its a pointer we need to connect it
2 cursor = mydb.cursor(buffered=True)
```

In [5]:

```
1 cursor.execute('show databases')
```

In [6]:

```
1 cursor.fetchall()
```

Out[6]:

```
[('dress_data',),
 ('ineuron',),
 ('ineuron_fsda',),
 ('ineuron_partition',),
 ('information_schema',),
 ('key_prim',),
 ('mysql',),
 ('online_retail',),
 ('operation',),
 ('performance_schema',),
 ('restaurant',),
 ('sakila',),
 ('sales',),
 ('sys',),
 ('test',),
 ('testraksha',),
 ('uk_project',),
 ('win_fun',),
 ('world',)]
```

In [7]:

```
1 cursor.execute('create database if not exists testraksha')
```

In [8]:

```
1 cursor.execute('create table testraksha.capgemini(studentid INT(10) , firstname varc
```



```
-----
-
MySQLInterfaceError                                     Traceback (most recent call las
t)
~\anaconda3\lib\site-packages\mysql\connector\connection_cext.py in cmd_qu
ery(self, query, raw, buffered, raw_as_string)
    534         query = query.encode('utf-8')
--> 535     self._cmysql.query(query,
    536                             raw=raw, buffered=buffered,
```

MySQLInterfaceError: Table 'capgemini' already exists

During handling of the above exception, another exception occurred:

```
ProgrammingError                                     Traceback (most recent call las
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_16320/2782354578.py in <mod
ule>
----> 1 cursor.execute('create table testraksha.capgemini(studentid INT(1
0) , firstname varchar(30) , lastname varchar(30), regid INT(10), classnam
e varchar(30))')

~\anaconda3\lib\site-packages\mysql\connector\cursor_cext.py in execute(se
lf, operation, params, multi)
    267
    268     try:
--> 269         result = self._cnx.cmd_query(stmt, raw=self._raw,
    270                                         buffered=self._buffered,
    271                                         raw_as_string=self._raw_a
s_string)

~\anaconda3\lib\site-packages\mysql\connector\connection_cext.py in cmd_qu
ery(self, query, raw, buffered, raw_as_string)
    538             query_attrs=self._query_attrs)
    539     except MySQLInterfaceError as exc:
--> 540         raise errors.get_mysql_exception(exc errno, msg=exc.ms
g,
    541                                         sqlstate=exc.sqlstat
e)
    542     except AttributeError:
```

ProgrammingError: 1050 (42S01): Table 'capgemini' already exists

In [9]:

```
1 cursor.execute('use testraksha')
```

In [10]:

```
1 cursor.execute('show tables')
```

In [93]:

```
1 cursor.fetchall()
```

Out[93]:

```
[('capgemini',), ('glassdata45',)]
```

In [104]:

```
1 cursor.execute('insert into testraksha.capgemini values(1, "sudh" , "kumar" , 41 , "')
```

In [95]:

```
1 mydb.commit()
```

In [13]:

```
1 cursor.execute('select * from testraksha.capgemini')
```

In [14]:

```
1 cursor.fetchall()
```

Out[14]:

```
[(1, 'sudh', 'kumar', 41, 'FSDS'),  
(1, 'sudh', 'kumar', 41, 'FSDS'),  
(1, 'sudh', 'kumar', 41, 'FSDS')]
```

In [98]:

```
1 # INSERTING BULK DATA
```

In [99]:

```
1 cursor.execute('create table testraksha.glassdata45(col1 INT(10), col2 float(10,5),
```

```
-  
MySQLInterfaceError                                     Traceback (most recent call last)  
t)  
~\anaconda3\lib\site-packages\mysql\connector\connection_cext.py in cmd_qu  
ery(self, query, raw, buffered, raw_as_string)  
    534         query = query.encode('utf-8')  
--> 535         self._cmysql.query(query,  
    536                         raw=raw, buffered=buffered,
```

MySQLInterfaceError: Table 'glassdata45' already exists

During handling of the above exception, another exception occurred:

```
ProgrammingError                                     Traceback (most recent call las  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_23076\2176826424.py in <mod  
ule>  
----> 1 cursor.execute('create table testraksha.glassdata45(col1 INT(10),  
col2 float(10,5), col3 float(10,5) , col4 float(10,5) , col5 float(10,5),  
col6 float(10,5), col7 float(10,5), col8 float(10,5), col9 float(10,5) , c  
ol10 float (10,5), col11 Int(10))')
```

```
~\anaconda3\lib\site-packages\mysql\connector\cursor_cext.py in execute(se  
lf, operation, params, multi)  
    267  
    268         try:  
--> 269             result = self._cnx.cmd_query(stmt, raw=self._raw,  
    270                                         buffered=self._buffered,  
    271                                         raw_as_string=self._raw_a  
s_string)
```

```
~\anaconda3\lib\site-packages\mysql\connector\connection_cext.py in cmd_qu  
ery(self, query, raw, buffered, raw_as_string)  
    538         query_attrs=self._query_attrs)  
    539     except MySQLInterfaceError as exc:  
--> 540         raise errors.get_mysql_exception(exc errno, msg=exc.ms  
g,  
    541                                         sqlstate=exc.sqlstat  
e)  
    542     except AttributeError:
```

ProgrammingError: 1050 (42S01): Table 'glassdata45' already exists

In [100]:

```
1 cursor.execute('show tables')
```

In [30]:

```
1 cursor.fetchall()
```

Out[30]:

```
[('capgemini',), ('glassdata45',)]
```

In [38]:

```
1 file = open('glass (1).data' , 'r+')
```

In [40]:

```
1 #file.read()
```

In [51]:

```
1 import csv
2 with open('glass (1).data', 'r') as f:
3     glass_data = csv.reader(f, delimiter ='\n')
4     for i in glass_data:
5         print(i)
```

```
['1,1.52101,13.64,4.49,1.10,71.78,0.06,8.75,0.00,0.00,1']
['2,1.51761,13.89,3.60,1.36,72.73,0.48,7.83,0.00,0.00,1']
['3,1.51618,13.53,3.55,1.54,72.99,0.39,7.78,0.00,0.00,1']
['4,1.51766,13.21,3.69,1.29,72.61,0.57,8.22,0.00,0.00,1']
['5,1.51742,13.27,3.62,1.24,73.08,0.55,8.07,0.00,0.00,1']
['6,1.51596,12.79,3.61,1.62,72.97,0.64,8.07,0.00,0.26,1']
['7,1.51743,13.30,3.60,1.14,73.09,0.58,8.17,0.00,0.00,1']
['8,1.51756,13.15,3.61,1.05,73.24,0.57,8.24,0.00,0.00,1']
['9,1.51918,14.04,3.58,1.37,72.08,0.56,8.30,0.00,0.00,1']
['10,1.51755,13.00,3.60,1.36,72.99,0.57,8.40,0.00,0.11,1']
['11,1.51571,12.72,3.46,1.56,73.20,0.67,8.09,0.00,0.24,1']
['12,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1']
['13,1.51589,12.88,3.43,1.40,73.28,0.69,8.05,0.00,0.24,1']
['14,1.51748,12.86,3.56,1.27,73.21,0.54,8.38,0.00,0.17,1']
['15,1.51763,12.61,3.59,1.31,73.29,0.58,8.50,0.00,0.00,1']
['16,1.51761,12.81,3.54,1.23,73.24,0.58,8.39,0.00,0.00,1']
['17,1.51784,12.68,3.67,1.16,73.11,0.61,8.70,0.00,0.00,1']
['18,1.52196,14.36,3.85,0.89,71.36,0.15,9.15,0.00,0.00,1']
['19,1.51911,13.90,3.73,1.18,72.12,0.06,8.89,0.00,0.00,1']
['20,1.51775,12.82,3.51,1.60,72.72,0.54,8.11,0.00,0.07,1']
```

In [62]:

```
1 import csv
2 with open('glass (1).data', 'r') as f:
3     glass_data = csv.reader(f, delimiter ='\n')
4     for i in glass_data:
5         print(i[0])
```

```
1,1.52101,13.64,4.49,1.10,71.78,0.06,8.75,0.00,0.00,1
2,1.51761,13.89,3.60,1.36,72.73,0.48,7.83,0.00,0.00,1
3,1.51618,13.53,3.55,1.54,72.99,0.39,7.78,0.00,0.00,1
4,1.51766,13.21,3.69,1.29,72.61,0.57,8.22,0.00,0.00,1
5,1.51742,13.27,3.62,1.24,73.08,0.55,8.07,0.00,0.00,1
6,1.51596,12.79,3.61,1.62,72.97,0.64,8.07,0.00,0.26,1
7,1.51743,13.30,3.60,1.14,73.09,0.58,8.17,0.00,0.00,1
8,1.51756,13.15,3.61,1.05,73.24,0.57,8.24,0.00,0.00,1
9,1.51918,14.04,3.58,1.37,72.08,0.56,8.30,0.00,0.00,1
10,1.51755,13.00,3.60,1.36,72.99,0.57,8.40,0.00,0.11,1
11,1.51571,12.72,3.46,1.56,73.20,0.67,8.09,0.00,0.24,1
12,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
13,1.51589,12.88,3.43,1.40,73.28,0.69,8.05,0.00,0.24,1
14,1.51748,12.86,3.56,1.27,73.21,0.54,8.38,0.00,0.17,1
15,1.51763,12.61,3.59,1.31,73.29,0.58,8.50,0.00,0.00,1
16,1.51761,12.81,3.54,1.23,73.24,0.58,8.39,0.00,0.00,1
17,1.51784,12.68,3.67,1.16,73.11,0.61,8.70,0.00,0.00,1
18,1.52196,14.36,3.85,0.89,71.36,0.15,9.15,0.00,0.00,1
19,1.51911,13.90,3.73,1.18,72.12,0.06,8.89,0.00,0.00,1
20,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
21,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
22,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
23,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
24,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
25,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
26,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
27,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
28,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
29,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
30,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
31,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
32,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
33,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
34,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
35,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
36,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
37,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
38,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
39,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
40,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
41,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
42,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
43,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
44,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
45,1.51763,12.80,3.66,1.27,73.01,0.60,8.56,0.00,0.00,1
```

In [52]:

```
1 cursor.execute('insert into testraksha.glassdata45 values(1,1.52101,13.64,4.49,1.10,
```

In [53]:

```
1 mydb.commit()
```

In [54]:

```
1 cursor.execute('select * from testraksha.glassdata45 ')
```

In [55]:

```
1 cursor.fetchall()
```

Out[55]:

```
[(1, 1.52101, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1)]
```

In [80]:

```
1 import csv
2 with open('glass (1).data', 'r') as f:
3     glass_data = csv.reader(f, delimiter ='\n')
4     for i in glass_data:
5         #print(type(i[0]))
6         #print(f'insert into glassdata45 values({str(i[0])})')
7         cursor.execute(f'insert into testraksha.glassdata45 values({str(i[0])})')
8         #cursor.execute('insert into testraksha.glassdata45 values(i[0]) ')
9 mydb.commit()
10
```

In [81]:

```
1 cursor.execute('select * from testraksha.glassdata45 ')
```

In [82]:

```
1 cursor.fetchall()
```

Out[82]:

```
((1, 1.52101, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 0.0, 1),
(1, 1.52101, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 0.0, 1),
(2, 1.51761, 13.89, 3.6, 1.36, 72.73, 0.48, 7.83, 0.0, 0.0, 0.0, 1),
(3, 1.51618, 13.53, 3.55, 1.54, 72.99, 0.39, 7.78, 0.0, 0.0, 0.0, 1),
(4, 1.51766, 13.21, 3.69, 1.29, 72.61, 0.57, 8.22, 0.0, 0.0, 0.0, 1),
(5, 1.51742, 13.27, 3.62, 1.24, 73.08, 0.55, 8.07, 0.0, 0.0, 0.0, 1),
(6, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 0.0, 1),
(7, 1.51743, 13.3, 3.6, 1.14, 73.09, 0.58, 8.17, 0.0, 0.0, 0.0, 1),
(8, 1.51756, 13.15, 3.61, 1.05, 73.24, 0.57, 8.24, 0.0, 0.0, 0.0, 1),
(9, 1.51918, 14.04, 3.58, 1.37, 72.08, 0.56, 8.3, 0.0, 0.0, 0.0, 1),
(10, 1.51755, 13.0, 3.6, 1.36, 72.99, 0.57, 8.4, 0.0, 0.11, 0.0, 1),
(11, 1.51571, 12.72, 3.46, 1.56, 73.2, 0.67, 8.09, 0.0, 0.24, 0.0, 1),
(12, 1.51763, 12.8, 3.66, 1.27, 73.01, 0.6, 8.56, 0.0, 0.0, 0.0, 1),
(13, 1.51589, 12.88, 3.43, 1.4, 73.28, 0.69, 8.05, 0.0, 0.24, 0.0, 1),
(14, 1.51748, 12.86, 3.56, 1.27, 73.21, 0.54, 8.38, 0.0, 0.17, 0.0, 1),
(15, 1.51763, 12.61, 3.59, 1.31, 73.29, 0.58, 8.5, 0.0, 0.0, 0.0, 1),
(16, 1.51761, 12.81, 3.54, 1.23, 73.24, 0.58, 8.39, 0.0, 0.0, 0.0, 1),
(17, 1.51784, 12.68, 3.67, 1.16, 73.11, 0.61, 8.7, 0.0, 0.0, 0.0, 1)).
```

In [83]:

```
1 import pandas as pd
```

In [84]:

```
1 pd.read_sql('select * from testraksha.glassdata45', mydb)
```

C:\Users\RAKSHANDA\anaconda3\lib\site-packages\pandas\io\sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engine/connection) or database string URI or sqlite3 DBAPI2 connection other DBAPI2 objects are not tested, please consider using SQLAlchemy

```
    warnings.warn(
```

Out[84]:

	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11
0	1	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0	1
1	1	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0	1
2	2	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
3	3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
4	4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
...
1494	210	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
1495	211	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
1496	212	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
1497	213	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
1498	214	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

1499 rows × 11 columns

In [86]:

```
1 pd.read_sql('show databases', mydb)
```

```
C:\Users\RAKSHANDA\anaconda3\lib\site-packages\pandas\io\sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engine/connection) or database string URI or sqlite3 DBAPI2 connection other DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

Out[86]:

	Database
0	dress_data
1	ineuron
2	ineuron_fsda
3	ineuron_partition
4	information_schema
5	key_prim
6	mysql
7	online_retail
8	operation
9	performance_schema
10	restaurant
11	sakila
12	sales
13	sys
14	test
15	testraksha
16	uk_project
17	win_fun
18	world

In [90]:

```
1 pd.read_sql('use testraksha', mydb)
```

```
-  
TypeError  
Traceback (most recent call last)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_23076\111347379.py in <module>  
----> 1 pd.read_sql('use testraksha', mydb)  
  
~\anaconda3\lib\site-packages\pandas\io\sql.py in read_sql(sql, con, index_col, coerce_float, params, parse_dates, columns, chunksize)  
    564  
    565        if isinstance(pandas_sql, SQLiteDatabase):  
--> 566            return pandas_sql.read_query(  
    567                sql,  
    568                index_col=index_col,  
  
~\anaconda3\lib\site-packages\pandas\io\sql.py in read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize, dtype)  
2079            args = _convert_params(sql, params)  
2080            cursor = self.execute(*args)  
-> 2081            columns = [col_desc[0] for col_desc in cursor.description]  
2082  
2083            if chunksize is not None:  
  
TypeError: 'NoneType' object is not iterable
```

In [15]:

```
1 pwd
```

Out[15]:

```
'D:\\Python_Basics\\SQL_Basics'
```

In []:

```
1
```

In [1]:

```
1 import mysql.connector as conn
```

In [2]:

```
1 mydb = conn.connect(host = 'localhost' , user = 'root' , passwd = 'Rakshanda@26')
```

In [3]:

```
1 cursor = mydb.cursor()
```

In [4]:

```
1 cursor.execute('show databases')
```

In [5]:

```
1 cursor.fetchall()
```

Out[5]:

```
[('dress_data',),
 ('ineuron',),
 ('ineuron_fsda',),
 ('ineuron_partition',),
 ('information_schema',),
 ('key_prim',),
 ('mysql',),
 ('online_retail',),
 ('operation',),
 ('performance_schema',),
 ('restaurant',),
 ('sakila',),
 ('sales',),
 ('sys',),
 ('test',),
 ('testraksha',),
 ('uk_project',),
 ('win_fun',),
 ('world',)]
```

In [6]:

```
1 cursor.execute('select * from testraksha.glassdata45')
```

In [7]:

```
1 cursor.fetchall()
```

Out[7]:

```
[(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(2, 1.51761, 13.89, 3.6, 1.36, 72.73, 0.48, 7.83, 0.0, 0.0, 1),  
(3, 1.51618, 13.53, 3.55, 1.54, 72.99, 0.39, 7.78, 0.0, 0.0, 1),  
(5, 1.51742, 13.27, 3.62, 1.24, 73.08, 0.55, 8.07, 0.0, 0.0, 1),  
(6, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1),  
(7, 1.51743, 13.3, 3.6, 1.14, 73.09, 0.58, 8.17, 0.0, 0.0, 1),  
(8, 1.51756, 13.15, 3.61, 1.05, 73.24, 0.57, 8.24, 0.0, 0.0, 1),  
(9, 1.51918, 14.04, 3.58, 1.37, 72.08, 0.56, 8.3, 0.0, 0.0, 1),  
(10, 1.51755, 13.0, 3.6, 1.36, 72.99, 0.57, 8.4, 0.0, 0.11, 1),  
(11, 1.51571, 12.72, 3.46, 1.56, 73.2, 0.67, 8.09, 0.0, 0.24, 1),  
(12, 1.51763, 12.8, 3.66, 1.27, 73.01, 0.6, 8.56, 0.0, 0.0, 1),  
(13, 1.51589, 12.88, 3.43, 1.4, 73.28, 0.69, 8.05, 0.0, 0.24, 1),  
(14, 1.51748, 12.86, 3.56, 1.27, 73.21, 0.54, 8.38, 0.0, 0.17, 1),  
(15, 1.51763, 12.61, 3.59, 1.31, 73.29, 0.58, 8.5, 0.0, 0.0, 1),  
(16, 1.51761, 12.81, 3.54, 1.23, 73.24, 0.58, 8.39, 0.0, 0.0, 1),  
(17, 1.51784, 12.68, 3.67, 1.16, 73.11, 0.61, 8.7, 0.0, 0.0, 1),  
(18, 1.52196, 14.36, 3.85, 0.89, 71.36, 0.15, 9.15, 0.0, 0.0, 1).
```

In [8]:

```
1 cursor.execute('select col1,col5, col10, col1 from testraksha.glassdata45')
```

In [9]:

```
1 cursor.fetchall()
```

Out[9]:

```
[(6, 1.1, 0.0, 6),  
(6, 1.1, 0.0, 6),  
(2, 1.36, 0.0, 2),  
(3, 1.54, 0.0, 3),  
(5, 1.24, 0.0, 5),  
(6, 1.62, 0.26, 6),  
(7, 1.14, 0.0, 7),  
(8, 1.05, 0.0, 8),  
(9, 1.37, 0.0, 9),  
(10, 1.36, 0.11, 10),  
(11, 1.56, 0.24, 11),  
(12, 1.27, 0.0, 12),  
(13, 1.4, 0.24, 13),  
(14, 1.27, 0.17, 14),  
(15, 1.31, 0.0, 15),  
(16, 1.23, 0.0, 16),  
(17, 1.16, 0.0, 17),  
(18, 0.89, 0.0, 18)].
```

In [10]:

```
1 cursor.execute('select * from testraksha.glassdata45 where col1 =4 and col11 = 0')
```

In [11]:

```
1 cursor.fetchall()
```

Out[11]:

```
[]
```

In [12]:

```
1 cursor.execute('select * from testraksha.glassdata45 where col1 =4')
```

In [13]:

```
1 cursor.fetchall()
```

Out[13]:

```
[]
```

In [14]:

```
1 cursor.execute('delete from testraksha.glassdata45 where col1 = 4 ')
```

In [15]:

```
1 mydb.commit()
```

In [16]:

```
1 cursor.execute('select * from testraksha.glassdata45 where col1 =4')
```

In [17]:

```
1 cursor.fetchall()
```

Out[17]:

```
[]
```

In [18]:

```
1 cursor.execute('update testraksha.glassdata45 set col1 = 6 ,col2 = 8 where col1 = 1')
```

In [19]:

```
1 mydb.commit()
```

In [20]:

```
1 cursor.execute('select * from testraksha.glassdata45 where col1 = 6 and col2 = 8')
```

In [21]:

```
1 cursor.fetchall()
```

Out[21]:

```
[(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1)]
```

In [22]:

```
1 cursor.execute('select count(col1) ,col2 from testraksha.glassdata45 group by col1')
```

In [23]:

```
1 cursor.fetchall()
```

Out[23]:

```
[(15, 8.0),  
(7, 1.51761),  
(7, 1.51618),  
(7, 1.51742),  
(7, 1.51743),  
(7, 1.51756),  
(7, 1.51918),  
(7, 1.51755),  
(7, 1.51571),  
(7, 1.51763),  
(7, 1.51589),  
(7, 1.51748),  
(7, 1.51763),  
(7, 1.51761),  
(7, 1.51784),  
(7, 1.52196),  
(7, 1.51911),  
(7, 1.51735).
```

In [24]:

```
1 cursor.execute('select col1 ,col2 from testraksha.glassdata45 group by col1,col2')
```

In [25]:

```
1 cursor.fetchall()
```

Out[25]:

```
[(6, 8.0),  
(2, 1.51761),  
(3, 1.51618),  
(5, 1.51742),  
(6, 1.51596),  
(7, 1.51743),  
(8, 1.51756),  
(9, 1.51918),  
(10, 1.51755),  
(11, 1.51571),  
(12, 1.51763),  
(13, 1.51589),  
(14, 1.51748),  
(15, 1.51763),  
(16, 1.51761),  
(17, 1.51784),  
(18, 1.52196),  
(19, 1.51911).
```

In [26]:

```
1 cursor.execute('select col1, count(col1) , col11 from testraksha.glassdata45 group b
```

In [27]:

```
1 cursor.fetchall()
```

Out[27]:

```
[(6, 15, 1),  
(2, 7, 1),  
(3, 7, 1),  
(5, 7, 1),  
(7, 7, 1),  
(8, 7, 1),  
(9, 7, 1),  
(10, 7, 1),  
(11, 7, 1),  
(12, 7, 1),  
(13, 7, 1),  
(14, 7, 1),  
(15, 7, 1),  
(16, 7, 1),  
(17, 7, 1),  
(18, 7, 1),  
(19, 7, 1),  
(20, 7, 1).
```

In [28]:

```
1 cursor.execute('select * from testraksha.glassdata45 where col1 = 2')
```

In [29]:

```
1 cursor.fetchall()
```

Out[29]:

In [30]:

```
1 cursor.execute('select col1, count(*) , col11 from testraksha.glassdata45 group by c
```

In [31]:

```
1 cursor.fetchall()
```

Out[31]:

```
[ (2, 7, 1),
  (3, 7, 1),
  (5, 7, 1),
  (6, 15, 1),
  (7, 7, 1),
  (8, 7, 1),
  (9, 7, 1),
  (10, 7, 1),
  (11, 7, 1),
  (12, 7, 1),
  (13, 7, 1),
  (14, 7, 1),
  (15, 7, 1),
  (16, 7, 1),
  (17, 7, 1),
  (18, 7, 1),
  (19, 7, 1),
  (20, 7, 1),
```

In [32]:

```
1 cursor.execute('select * from testraksha.glassdata45 where col1=26')
```

In [33]:

```
1 cursor.fetchall()
```

Out[33]:

In [34]:

```
1 cursor.execute('select * from testraksha.capgemini ')
```

In [35]:

```
1 cursor.fetchall()
```

Out[35]:

```
[(1, 'sudh', 'kumar', 41, 'FSDS'),  
(1, 'sudh', 'kumar', 41, 'FSDS'),  
(1, 'sudh', 'kumar', 41, 'FSDS')]
```

In [36]:

```
1 cursor.execute('select count(*) from testraksha.capgemini ')
```

In [37]:

```
1 cursor.fetchall()
```

Out[37]:

```
[(3,)]
```

In [39]:

```
1 cursor.execute('select count(*) from testraksha.capgemini where classname = "FSDS"')
```

In [40]:

```
1 cursor.fetchall()
```

Out[40]:

```
[(3,)]
```

In [46]:

```
1 cursor.execute('select count(*) , classname from testraksha.capgemini group by class')
```

In [47]:

```
1 cursor.fetchall()
```

Out[47]:

```
[(3, 'FSDS')]
```

In [48]:

```
1 # drop table tablename
```

In [52]:

```
1 cursor.execute('select col1,col2,col11 from testraksha.glassdata45 where col1 = 11 o
```

In [53]:

```
1 cursor.fetchall()
```

Out[53]:

```
[(11, 1.51571, 1),  
(71, 1.51574, 2),  
(72, 1.51848, 2),  
(73, 1.51593, 2),  
(74, 1.51631, 2),  
(75, 1.51596, 2),  
(76, 1.5159, 2),  
(77, 1.51645, 2),  
(78, 1.51627, 2),  
(79, 1.51613, 2),  
(80, 1.5159, 2),  
(81, 1.51592, 2),  
(82, 1.51593, 2),  
(83, 1.51646, 2),  
(84, 1.51594, 2),  
(85, 1.51409, 2),  
(86, 1.51625, 2),  
(87, 1.51569, 2).
```

In [54]:

```
1 cursor.execute('select col1,col2,col11 from testraksha.glassdata45 where col1 = 11 a
```

In [55]:

```
1 cursor.fetchall()
```

Out[55]:

```
[]
```

In [63]:

```
1 cursor.execute("select * from testraksha.glassdata45 where col2 LIKE '1.515%' ")
```

In [64]:

```
1 cursor.fetchall()
```

Out[64]:

```
[(6, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1),  
(11, 1.51571, 12.72, 3.46, 1.56, 73.2, 0.67, 8.09, 0.0, 0.24, 1),  
(13, 1.51589, 12.88, 3.43, 1.4, 73.28, 0.69, 8.05, 0.0, 0.24, 1),  
(36, 1.51567, 13.29, 3.45, 1.21, 72.74, 0.56, 8.57, 0.0, 0.0, 1),  
(71, 1.51574, 14.86, 3.67, 1.74, 71.87, 0.16, 7.36, 0.0, 0.12, 2),  
(73, 1.51593, 13.09, 3.59, 1.52, 73.1, 0.67, 7.83, 0.0, 0.0, 2),  
(75, 1.51596, 13.02, 3.56, 1.54, 73.11, 0.72, 7.9, 0.0, 0.0, 2),  
(76, 1.5159, 13.02, 3.58, 1.51, 73.12, 0.69, 7.96, 0.0, 0.0, 2),  
(80, 1.5159, 12.82, 3.52, 1.9, 72.86, 0.69, 7.97, 0.0, 0.0, 2),  
(81, 1.51592, 12.86, 3.52, 2.12, 72.66, 0.69, 7.97, 0.0, 0.0, 2),  
(82, 1.51593, 13.25, 3.45, 1.43, 73.17, 0.61, 7.86, 0.0, 0.0, 2),  
(84, 1.51594, 13.09, 3.52, 1.55, 72.87, 0.68, 8.05, 0.0, 0.09, 2),  
(87, 1.51569, 13.24, 3.49, 1.47, 73.25, 0.38, 8.03, 0.0, 0.0, 2),  
(93, 1.51588, 13.12, 3.41, 1.58, 73.26, 0.07, 8.39, 0.0, 0.19, 2),  
(94, 1.5159, 13.24, 3.34, 1.47, 73.1, 0.39, 8.22, 0.0, 0.0, 2),  
(164, 1.51514, 14.01, 2.68, 3.5, 69.89, 1.68, 5.87, 2.2, 0.0, 5),  
(196, 1.51545, 14.14, 0.0, 2.68, 73.39, 0.08, 9.07, 0.61, 0.05, 7),  
(197, 1.51556, 13.87, 0.0, 2.54, 73.23, 0.14, 9.41, 0.81, 0.01, 7).
```

In [67]:

```
1 cursor.execute('select * from testraksha.glassdata45 order by col2 desc')
```

In [68]:

```
1 cursor.fetchall()
```

Out[68]:

```
[(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(108, 1.53393, 12.3, 0.0, 1.0, 70.16, 0.12, 16.19, 0.0, 0.24, 2),  
(108, 1.53393, 12.3, 0.0, 1.0, 70.16, 0.12, 16.19, 0.0, 0.24, 2),  
(108, 1.53393, 12.3, 0.0, 1.0, 70.16, 0.12, 16.19, 0.0, 0.24, 2),  
(108, 1.53393, 12.3, 0.0, 1.0, 70.16, 0.12, 16.19, 0.0, 0.24, 2),  
(108, 1.53393, 12.3, 0.0, 1.0, 70.16, 0.12, 16.19, 0.0, 0.24, 2),  
(108, 1.53393, 12.3, 0.0, 1.0, 70.16, 0.12, 16.19, 0.0, 0.24, 2),  
(108, 1.53393, 12.3, 0.0, 1.0, 70.16, 0.12, 16.19, 0.0, 0.24, 2),  
(108, 1.53393, 12.3, 0.0, 1.0, 70.16, 0.12, 16.19, 0.0, 0.24, 2),  
(108, 1.53393, 12.3, 0.0, 1.0, 70.16, 0.12, 16.19, 0.0, 0.24, 2),  
(107, 1.53125, 10.73, 0.0, 2.1, 69.81, 0.58, 13.3, 3.15, 0.28, 2),  
(107, 1.53125, 10.73, 0.0, 2.1, 69.81, 0.58, 13.3, 3.15, 0.28, 2),  
(107, 1.53125, 10.73, 0.0, 2.1, 69.81, 0.58, 13.3, 3.15, 0.28, 2).
```

In [70]:

```
1 cursor.execute('select * from testraksha.glassdata45 where col1 > 112 ')
```

In [71]:

```
1 cursor.fetchall()
```

Out[71]:

```
[(113, 1.52777, 12.64, 0.0, 0.67, 72.02, 0.06, 14.4, 0.0, 0.0, 2),  
(114, 1.51892, 13.46, 3.83, 1.26, 72.55, 0.57, 8.21, 0.0, 0.14, 2),  
(115, 1.51847, 13.1, 3.97, 1.19, 72.44, 0.6, 8.43, 0.0, 0.0, 2),  
(116, 1.51846, 13.41, 3.89, 1.33, 72.38, 0.51, 8.28, 0.0, 0.0, 2),  
(117, 1.51829, 13.24, 3.9, 1.41, 72.33, 0.55, 8.31, 0.0, 0.1, 2),  
(118, 1.51708, 13.72, 3.68, 1.81, 72.06, 0.64, 7.88, 0.0, 0.0, 2),  
(119, 1.51673, 13.3, 3.64, 1.53, 72.53, 0.65, 8.03, 0.0, 0.29, 2),  
(120, 1.51652, 13.56, 3.57, 1.47, 72.45, 0.64, 7.96, 0.0, 0.0, 2),  
(121, 1.51844, 13.25, 3.76, 1.32, 72.4, 0.58, 8.42, 0.0, 0.0, 2),  
(122, 1.51663, 12.93, 3.54, 1.62, 72.96, 0.64, 8.03, 0.0, 0.21, 2),  
(123, 1.51687, 13.23, 3.54, 1.48, 72.84, 0.56, 8.1, 0.0, 0.0, 2),  
(124, 1.51707, 13.48, 3.48, 1.71, 72.52, 0.62, 7.99, 0.0, 0.0, 2),  
(125, 1.52177, 13.2, 3.68, 1.15, 72.75, 0.54, 8.52, 0.0, 0.0, 2),  
(126, 1.51872, 12.93, 3.66, 1.56, 72.51, 0.58, 8.55, 0.0, 0.12, 2),  
(127, 1.51667, 12.94, 3.61, 1.26, 72.75, 0.56, 8.6, 0.0, 0.0, 2),  
(128, 1.52081, 13.78, 2.28, 1.43, 71.99, 0.49, 9.85, 0.0, 0.17, 2),  
(129, 1.52068, 13.55, 2.09, 1.67, 72.18, 0.53, 9.57, 0.27, 0.17, 2),  
(130, 1.5202, 13.98, 1.35, 1.63, 71.76, 0.39, 10.56, 0.0, 0.18, 2).
```

In [85]:

```
1 cursor.execute('select * from (select * from testraksha.glassdata45 where col1 > 112
```

In [86]:

```
1 cursor.fetchall()
```

Out[86]:

```
[(147, 1.51769, 13.65, 3.66, 1.11, 72.77, 0.11, 8.6, 0.0, 0.0, 3),  
(148, 1.5161, 13.33, 3.53, 1.34, 72.67, 0.56, 8.33, 0.0, 0.0, 3),  
(149, 1.5167, 13.24, 3.57, 1.38, 72.7, 0.56, 8.44, 0.0, 0.1, 3),  
(150, 1.51643, 12.16, 3.52, 1.35, 72.89, 0.57, 8.53, 0.0, 0.0, 3),  
(151, 1.51665, 13.14, 3.45, 1.76, 72.48, 0.6, 8.38, 0.0, 0.17, 3),  
(152, 1.52127, 14.32, 3.9, 0.83, 71.5, 0.0, 9.49, 0.0, 0.0, 3),  
(153, 1.51779, 13.64, 3.65, 0.65, 73.0, 0.06, 8.93, 0.0, 0.0, 3),  
(154, 1.5161, 13.42, 3.4, 1.22, 72.69, 0.59, 8.32, 0.0, 0.0, 3),  
(155, 1.51694, 12.86, 3.58, 1.31, 72.61, 0.61, 8.79, 0.0, 0.0, 3),  
(156, 1.51646, 13.04, 3.4, 1.26, 73.01, 0.52, 8.58, 0.0, 0.0, 3),  
(157, 1.51655, 13.41, 3.39, 1.28, 72.64, 0.52, 8.65, 0.0, 0.0, 3),  
(158, 1.52121, 14.03, 3.76, 0.58, 71.79, 0.11, 9.65, 0.0, 0.0, 3),  
(159, 1.51776, 13.53, 3.41, 1.52, 72.04, 0.58, 8.79, 0.0, 0.0, 3),  
(160, 1.51796, 13.5, 3.36, 1.63, 71.94, 0.57, 8.81, 0.0, 0.09, 3),  
(161, 1.51832, 13.33, 3.34, 1.54, 72.14, 0.56, 8.99, 0.0, 0.0, 3),  
(162, 1.51934, 13.64, 3.54, 0.75, 72.65, 0.16, 8.89, 0.15, 0.24, 3),  
(163, 1.52211, 14.19, 3.78, 0.91, 71.36, 0.23, 9.14, 0.0, 0.37, 3),  
(147, 1.51769, 13.65, 3.66, 1.11, 72.77, 0.11, 8.6, 0.0, 0.0, 3).
```

In [88]:

```
1 cursor.execute('select * from testraksha.glassdata45 where count(col1) > 2 group by
```

```
-  
MySQLInterfaceError                                     Traceback (most recent call last)  
t)  
~\anaconda3\lib\site-packages\mysql\connector\connection_cext.py in cmd_qu  
ery(self, query, raw, buffered, raw_as_string)  
    534         query = query.encode('utf-8')  
--> 535         self._cmysql.query(query,  
    536                         raw=raw, buffered=buffered,
```

MySQLInterfaceError: Invalid use of group function

During handling of the above exception, another exception occurred:

```
DatabaseError                                         Traceback (most recent call las  
t)  
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_21524\896776599.py in <modu  
le>  
----> 1 cursor.execute('select * from testraksha.glassdata45 where count(c  
ol1) > 2 group by col1')  
  
~\anaconda3\lib\site-packages\mysql\connector\cursor_cext.py in execute(se  
lf, operation, params, multi)  
    267  
    268     try:  
--> 269         result = self._cnx.cmd_query(stmt, raw=self._raw,  
    270                         buffered=self._buffered,  
    271                         raw_as_string=self._raw_a  
s_string)  
  
~\anaconda3\lib\site-packages\mysql\connector\connection_cext.py in cmd_qu  
ery(self, query, raw, buffered, raw_as_string)  
    538             query_attrs=self._query_attrs)  
    539     except MySQLInterfaceError as exc:  
--> 540         raise errors.get_mysql_exception(exc errno, msg=exc.ms  
g,  
    541                                         sqlstate=exc.sqlstat  
e)  
    542     except AttributeError:
```

DatabaseError: 1111 (HY000): Invalid use of group function

In [106]:

```
1 cursor.execute('select * from testraksha.glassdata45 group by col1 having count(col1
```

In [107]:

```
1 cursor.fetchall()
```

Out[107]:

```
[(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1)]
```

In [103]:

```
1 cursor.execute('select * from testraksha.glassdata45 where col1 = 6 ')
```

In [96]:

```
1 cursor.fetchall()
```

Out[96]:

```
[(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1),  
(6, 8.0, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1),  
(6, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1)]
```

In []:

```
1
```

In [1]:

```
1 s = "sudh"
```

In [2]:

```
1 type(s)
```

Out[2]:

str

In [3]:

```
1 s[0]
```

Out[3]:

's'

In [4]:

```
1 s[-1]
```

Out[4]:

'h'

In [5]:

```
1 s[2]
```

Out[5]:

'd'

In [6]:

```
1 s[3]
```

Out[6]:

'h'

In [8]:

```
1 s[-2]
```

Out[8]:

'd'

In [9]:

```
1 s[-3]
```

Out[9]:

'u'

In [10]:

```
1 s[-2]
```

Out[10]:

'd'

In [11]:

```
1 s[-4]
```

Out[11]:

's'

In [12]:

```
1 a = "my name is sudh"
```

In [13]:

```
1 a[0]
```

Out[13]:

'm'

In [15]:

```
1 len(a)
```

Out[15]:

15

In [16]:

```
1 a
```

Out[16]:

'my name is sudh'

In [17]:

```
1 a[0:10]
```

Out[17]:

'my name is'

In [18]:

```
1 a[0:]
```

Out[18]:

```
'my name is sudh'
```

In [19]:

```
1 b = 'ineuron'
```

In [20]:

```
1 b[0:3]
```

Out[20]:

```
'ine'
```

In [21]:

```
1 b[0:300]
```

Out[21]:

```
'ineuron'
```

In [23]:

```
1 a[0:100]
```

Out[23]:

```
'my name is sudh'
```

In [26]:

```
1 b[300]
```

```
-----
-
IndexError                                Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_9308/2380424124.py in <module>
     1 b[300]
```

IndexError: string index out of range

In [27]:

```
1 b
```

Out[27]:

```
'ineuron'
```

In [28]:

```
1 b[-1]
```

Out[28]:

```
'n'
```

In [30]:

```
1 b[-1:-4]
```

Out[30]:

```
' '
```

In [31]:

```
1 b[0:4]
```

Out[31]:

```
'ineu'
```

In [32]:

```
1 a = "kumar"
```

In [33]:

```
1 a[0:3]
```

Out[33]:

```
'kum'
```

In [34]:

```
1 a[0:300]
```

Out[34]:

```
'kumar'
```

In [35]:

```
1 a[0:300:1]
```

Out[35]:

```
'kumar'
```

In [36]:

```
1 a[-1:-100:-1]
```

Out[36]:

```
'ramuk'
```

In [37]:

```
1 a[0:300:2]
```

Out[37]:

'kmr'

In [38]:

```
1 a[0:300:3]
```

Out[38]:

'ka'

In [39]:

```
1 a
```

Out[39]:

'kumar'

In [40]:

```
1 a[0:100:-1]
```

Out[40]:

''

In [41]:

```
1 a
```

Out[41]:

'kumar'

In [42]:

```
1 a[-1:-4:-1]
```

Out[42]:

''

In [43]:

```
1 a[-1:-4:-1]
```

Out[43]:

'ram'

In [44]:

```
1 a[0:-10:-1]
```

Out[44]:

'k'

In [45]:

```
1 a[::-1]
```

Out[45]:

'kumar'

In [46]:

```
1 a[::-1]
```

Out[46]:

'ramuk'

In [47]:

```
1 a[:8]
```

Out[47]:

'kumar'

In [48]:

```
1 a[-2:]
```

Out[48]:

'ar'

In [49]:

```
1 a
```

Out[49]:

'kumar'

In [50]:

```
1 a[-2:-1]
```

Out[50]:

'a'

In [51]:

```
1 a[1:2]
```

Out[51]:

```
'u'
```

In [52]:

```
1 a
```

Out[52]:

```
'kumar'
```

In [53]:

```
1 a[::-1]
```

Out[53]:

```
'ramuk'
```

In [54]:

```
1 a[-1::-1]
```

Out[54]:

```
'ramuk'
```

In [55]:

```
1 a = "I am working with CG"
```

In [56]:

```
1 a[::-1]
```

Out[56]:

```
'GC htiw gnikrow ma I'
```

In [57]:

```
1 a
```

Out[57]:

```
'I am working with CG'
```

In [59]:

```
1 a[-5:5]
```

Out[59]:

```
' '
```

In [60]:

```
1 a[-5:5:-1]
```

Out[60]:

```
'tiw gnikro'
```

In [61]:

```
1 a[0:100:3]
```

Out[61]:

```
'Imoi tC'
```

In [62]:

```
1 "sudh"*3
```

Out[62]:

```
'sudhsudhsudh'
```

In [65]:

```
1 "sudh" +"kumar"
```

Out[65]:

```
'sudhkumar'
```

In [66]:

```
1 a
```

Out[66]:

```
'I am working with CG'
```

In [67]:

```
1 len(a)
```

Out[67]:

```
20
```

In [68]:

```
1 a
```

Out[68]:

```
'I am working with CG'
```

In [69]:

```
1 a.find('a')
```

Out[69]:

2

In [72]:

```
1 a.find('Ia')
```

Out[72]:

-1

In [73]:

```
1 a.find('in')
```

Out[73]:

9

In [77]:

```
1 a.count('i')
```

Out[77]:

2

In [78]:

```
1 a
```

Out[78]:

'I am working with CG'

In [80]:

```
1 a.split()
```

Out[80]:

['I', 'am', 'working', 'with', 'CG']

In [81]:

```
1 type(a.split())
```

Out[81]:

list

In [82]:

```
1 l = a.split()
```

In [83]:

```
1 l
```

Out[83]:

```
['I', 'am', 'working', 'with', 'CG']
```

In [84]:

```
1 l[0]
```

Out[84]:

```
'I'
```

In [85]:

```
1 l[3]
```

Out[85]:

```
'with'
```

In [86]:

```
1 l[2]
```

Out[86]:

```
'working'
```

In [87]:

```
1 l[4]
```

Out[87]:

```
'CG'
```

In [88]:

```
1 len(l)
```

Out[88]:

```
5
```

In [90]:

```
1 a
```

Out[90]:

```
'I am working with CG'
```

In [93]:

```
1 a.split('wo')
```

Out[93]:

```
['I am ', 'rking with CG']
```

In [94]:

```
1 a.upper()
```

Out[94]:

```
'I AM WORKING WITH CG'
```

In [95]:

```
1 s = "sUdh"
```

In [96]:

```
1 s.swapcase()
```

Out[96]:

```
'SuDH'
```

In [97]:

```
1 s.title()
```

Out[97]:

```
'Sudh'
```

In [98]:

```
1 s.capitalize()
```

Out[98]:

```
'Sudh'
```

In [99]:

```
1 b = "sudh"
2 c = "ineuron"
```

In [100]:

```
1 b.join(c)
```

Out[100]:

```
'isudhnsudhesudhusudhrsudhosudhn'
```

In [102]:

```
1 " ".join("sudh")
```

Out[102]:

```
's u d h'
```

In [103]:

```
1 reversed("sudh")
```

Out[103]:

```
<reversed at 0x2667bc17bb0>
```

In [104]:

```
1 for i in reversed("sudh"):
2     print(i)
```

```
h
d
u
s
```

In [105]:

```
1 s = "sudh"
2 s[::-1]
```

Out[105]:

```
'hdus'
```

In [106]:

```
1 s = " sudh "
```

In [107]:

```
1 s.rstrip()
```

Out[107]:

```
' sudh'
```

In [108]:

```
1 s.lstrip()
```

Out[108]:

```
'sudh '
```

In [109]:

```
1 s.strip()
```

Out[109]:

'sudh'

In [110]:

```
1 s = "sudh"
```

In [111]:

```
1 s.replace("u" , "xyz")
```

Out[111]:

'sxyzdh'

In [112]:

```
1 s.replace('t','xys')
```

Out[112]:

'sudh'

In [113]:

```
1 "sudh\tkumar".expandtabs()
```

Out[113]:

'sudh kumar'

In [114]:

```
1 s= 'sudh'
```

In [116]:

```
1 s.center(40, '#')
```

Out[116]:

'#####sudh#####'

In [117]:

```
1 s.isupper()
```

Out[117]:

False

In [118]:

```
1 s
```

Out[118]:

'sudh'

In [119]:

```
1 s = "sudh"
```

In [121]:

```
1 s.islower()
```

Out[121]:

True

In [122]:

```
1 s.isspace()
```

Out[122]:

False

In [125]:

```
1 s = " "
```

In [126]:

```
1 s.isspace()
```

Out[126]:

True

In [127]:

```
1 s= " sudh"
```

In [128]:

```
1 s.isspace()
```

Out[128]:

False

In [129]:

```
1 s= "sudh123"
```

In [130]:

```
1 s.isdigit()
```

Out[130]:

False

In [131]:

```
1 s= "123"
```

In [132]:

```
1 s.isdigit()
```

Out[132]:

True

In [133]:

```
1 s = "sudh"
```

In [134]:

```
1 s.endswith('h')
```

Out[134]:

True

In [135]:

```
1 s.startswith('t')
```

Out[135]:

False

In [136]:

```
1 s.startswith('s')
```

Out[136]:

True

In [137]:

```
1 s.istitle()
```

Out[137]:

False

In [138]:

```
1 s.isalpha()
```

Out[138]:

True

In [139]:

```
1 s
```

Out[139]:

'sudh'

In [140]:

```
1 s.encode()
```

Out[140]:

b'sudh'

In [141]:

```
1 l = ['sudh', "kumar", 34, 45.50, True, 4+6j]
```

In [142]:

```
1 type(l)
```

Out[142]:

list

In [143]:

```
1 l[-1]
```

Out[143]:

(4+6j)

In [144]:

```
1 l[0]
```

Out[144]:

'sudh'

In [145]:

```
1 l[-5]
```

Out[145]:

'kumar'

In [146]:

```
1 l
```

Out[146]:

```
['sudh', 'kumar', 34, 45.5, True, (4+6j)]
```

In [147]:

```
1 l[0:4]
```

Out[147]:

```
['sudh', 'kumar', 34, 45.5]
```

In [149]:

```
1 l[::-1]
```

Out[149]:

```
[(4+6j), True, 45.5, 34, 'kumar', 'sudh']
```

In [150]:

```
1 l[-1:6]
```

Out[150]:

```
[(4+6j)]
```

In [153]:

```
1 l[0][1]
```

Out[153]:

```
'u'
```

In [154]:

```
1 l[3]
```

Out[154]:

```
45.5
```

In [155]:

```
1 l[4].real
```

Out[155]:

```
1
```

In [156]:

```
1 l[4]
```

Out[156]:

True

In [157]:

```
1 l
```

Out[157]:

```
['sudh', 'kumar', 34, 45.5, True, (4+6j)]
```

In [158]:

```
1 l[5]
```

Out[158]:

```
(4+6j)
```

In [160]:

```
1 l[5].imag
```

Out[160]:

6.0

In [161]:

```
1 l1 = ['sudh', 'kumar', 3452]
2 l2 = ["xyz", 'pqr', 324.34]
```

In [162]:

```
1 l1+l2
```

Out[162]:

```
['sudh', 'kumar', 3452, 'xyz', 'pqr', 324.34]
```

In [163]:

```
1 l1 +'sudh'
```

```
-----
-
TypeError                                Traceback (most recent call last
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_9308/1104314627.py in <modu
le>
----> 1 l1 +'sudh'

TypeError: can only concatenate list (not "str") to list
```

In [164]:

```
1 l1 +["sudh"]
```

Out[164]:

```
['sudh', 'kumar', 3452, 'sudh']
```

In [167]:

```
1 l1*2
```

Out[167]:

```
['sudh', 'kumar', 3452, 'sudh', 'kumar', 3452]
```

In [166]:

```
1 l1
```

Out[166]:

```
['sudh', 'kumar', 3452]
```

In [168]:

```
1 l1
```

Out[168]:

```
['sudh', 'kumar', 3452]
```

In [169]:

```
1 l1[0].replace('sudh','kumar')
```

Out[169]:

```
'kumar'
```

In [170]:

```
1 l1
```

Out[170]:

```
['sudh', 'kumar', 3452]
```

In [171]:

```
1 l1
```

Out[171]:

```
['sudh', 'kumar', 3452]
```

In [174]:

```
1 l1[0]='433q42'
```

In [175]:

```
1 l1
```

Out[175]:

```
['433q42', 'kumar', 3452]
```

In [178]:

```
1 l1[1].replace('k','s')
```

Out[178]:

```
'sumar'
```

In [179]:

```
1 s = "sudh"
```

In [181]:

```
1 #string objects are immutable
2 s[0] = "k"
```

-
TypeError

Traceback (most recent call last)

```
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_9308/4082584902.py in <module>
      1 #string objects are immutable
----> 2 s[0] = "k"
```

TypeError: 'str' object does not support item assignment

In [182]:

```
1 l1
```

Out[182]:

```
['433q42', 'kumar', 3452]
```

In [183]:

```
1 len(l1)
```

Out[183]:

In [184]:

```
1 'kumar' in l1
```

Out[184]:

True

In [185]:

```
1 l2
```

Out[185]:

```
['xyz', 'pqr', 324.34]
```

In [186]:

```
1 l2.append("sudh")
```

In [187]:

```
1 l2
```

Out[187]:

```
['xyz', 'pqr', 324.34, 'sudh']
```

In [188]:

```
1 l2.pop()
```

Out[188]:

'sudh'

In [189]:

```
1 l2
```

Out[189]:

```
['xyz', 'pqr', 324.34]
```

In [190]:

```
1 l2.append(343)
```

In [191]:

```
1 l2
```

Out[191]:

```
['xyz', 'pqr', 324.34, 343]
```

In [192]:

```
1 12.insert(1,"rakshanda")
```

In [194]:

```
1 12
```

Out[194]:

```
['xyz', 'rakshanda', 'pqr', 324.34, 343]
```

In [195]:

```
1 12.insert(3,[1,2,3,4])
```

In [196]:

```
1 12
```

Out[196]:

```
['xyz', 'rakshanda', 'pqr', [1, 2, 3, 4], 324.34, 343]
```

In [197]:

```
1 12[::-1]
```

Out[197]:

```
[343, 324.34, [1, 2, 3, 4], 'pqr', 'rakshanda', 'xyz']
```

In [198]:

```
1 12.reverse()
```

In [199]:

```
1 12
```

Out[199]:

```
[343, 324.34, [1, 2, 3, 4], 'pqr', 'rakshanda', 'xyz']
```

In [200]:

```
1 11
```

Out[200]:

```
['433q42', 'kumar', 3452]
```

In [201]:

```
1 12
```

Out[201]:

```
[343, 324.34, [1, 2, 3, 4], 'pqr', 'rakshanda', 'xyz']
```

In [202]:

```
1 12[2][3]
```

Out[202]:

```
4
```

In [203]:

```
1 12
```

Out[203]:

```
[343, 324.34, [1, 2, 3, 4], 'pqr', 'rakshanda', 'xyz']
```

In [205]:

```
1 12.count('rakshanda')
```

Out[205]:

```
1
```

In [206]:

```
1 12.count('sf')
```

Out[206]:

```
0
```

In [207]:

```
1 12.append([3,4,4,2])
```

In [208]:

```
1 12
```

Out[208]:

```
[343, 324.34, [1, 2, 3, 4], 'pqr', 'rakshanda', 'xyz', [3, 4, 4, 2]]
```

In [209]:

```
1 11
```

Out[209]:

```
['433q42', 'kumar', 3452]
```

In [210]:

```
1 l1.extend([2,4,45.3,True])
```

In [211]:

```
1 l1
```

Out[211]:

```
['433q42', 'kumar', 3452, 2, 4, 45.3, True]
```

In [218]:

```
1 key = ("name", "mob_no", "email_id")
2 value = "sudh"
```

In [219]:

```
1 d1 = {'key1': 'raksha'}
```

In [220]:

```
1 d1
```

Out[220]:

```
{'key1': 'raksha'}
```

In [221]:

```
1 d = d1.fromkeys(key,value)
```

In [222]:

```
1 d
```

Out[222]:

```
{'name': 'sudh', 'mob_no': 'sudh', 'email_id': 'sudh'}
```

In []:

```
1
```

In [47]:

```
1 class dict_parsing():
2
3     def __init__(self,a):
4         self.a = a
5
6     def get_keys(self):
7         if self.verify_dict() :
8             return list(self.a.keys())
9
10    def get_values(self):
11        if self.verify_dict() :
12            return list(self.a.values())
13
14    def verify_dict(self):
15        if type(self.a) !=dict:
16            raise Exception('input is not dictionary' , self.a)
17
18    def insertion(self,**kwargs):
19        for k ,v in kwargs.items():
20            self.a[k] = v
21
22    return self.a
```

In [48]:

```
1 d = dict_parsing({'a': 5, 'b': 6, 'c': 7, 'h': '67'})
```

In [50]:

```
1 d.insertion(name = 'sudh')
```

Out[50]:

```
{'a': 5, 'b': 6, 'c': 7, 'h': '67', 'name': 'sudh'}
```

In [52]:

```
1 d.insertion(t = 78)
```

Out[52]:

```
{'a': 5, 'b': 6, 'c': 7, 'h': '67', 'name': 'sudh', 't': 78}
```

In [54]:

```
1 d.insertion(h = 56, mail_id = 'raksha@gmail.com' , mob_No = 4532)
```

Out[54]:

```
{'a': 5,
'b': 6,
'c': 7,
'h': 56,
'name': 'sudh',
't': 78,
'mail_id': 'raksha@gmail.com',
'mob_No': 4532}
```

In [32]:

```
1 d.verify_dict()
```

In [29]:

```
1 d.get_keys()
```

In [33]:

```
1 d.get_values()
```

In [16]:

```
1 class dict_parsing():
2
3     def __init__(self,a):
4         if type(a)!= dict:
5             raise Exception('input is not dictionary' , self.a)
6         self.a = a
7
8
9
10    # def verify_dict(self):
11    #     if type(self.a) !=dict:
12    #         raise Exception('input is not dictionary' , self.a)
13
14
15    def get_keys(self):
16        for i in self.a.keys():
17            print(i)
18
19
20    def get_values(self):
21        if self.verify_dict() :
22            return list(self.a.values())
23
24
25    def insertion(self,**kwargs):
26
27        for k ,v in kwargs.items():
28            self.a[k] = v
29        return self.a
```

In [17]:

```
1 d = dict_parsing({'a': 5, 'b': 6, 'c': 7, 'h': '67'})
```

In [18]:

```
1 d.insertion()
```

Out[18]:

```
{'a': 5, 'b': 6, 'c': 7, 'h': '67'}
```

In [21]:

```
1 d.get_keys()
```

```
a  
b  
c  
h
```

In [2]:

```
1 class dict_fun():
2     def __init__(self,a):
3         self.a = a
4
5     def get_keys(self):
6         if self.verify_input():
7             return list(self.a.keys())
8
9     def get_values(self):
10        if self.verify_input():
11            return self.a.values()
12
13    def verify_input(self):
14        if type(self.a) !=dict:
15            raise Exception('input is not dictionary' , b)
```

In [3]:

```
1 d = dict_fun({'a': 5, 'b': 6, 'c': 7, 'h': '67'})
```

In [4]:

```
1 d.get_keys()
```

In [21]:

```
1 import mydict
```

In [22]:

```
1 mydict.Dict_function
```

Out[22]:

```
mydict.Dict_function
```

In [23]:

```
1 a = mydict.Dict_function()
```

```
-----
-
TypeError                                 Traceback (most recent call last)
t)
C:\Users\RAKSHA~1\AppData\Local\Temp\ipykernel_23520\1882946716.py in <module>
----> 1 a = mydict.Dict_function()

TypeError: __init__() missing 1 required positional argument: 'a'
```

In []:

1

① Central Limit Theorem

② Influential Statistics

a) Z test {Z table} [5-6 problems]

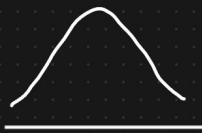
b) t test {t table}

c) Z test proportion population.

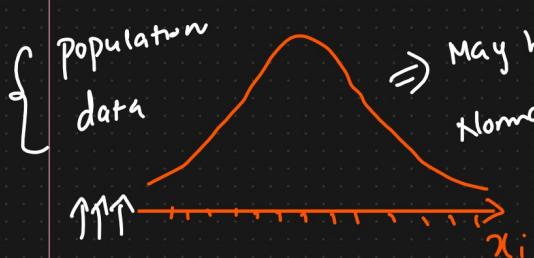
d) Chi Square (Categorical Test)

e) ANNOVA (F Test)

Influential



① Central Limit Theorem



\Rightarrow May be Gaussian
Normal Distr → $[x_1, x_2, x_3, x_4, \dots, x_{30}] \rightarrow \bar{x}_1$
Sample 1

Sample mean distribution

Sample 2 $[x_1, x_2, x_3, x_4, \dots, x_{30}] \rightarrow \bar{x}_2$

$\rightarrow \bar{x}_3$

$\rightarrow \bar{x}_4$

\vdots

$\rightarrow \bar{x}_m$

\Rightarrow It may not

Sample m =

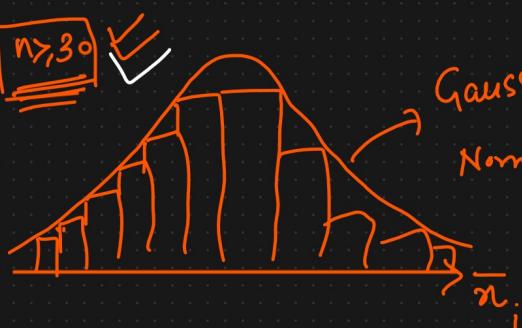
x_i

10,

$n > 30$

Sample mean

distribution

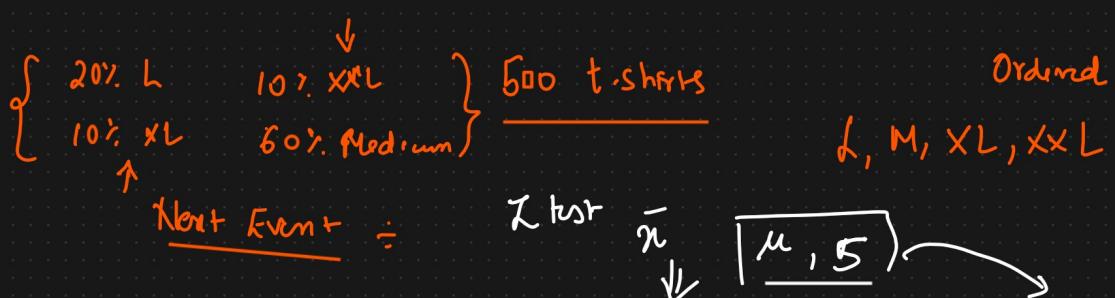


Gaussian Distribution
Normal Distribution

(2) Influential Statistics {Data Analyst, Data Scientist}

① 100K \Rightarrow T-shirt \Rightarrow No \Rightarrow Sample data \Rightarrow X_L, L, Small

② iNeuron \rightarrow Meetup \rightarrow Hitesh \Rightarrow 300-400 people \rightarrow T-shirts



③ ATM ④ Measure the size of entire sharks CI []

⑤ Amazon delivery { Percentile, Quantiles } \Rightarrow

(*) Hypothesis Testing

① A factory has a machine that fills 80ml of baby medicine in a batch. An employee believes the average amount of baby medicine is not 80ml. Using 40 Samples, he measures the average amount dispersed by the machine to be 78ml with a standard deviation of 2.5

(a) State Null and Alternate Hypothesis

(b) At a 95% CI, is there enough evidence to support machine is not working properly.

Ans) Step 1

$$n=40 \quad \bar{x}=78 \quad s=2.5$$

$H_0: \mu = 80$ {Null Hypothesis}

$$H_1: \mu \neq 80 \quad \{ \text{Alternate Hypothesis} \}$$

Why Z test?

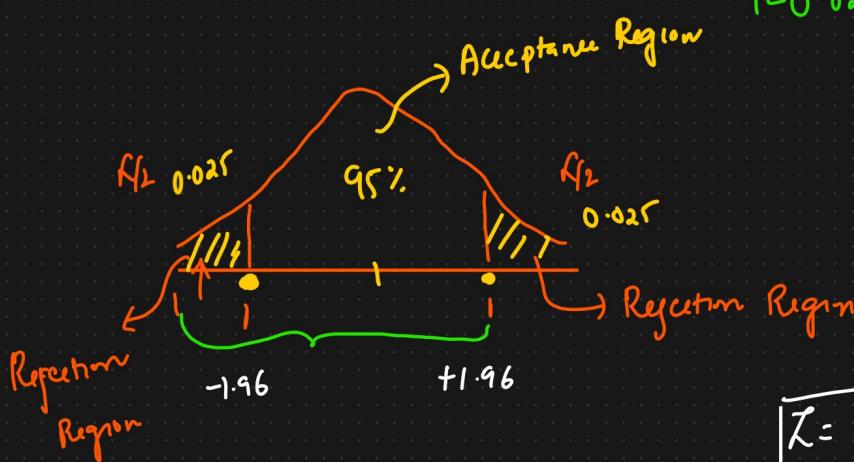
Step 2:

$$\alpha = 0.05$$

$$C.I = 95\%$$

$$\left. \begin{array}{l} \text{① } n > 30 \\ \text{② } \frac{n \leq 30}{\text{population std or sample std}} \end{array} \right\}$$

Step 3: Decision Boundary



$$1 - 0.025 = 0.9750$$

$$\left. \begin{array}{l} \text{① Sample std} \\ \text{② } n < 30 \end{array} \right\} =$$

$$n=1 \}$$

$$Z = \frac{\bar{x}_i - \mu}{\sigma / \sqrt{n}}$$

$$Z = \frac{\bar{x} - \mu}{$$

$$\left[S / \sqrt{n} \right] \Rightarrow \text{Standard Error}$$

Sample
Standard

$$\text{deviation} = \frac{78 - 80}{2.5 / \sqrt{40}} = \frac{-2 \times \sqrt{40}}{2.5} = \frac{-2}{2.5} \times 6.32 = \boxed{-5.05}$$

⑤ State the Results

Decision Rule: If $Z = -5.05$ is less than -1.96 or greater than 1.96 , then reject the null hypothesis with $95\% C.I$.

Reject H_0 Null hypothesis $\{ \text{There is some fault in the machine} \}$

Q) In the population the average IQ is 100 with a standard deviation of 15. A team of scientists wants to test a new medication to see if it has a +ve or -ve effect, or no effect at all.

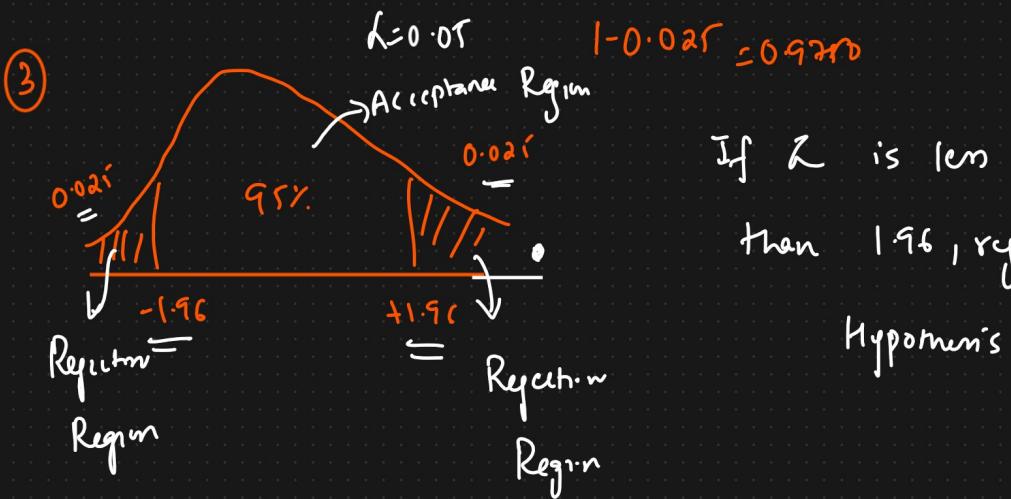
A sample of 30 participants who have taken the medication has a mean of 140. Did the medication affect Intelligence? $\left[\begin{array}{c} 95\% \\ \hline \downarrow \\ C.I. \end{array} \right]$

Ans) $\sigma = 15 \quad n = 30 \quad \bar{x} = 140$

① $H_0 : \mu = 100$

$H_1 : \mu \neq 100$

② $\alpha = 0.05 \quad C.I = 95\%$



④ $Z = \frac{\bar{x} - \mu}{\sigma / \sqrt{n}} = \frac{140 - 100}{15 / \sqrt{30}} = 14.60 \quad \text{---}$

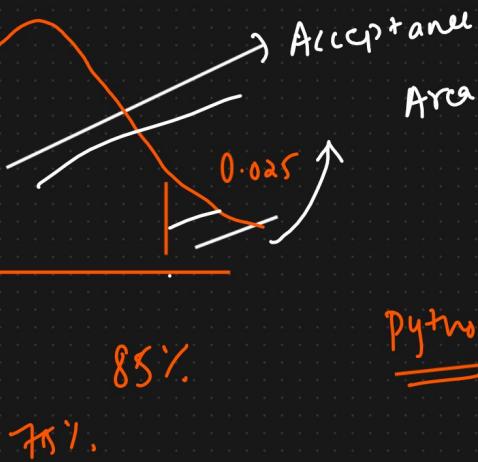
$14.60 > 1.96$ Reject the Null Hypothesis

(*) A Complain was registered, the boys in the Municipal Primary School are underfed. Average weight of boys of age 10 is 32 kgs with $S.D = 9 \text{ kgs}$. A sample of 25 boys was selected from the municipal school and the average weight was found to be 29.5 kgs? With $C.I = 95\%$. Check whether it's True or False?

$$\text{Ans) } \mu = 32 \text{ kgs} \quad \sigma = 9 \text{ kg} \quad n = 25 \quad \bar{x} = 29.5 \quad \alpha = 0.05 \\ \text{1) } H_0 : \mu = 32 \quad \text{2) } \alpha = 0.05 \quad \cdot \quad 1 - 0.95 = 0.05$$

$$H_1 : \mu < 32$$

$$\text{③} \quad \begin{array}{c} \text{Rejection} = 1.96 \\ \downarrow \\ 0.025 \end{array}$$



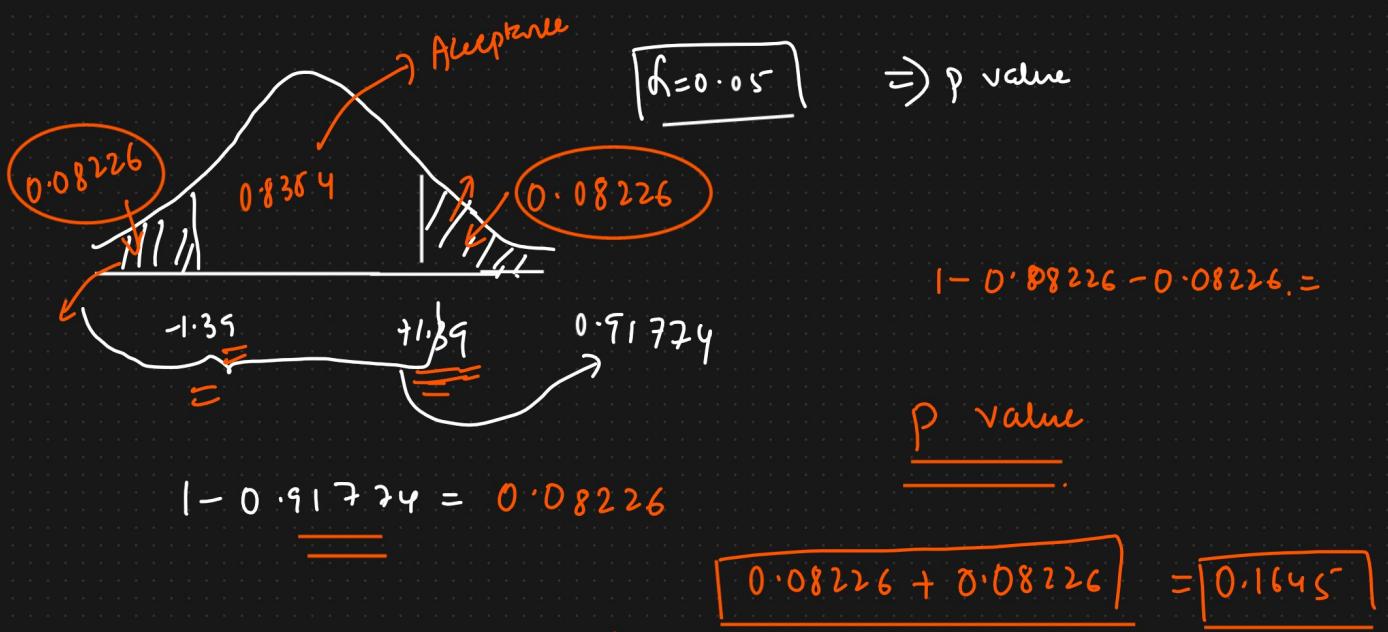
$$\text{④} \quad Z = \frac{\bar{x} - \mu}{\sigma / \sqrt{n}} \\ = \frac{29.5 - 32}{9 / \sqrt{25}} = -1.39$$

Conclusion : $-1.39 > -1.96$ therefore we accept the Null Hypothesis

So, the boys are not underfed.

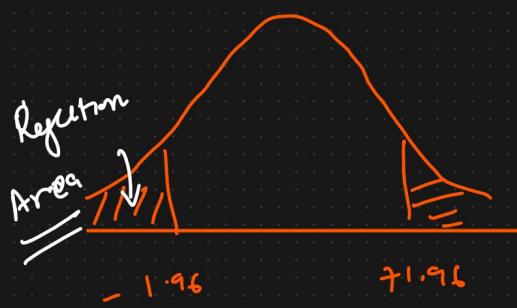
Significance value \Rightarrow

P value



Significance value
 $0.1645 > 0.05$

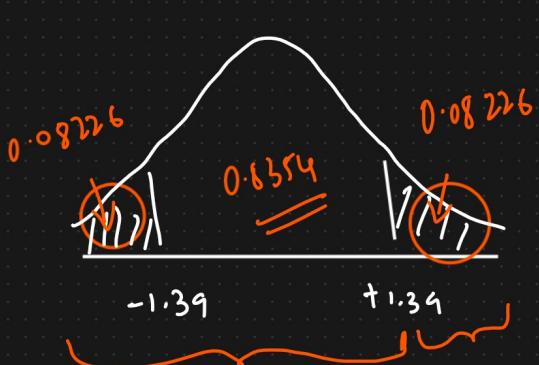
\Downarrow
 $P \geq 0.05 \Rightarrow \text{Accept the Null Hypothesis}$



Z test

\Downarrow

p value



\Rightarrow New p value = $0.08226 + 0.08226$
 $= 0.16$

$$1 - 0.08226 - 0.08226$$

Domain

\Downarrow

$0.1645 > \text{Significance}$

\Downarrow
 value

$$1 - 0.91774 = 0.08226$$

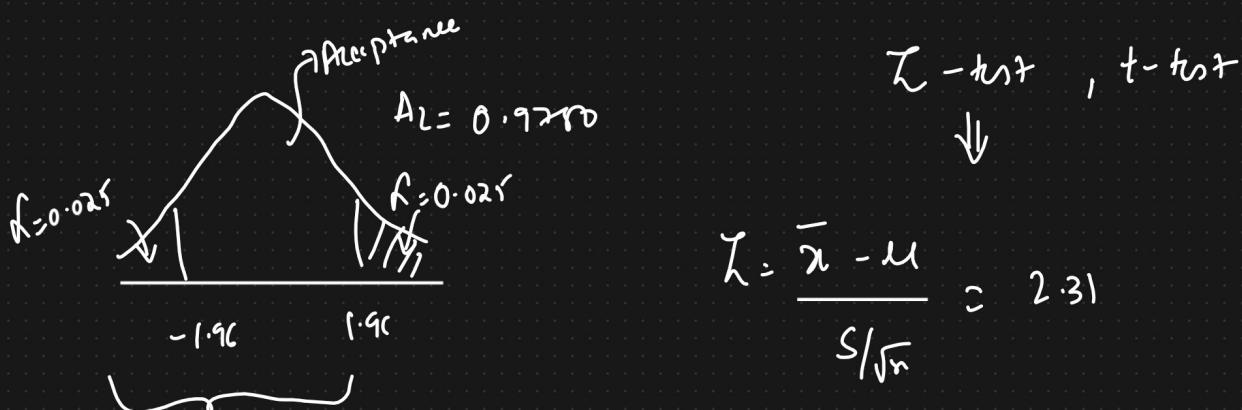
\Rightarrow Accept the Null Hypothesis.

④ The average weight of all residents in town XYZ is 168 lbs. A nutritionist believes the true mean to be different. She measured the weight of 36 individuals and found the mean to be 169.5 lbs with a standard deviation of 3.9.

(a) At 95% CI is there enough evidence to discard the Null Hypothesis??

$$\text{Ans}) \quad H_0 : \mu = 168 \quad n = 36 \quad \bar{x} = 169.5 \quad s = 3.9$$

$$H_1 : \mu \neq 168 \quad \underline{\quad} \quad c = 0.95 \quad \alpha = 1 - c \cdot I = 0.05$$



$2.31 > 1.96$ Reject the Null Hypothesis

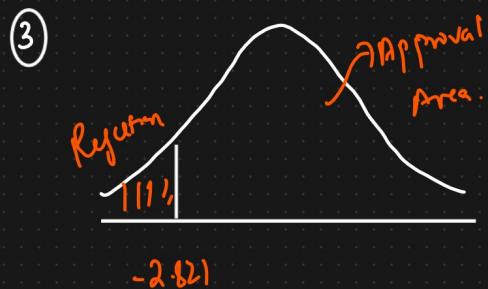
⑤ A company manufactures bike batteries with an average life span of 2 or more years. An engineer believes this value to be less. Using 10 samples, he measures the average life span to be 1.8 years with a standard deviation of 0.15.

a) State the Null and Alternative Hypothesis

b) At a 99% CI, is there enough evidence to discard the H_0 ?

Ans) $H_0 : \mu \geq 2$ $n=10$ $\bar{x}=1.8$ $S=0.15$ $\{$ of sample
 $H_1 : \mu < 2$ ≤ 3.0 $t-tst??$ Std is
 $\{$ given }

② $\alpha = 0.01$ $\alpha = 1 - C.I = 1 - 0.99 = 0.01$



Degrees of freedom: $n-1$

$= 9$

④ Calculate t-test Statistic:

$$t = \frac{\bar{x} - \mu}{S/\sqrt{n}} = \frac{1.8 - 2}{0.15/\sqrt{10}} = \frac{-0.2}{0.15/\sqrt{10}} = \frac{-0.2}{0.15/\sqrt{10}} = -4.216$$

⑤ Conclusion

$-4.216 < -2.821$ Reject the Null Hypothesis. }
 \Downarrow

Z test with proportions

⑥ A tech company believes that the percentage of residents in town XYZ that owns a cell phone is 70%. A marketing manager believes that this value to be different. He conducts a survey of 200 individuals and found that 130 responded yes to

Owning a cell phone

(a) State the Null and Alternative Hypothesis?

(b) At a 95% CI, is there enough evidence to reject the Null Hypothesis?

Ans) $H_0: p_0 = 0.70.$

$H_1: p_0 \neq 0.70$

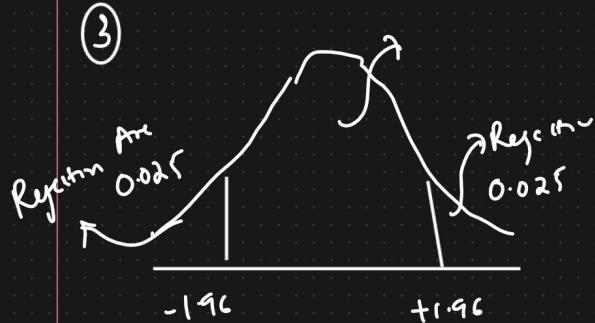
$$n = 200 \quad X = 130 \\ \hat{P} = \frac{X}{n} = \frac{130}{200} = \frac{13}{20} = 0.65$$

$$q_0 = 1 - p_0$$

② $\alpha = 0.05 \quad C.I = 95\%$

$$Z_{test} = \frac{\hat{P} - P_0}{\sqrt{\frac{p_0 q_0}{n}}}$$

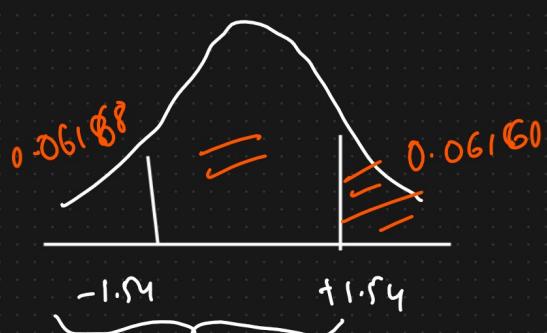
$$= \frac{0.65 - 0.70}{\sqrt{\frac{0.7 \times 0.3}{200}}} \approx -1.54$$



At 95% C.I there is

$-1.54 > -1.96$, so we accept

the Null Hypothesis



p-value
 \downarrow
 $2 \times 0.06168 > 0.05$

Accept Null Hypothesis

$$1 - 0.93822 = 0.06168$$

④ A car company believes that the percentage of residents in City ABC that owns a vehicle is 60% or less. A sales manager disagrees with this. He conducts a hypothesis testing surveying 250 residents and found that 170 responded yes to owning a vehicle.

- (a) State the Null & Alternate Hypothesis
- (b) At 10% significance level, is there enough evidence to support the idea that vehicle ownership in City ABC is 60% or less?

$$p\text{ value} = .014$$

Statistics

Statistics is the science of collecting, organizing and analyzing data.

Data: "facts or pieces of information"

Eg: Height of students in a classroom
 $\rightarrow \{175\text{cm}, 150\text{cm}, 140\text{cm}, 130\text{cm}, 155\text{cm}\}$

Eg: Intelligence Quotient (IQ) of 5 randomly selected individuals ($109, 89, 129, 101, 105, 106$) \rightarrow Data.

Two Types

Statistics



It consists of organizing and summarizing of data.

It consists of using that you've measured to form

Conclusions

Eg: Pdf, Histogram, Box plot, Bar chart, Pie charts

Eg: Hypothesis Testing, p value Z test, t-test, Anova, Chi-square

Eg: Let's say there are 20 maths classes at your university and you've collected the ages of students in one class.

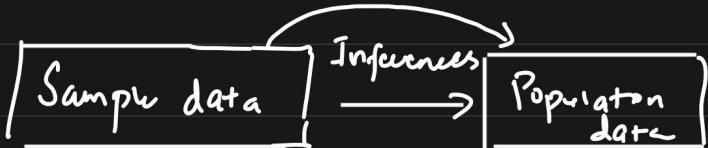
Ages $\{21, 20, 18, 34, 17, 22, 24, 25, 26, 23, 22\}$

$\min = \text{mode}$

Descriptive stats: What is the average age of student in

your maths class?

Inferrential question : Are the ages of students in this maths classroom similar to what you would expect in a normal maths class at this university?

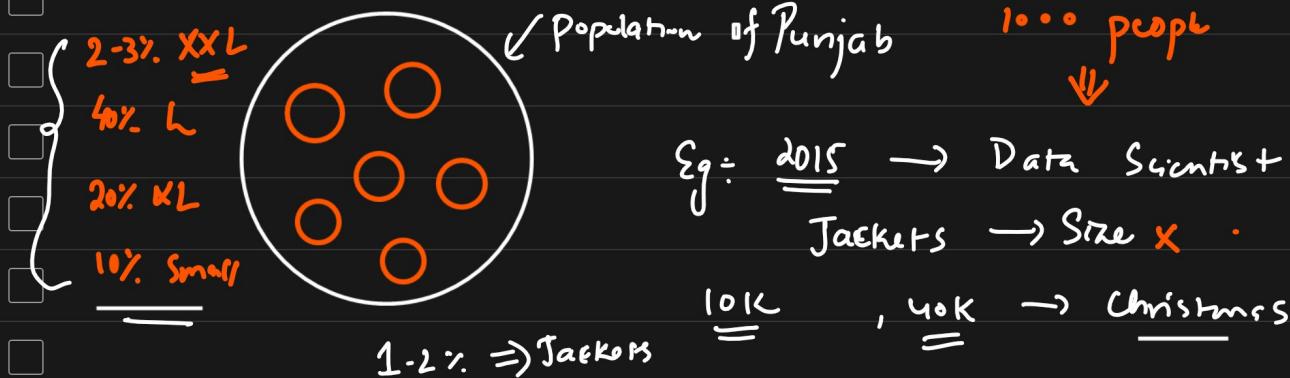


Population And Sample Data → Inferrential Statistics Results

Elections : Punjab

{ AAP, Congress }

Exit Polls ←

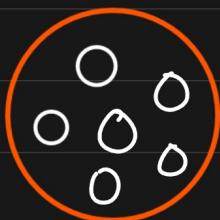


Population (N) ✓

Sample (n) ✓

Sampling Techniques

① Simple Random Sampling : Every member of the population (N) has an equal chance of being selected for your sample (n)



② Stratified Sampling

Strata → Layers
↓
Clusters
↑
Non overlapping groups

Gender → Male
Female

Blood Groups

Age groups
0-18 }
18-35 }
35-60 }

Tax Slabs
Courses

Education Qualification

Thamno

Australias

③ Systematic Sampling

Snap

Customs

(N) → Select every n^{th} individual

$\nearrow 6^{\text{th}}$
=

\downarrow
Stratified

Eg: Survey → Mail (SBI credit card)

④ Convenience Sampling : Only those people who are interested will only be participating.

Healthcare Disease

Eg: Data Science → AI }
YouTube Survey → }

{ Blind people }

→ RBI → Household Survey → Female ← $\frac{\downarrow \downarrow \downarrow \downarrow \downarrow}{\text{Economics}}$ → DATA Science }

Exit Poll : Stratified + Random Sampling

Variable

A variable is a property that can take on any value

Eg: Height = 182
150
145
160

{ 182, 170, 145, 160 }
↓
No

Two kinds of Variable

① Quantitative Variable → Measured Numerically { Add, Subtract, \times , \div }

② Qualitative Variable.

↳ Eg: Gender [Male { Based on some { characteristics } we can derive categorical variables }
Female]

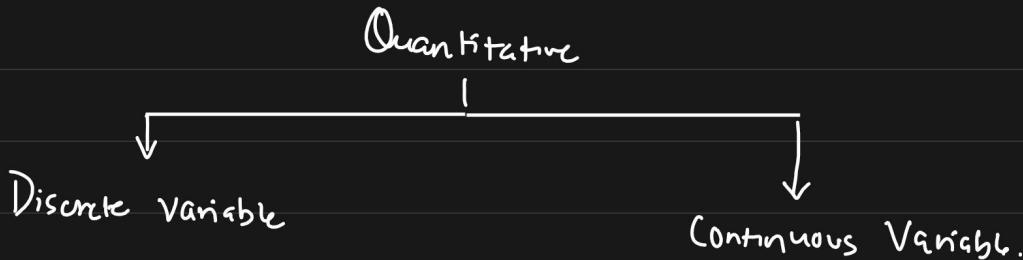
{ Quantitative → Qualitative Variable. }

Eg: IQ

0-10 10-50 50-100

↓ ↓ ↓

Low IQ Medium IQ Good IQ



Eg: whole number

Eg: No. of Bank accounts

{ 2, 3, 4, 5, 6, 7 } 2.5, X
 2.75 X

Eg: Total No. of children in a family

Eg: Height = 172.5, 162.5 cm,

163.5 cm.

Rainfall: 1.35, 1.25, 1.75, 2.25 cm

Weight

Temp

Eg: 2, 3, 4, 5

Stock price.

25, 2.75

Eg: Total no. of Employees in a Company {e.g.: 10k,

Ass:

- ① What kind of variable Marital Status is? Categorical
- ② What kind of variable Nile River length is? Continuous Quantitative
- ③ What kind of " Movie duration is? " "
- ④ What kind of Variable IQ is? " "

Frequency Distribution

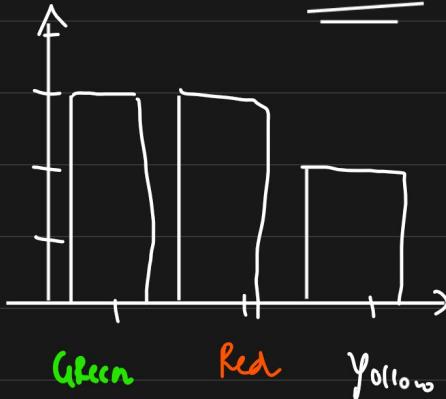
Sample Data: Green, Red, Yellow, Green, Red, Yellow, Green, Red

↓

Colors	Frequency
Green	3
Red	3
Yellow	2

① Bar Graph frequency

Bar Chart



① Variable Measurement Scales

4 types of Measured Variable.

① Nominal data { Categorical data }

Eg: Colors, Gender, Types of flowers

Ranking is not that important

② Ordinal data

Student (Marks)

→ 100

96

57

85

44

Rank

1

2

4

3

5

Percentiles

Ordinal Data.

PHD →
↓
{ NLP } ↓

Degree

PHD

B.G

Master

BCA

12

Salary

✓

✓

✓

✓

✓

Assignment

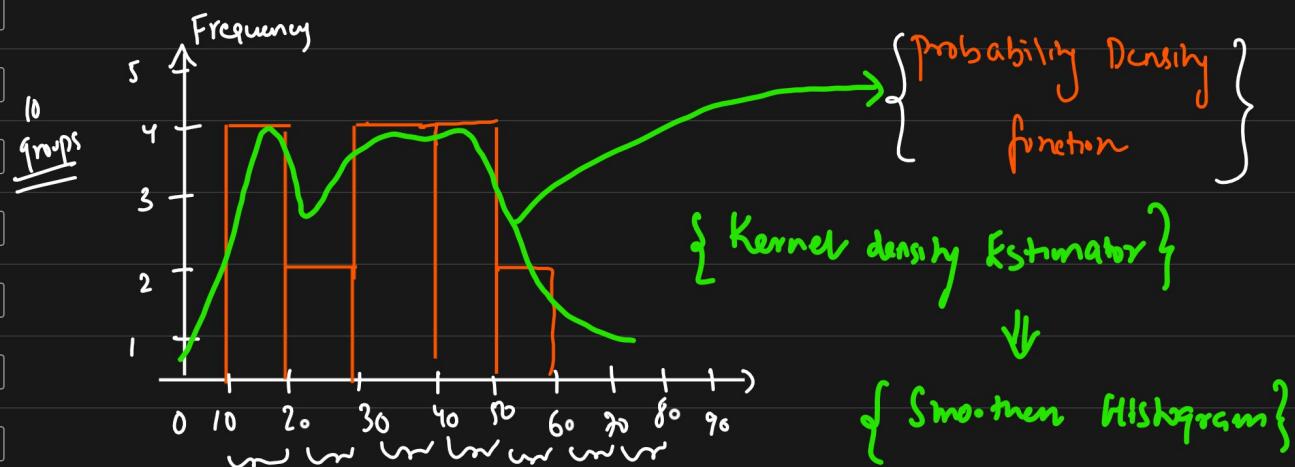
④ Ratio data ✓

③ Interval data ✓

⑥ Histograms ÷ Continuous

$$Age = \{ 10, 12, 14, 18, 24, 26, 30, 35, 36, 37, 40, 41, 42, 43, 50, 51 \}$$

Histogram $\rightarrow \text{Bins} = 10$ \equiv Mean, Median, Mode.



0 - 10 \rightarrow 0-5, 5-10, 10-15, 15-20, 20-25, 25-30, 30-35

Assignment

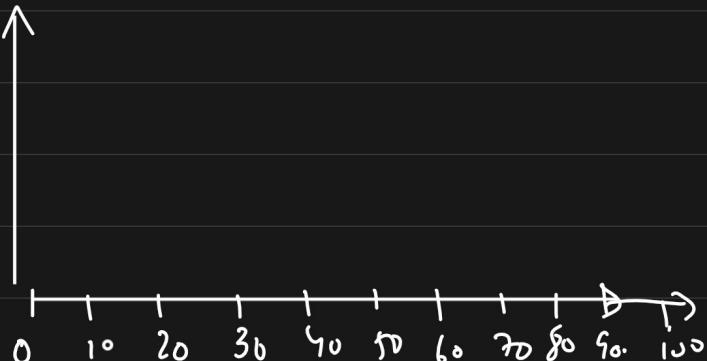
Eg: 10, 13, 18, 22, 27, 32, 38, 40, 45, 51, 56, 57, 88, 90, 92, 94, 99

bins \downarrow
10

0-10 10-20 20-30 30-40

40-50 50-60 60-70

70-80 80-90 90-100

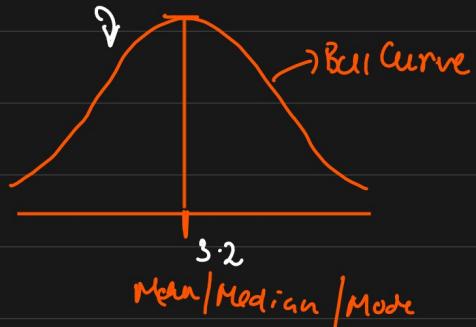


Intermediate Stats

- ① Measure of Central Tendency
- ② Measure of Dispersion
- ③ Gaussian Distribution
- ④ Z - Score
- ⑤ Standard Normal Distribution
- ⑥ Central Limit Theorem
-

- ① Measure of Central Tendency → Central position of the dataset
-

- ① Mean ✓
- ② Median ✓ { EDA & Feature Eng. }
- ③ Mode ✓
-



Population (N)

$$X = \{1, 1, 2, 2, 3, 3, 4, 5, 5, 6\}$$

$$\mu = \frac{\sum_{i=1}^N x_i}{N}$$

Population

mean

$$= \frac{1+1+2+2+3+3+4+5+5+6}{10}$$

$$= \frac{32}{10} = 3.2$$

Sample (n)

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

Sample

Mean

Median

1, 2, 2, 3, 4, 5

↓
1, 2, 2, 3, 4, 5, 100

$$\bar{x} = \frac{1+2+2+3+4+5}{6} = \frac{17}{6} = 2.83$$

$$\bar{x} = \frac{1+2+2+3+4+5+100}{7} = \frac{117}{7} = 16.71$$

Median ✓

1, 2, 2 3 4, 5, 100

$$\bar{x} = 16.71 //$$

$$\text{Median} = 3 \\ =$$

1, 2, 2, 3, 4, 5 → odd or even
↓ $\frac{2+3}{2} = 2.5$

2.5 $\approx 2.83 //$

Mode ÷ Highest frequency: Median

1, 2, 2, 3, 3, 3, 4, 5, 6, 6, 7

↓
3 ↓

EDA

1, 2, 2, 3, 3, 4, 4, 5, 5
↓
{mode}

[2, 3, 4]

Feature Engineering

↪ NAN values ⇒ Continuous Values + outlier
= = Mean ↓
= Median

⇒ Categorical Variable.

↓
Mode

Agnl

Lidley, Sunflower, Rock, - - - , Min, Max

Measure of Dispersion → {Dispersion}

① Variance

② Standard deviation

} \Downarrow

Spread ⇒ How the data is spread



① Variance

Population Variance

{
Basis (Correction)
Degree of freedom}

Sample Variance

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2$$

Population mean

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

Sample mean
 $n-1$

Eg:

$$X = \{1, 2, 2, 3, 4, 5\}$$

<u>x</u>	<u>\bar{x}</u>	<u>$x - \bar{x}$</u>	<u>$(x - \bar{x})^2$</u>
1	2.83	-1.83	3.34
2	2.83	-0.83	0.6889
2	2.83	-0.83	0.6889
3	2.83	0.17	0.03
4	2.83	1.17	1.37
5	2.83	2.17	4.71

$$\left[\frac{10.84}{5} \right] = 2.168$$

$n=6$

$n-1$

$$\mu = 2.83$$

{let consider

$$10.84$$

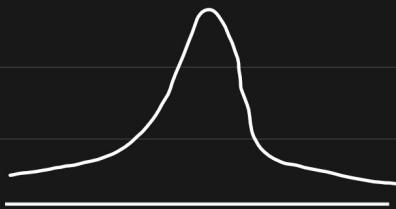
$$\frac{\sigma^2}{\text{Variance}} = \frac{6.42}{\text{as an example}}$$

Spread ↑↑

$$\frac{\sigma^2}{\text{Variance}} = 2.168$$

Variance ↑↑

Spread ↑↑



Standard deviation

$$\sigma = \sqrt{\text{Variance}} = \sqrt{2.168}$$

$$= \sqrt{1.472}$$

Variance

\downarrow

Spreadness

1, 2, 2, 3, 4, 5

2.83

-1.472

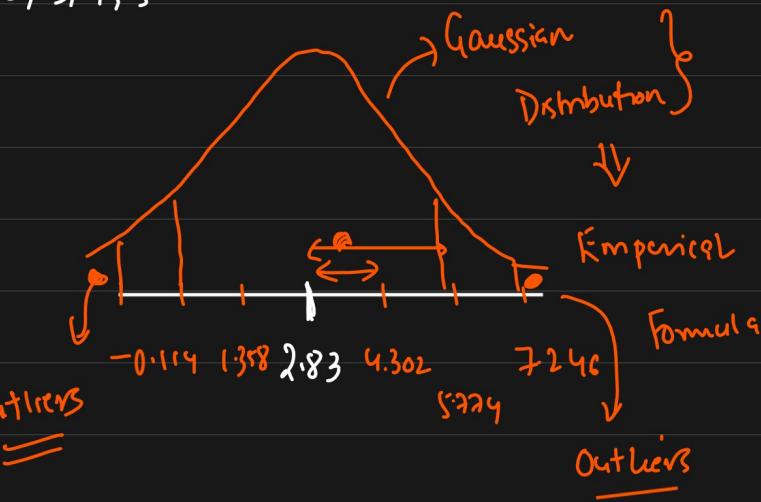
$\frac{1.358}{1.358}$

1.358

1.472

$\frac{1.358}{0.114}$

0.114



$$\begin{array}{r} 1 \\ 2.83 \\ 1.472 \\ \hline 4.302 \end{array}$$

$$\begin{array}{r} 1 \\ 1.472 \\ \hline 5.774 \end{array}$$

$$\begin{array}{r} 1 \\ 1.472 \\ \hline 7.246 \end{array}$$

(*) Percentiles And Quartiles



Percentiles : 1, 2, 3, 4, 5

% of the numbers that are odd?

$$\% \text{ of odd} = \frac{3}{5} = \underline{\underline{60\%}}$$

Percentiles : $\{CAT, GATE, SAT\} \Rightarrow \underline{\underline{99\%}}$

Defn : A percentile is a value below which a certain percentage of observations lie

99 percentiles mean the person has got better marks than 99% of the students.

Data set : 2, 2, 3, 4, 5, 5, 5, 6, 7, 8, 8, 8, 8, 8, 9, 9, 10, 11, 11, 12

What is the percentile ranking of 10? $n=20$

Percentile Rank of $x = \frac{\text{# of values below } x}{n} \times 100$

$$= \frac{16 + 0.8}{20} = \underline{\underline{80}} \text{ percentile}$$

$$= \frac{17}{20} = 85$$

② What value exists at percentile ranking of 25%?

$$\text{Value} = \frac{\text{Percentile} \times (n+1)}{100}$$

$$= \frac{25}{100} \times (21) = \underline{\underline{5.25}} \rightarrow \text{Index}$$

Value = 5

Quartiles (25%)

Five Number Summary

- ① Minimum
- ② First Quartile (25%) Q_1
- ③ Median
- ④ Third Quartile (75%) Q_3
- ⑤ Maximum

Removing the Outliers

Inter Quartile Range: (75% - 25%)
 $Q_3 - Q_1$

$$\{1, 2, 2, 2, 3, 3, 4, 5, 5, 5, 6, 6, 6, 6, 7, 8, 8, 9, \cancel{10}\}$$

[Lower Fence \longleftrightarrow Higher Fence]

$$\text{Lower Fence} = Q_1 - 1.5(\text{IQR}) \quad (25\%) \quad Q_1 = \frac{3+5}{2} \times 20^{\text{th}} \text{ index}$$

$$\text{Higher Fence} = Q_3 + 1.5(\text{IQR})$$

$$\text{IQR} = Q_3 - Q_1 = 7 - 3 = 4 \quad (75\%) \quad Q_3 = \frac{7+8}{2} \times 20^{\text{th}} = 15^{\text{th}} \text{ index}$$

$$Q_3 = 7$$

$$\text{Lower Fence} = 3 - 1.5(4) = 3 - 6 = -3$$

$$\text{Higher Fence} = 7 + 1.5(4) = 7 + 6 = 13$$

$$[-3 \longleftrightarrow 13] \quad -\text{ve} \quad \underline{\underline{3}}$$

$$\text{Remaining} \quad \frac{5+5}{2} = 5$$

$$1, 2, 2, 2, 3, 3, 4, 5, 5, 5, 6, 6, 6, 6, 7, 8, 8, 9, \cancel{10}$$

5 Number Summary

Minimum = 1

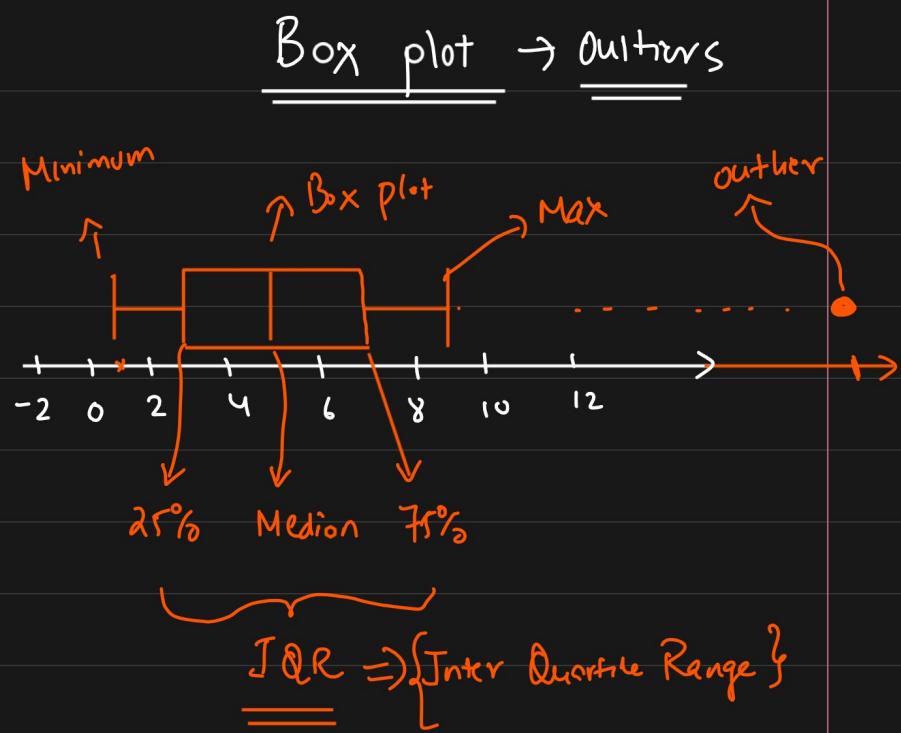
$Q_1 = 3$

Median = 5

$Q_3 = 7$

Max = 9

Use of Box plot



① Distributions

① Normal / Gaussian Distribution ✓

② Standard Normal Distribution ✓

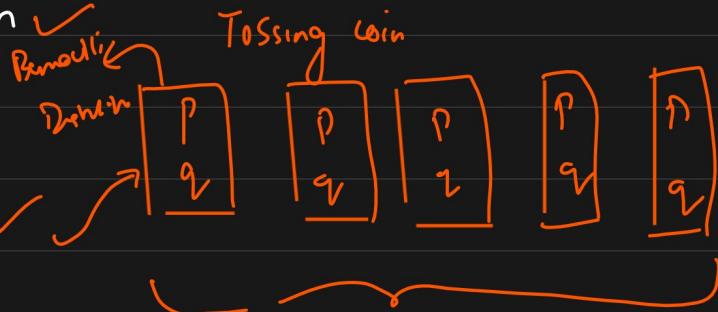
③ Z-score ✓

④ Log Normal Distr ✓

⑤ Bernoulli's Distribution ✓

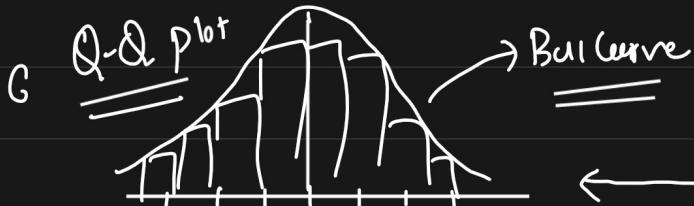
⑥ Binomial Distribution }

① Gaussian / Normal Distribution



Properties

{ Power Law }



① Empirical Rule of
Gaussian Distribution \Downarrow
80-20%

↳ DATASET → IRIS Dataset } → Petal, Sepal length
domain expansion }

② Weight of human brain

③ Height → Doctor

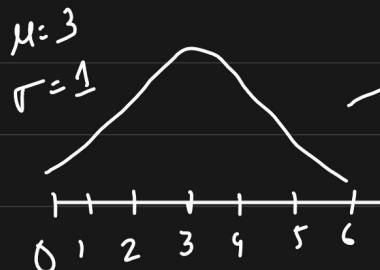
68.2, -95.4 - 99.7

Outliers

Standard Normal Distribution

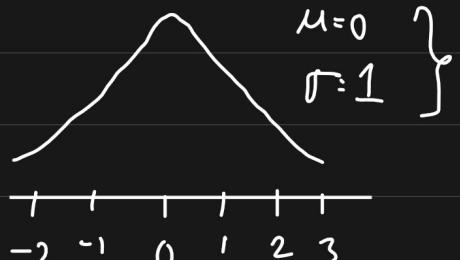
{1, 2, 3, 4, 5}

$$\mu = 3$$



$$\sigma = 1.414 \approx 1$$

$$\Rightarrow \begin{cases} \mu = 0 \\ \sigma = 1 \end{cases}$$



{1, 2, 3, 4, 5}

$$\left\{ Z\text{-Score} = \frac{x-\mu}{\sigma} \right\}$$

=

$$\begin{aligned} & \frac{3-3}{1} = 0 & & = \frac{1-3}{1} \\ & \frac{2-3}{1} & & = -2 \end{aligned}$$

Why 22

$$\begin{cases} \mu=0 \\ \sigma=1 \end{cases}$$

✓

Standardization vs Normalization

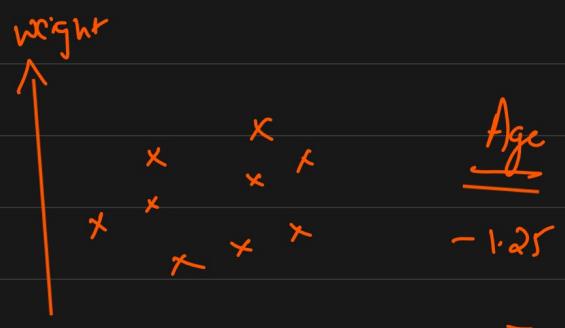
Years ↑
Age ↗ Different unit
Weight ↗ kg

25
26
28
30
32
un

INR
Salary
25K
30K
40K
80K
un

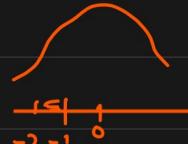
$$\frac{25 - 28.2}{25.6}$$

Same unit scale ??



Maths → Scale

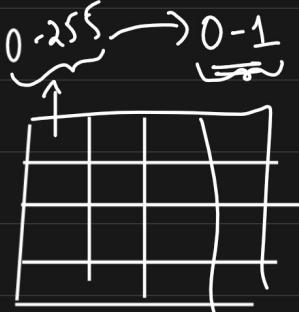
Standardization



Normalization

[Min Max Scalar]

$$\begin{matrix} \downarrow \\ 0 \text{ to } 4 \end{matrix}$$
$$\left. \begin{matrix} \downarrow \\ 0 \text{ to } 1 \end{matrix} \right\}$$



Convolutional

Neural Netw.

ML Disease
✓
Standardization

g

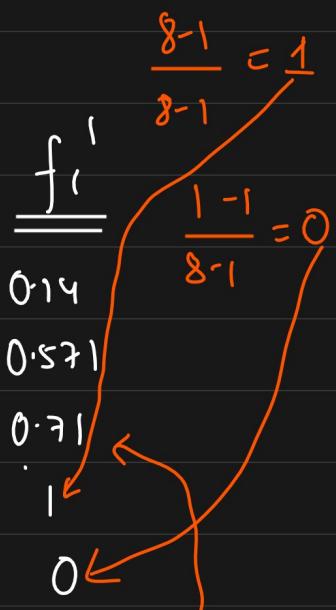
Normalization

$\leftarrow \boxed{\text{CNN}}$

f1

Normalization

(0 - 1)



$$\left\{ \begin{array}{l} x_{\text{Norm}} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \\ \parallel \end{array} \right.$$

Min Max Scalar

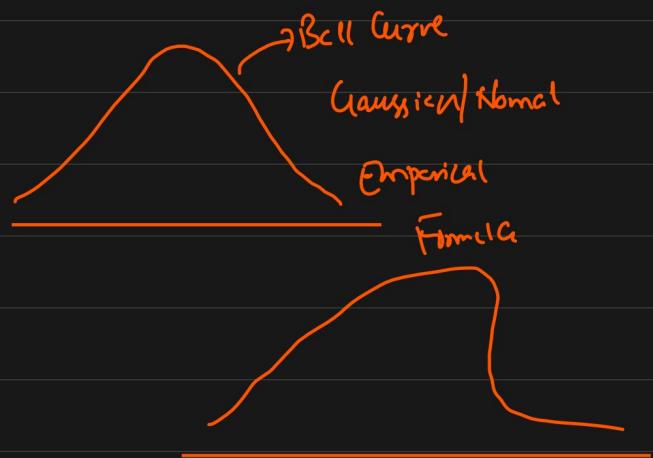
 = 0 to 1

$$= \frac{2 - 1}{8 - 1} = \frac{1}{7} = 0.142$$

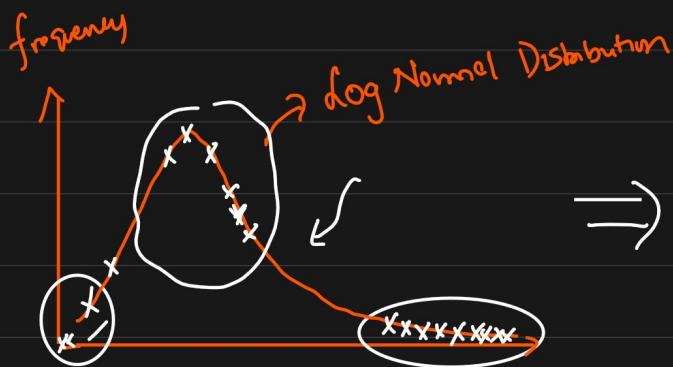
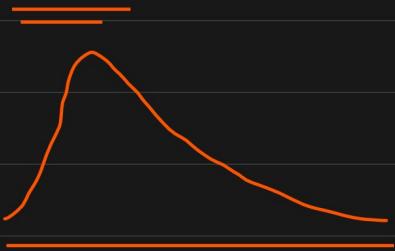
$$\frac{6 - 1}{8 - 1} = \frac{5}{7}$$

$$\frac{5 - 1}{8 - 1} = \frac{4}{7} = 0.571$$

Log Normal Distribution

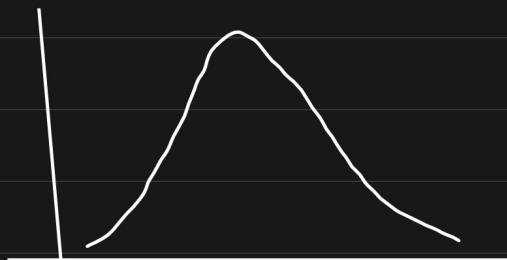


Skewed Curve



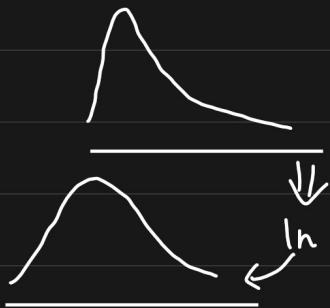
Gaussian Distribution

Normal Distn.



$X = \text{log Normal Distributed}$

$$\left\{ Y = \ln(X) \right\} \quad \begin{array}{l} \text{Gaussian} \\ \text{Distribution} \end{array}$$



$$\left\{ X = \exp(Y) \right\} \rightarrow c^y$$

X

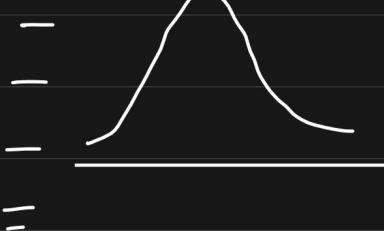
$\mathcal{Y} = \ln(x)$

25

30

40

45

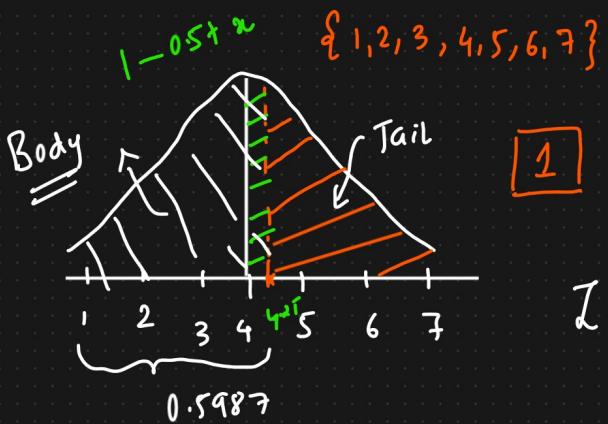


① Bernoulli's Distribution

Day 2 - Stats

$$\textcircled{1} \quad Z\text{-Score} = \frac{x_i - \mu}{\sigma}$$

Stats Interview Question



How many standard deviation

4.25 fall from the mean??

$$Z\text{-Score} = \frac{x_i - \mu}{\sigma} = \frac{4.25 - 4}{1} = 0.25$$

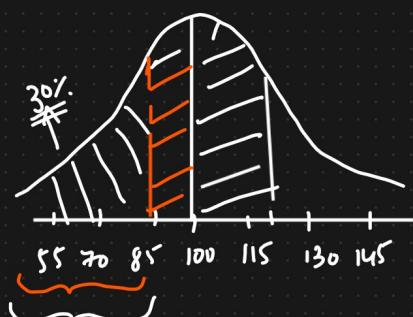
Question : What percentage of scores fall above 4.25?

$$1 - 0.59871 = 0.4013 \Rightarrow 40.13\%$$

2 In India the average IQ is 100, with a standard deviation of 15.

What is the percentage of the population would you expect to have an IQ lower than 85?

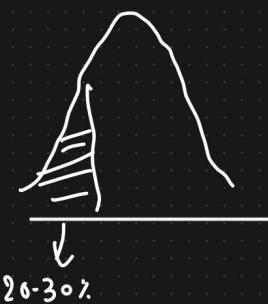
Ans)



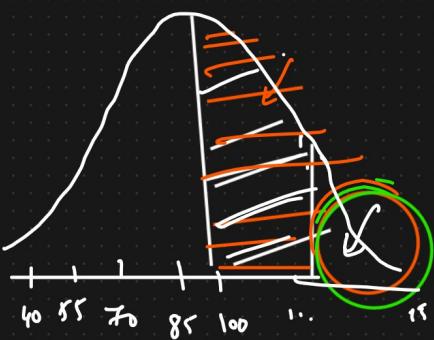
$$Z\text{-Score} = \frac{85 - 100}{15} = \frac{-15}{15} = \boxed{-1}$$

① Area under this curve

$$0.5 - 0.15866 = 0.34143 \Rightarrow \boxed{34.14\%}$$



$$\{ \text{Growth} = 100 \text{ less than } 125 \}$$

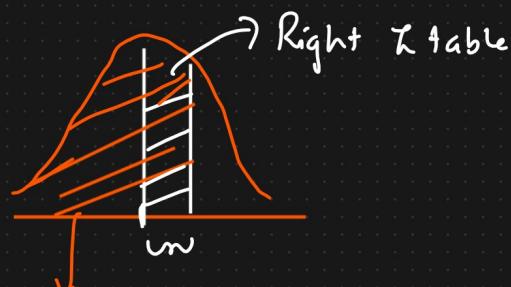


$$Z\text{score} = \frac{125 - 100}{15} = \frac{25}{15} = 1.65$$

$$\text{Ans} = 0.4515 \Rightarrow 45.15\%$$

1.65 ↗

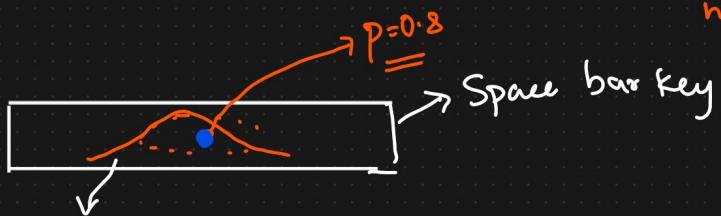
$$\underline{0.5 - 0.4515 = 0.0485} \Rightarrow 4.8\%$$



Left Z-table

P value, Hypothesis Testing, Confidence Interval

Out of all 100 touches, the no. of touches is 80



$$P=0.4$$

Out of all 100 touches, the no. of times 40 times.

Hypothesis Testing, C.I., Significance value Together Fair Coin

Coin → Test whether the coin is a fair coin or not by performing 100 tosses

$$\begin{array}{c} P(H) = 0.5 \\ = \\ P(T) = 0.5 \end{array}$$

Hypothesis Testing

Criminal is \rightarrow Court

SMOLAY

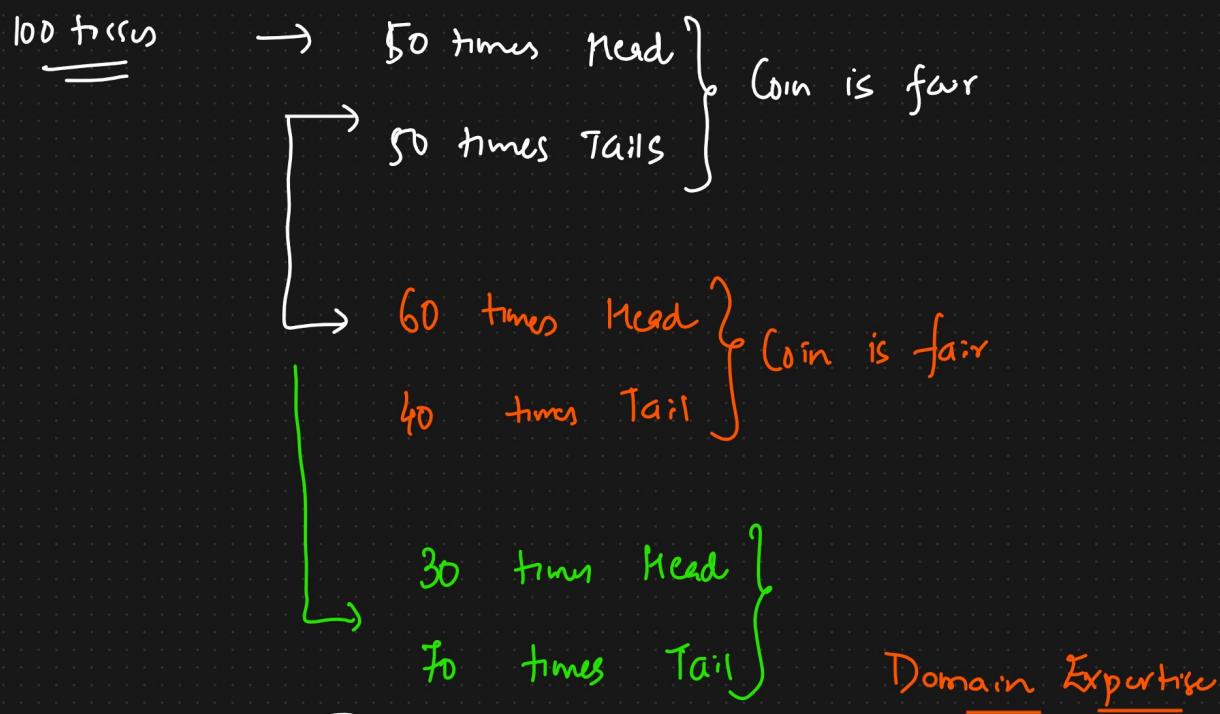
$$P(H) = 100\% \quad P(T) = 0\%$$

① Null Hypothesis — Coin is fair $\rightarrow (H_0)$

② Alternative Hypothesis — Coin is not fair $\rightarrow (H_1)$

③ Experiments

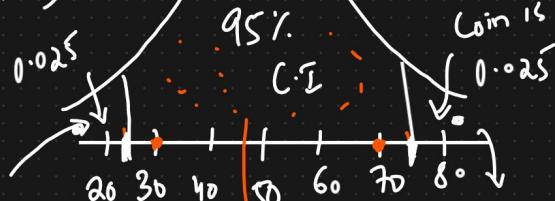
④ Reject or Accept the Null Hypothesis



Confidence Interval, Significance Values

$$C.I. = 1 - 0.025 - 0.025$$

$$= 0.95 \geq 95\%$$



Reject

Null Hypothesis \rightarrow 10 Head

Reject

Null Hypothesis

90 Null Hypothesis

Null Hypothesis

Health Care

Covid vaccine Test

Fever $\downarrow \downarrow$

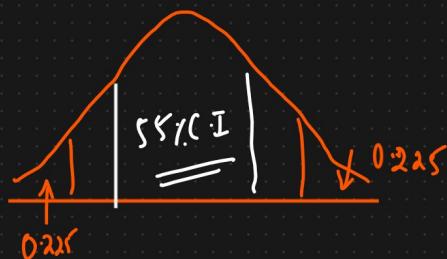
Significance Value $= 0.05$

$$\lambda = 0.45$$

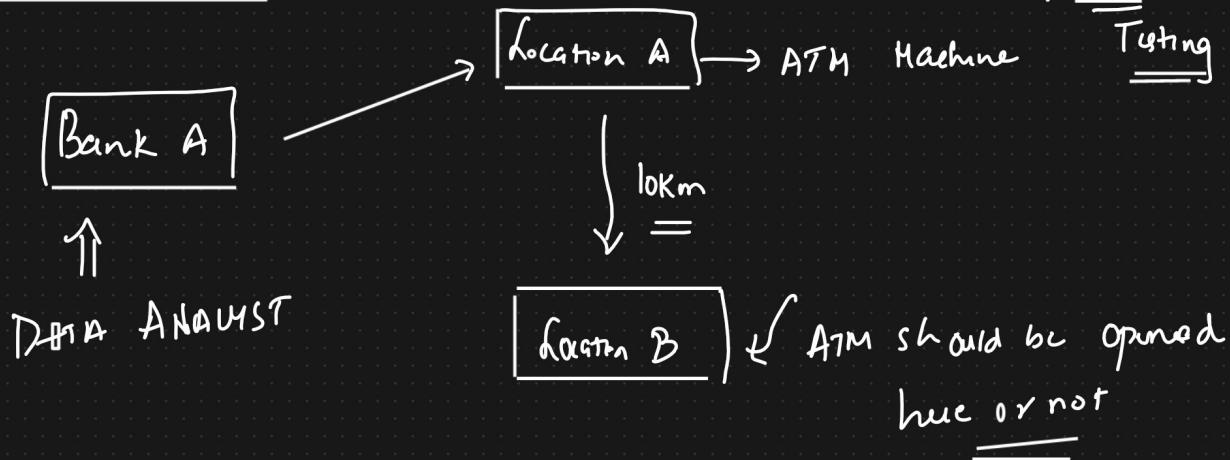
Medical

$f \uparrow \uparrow$

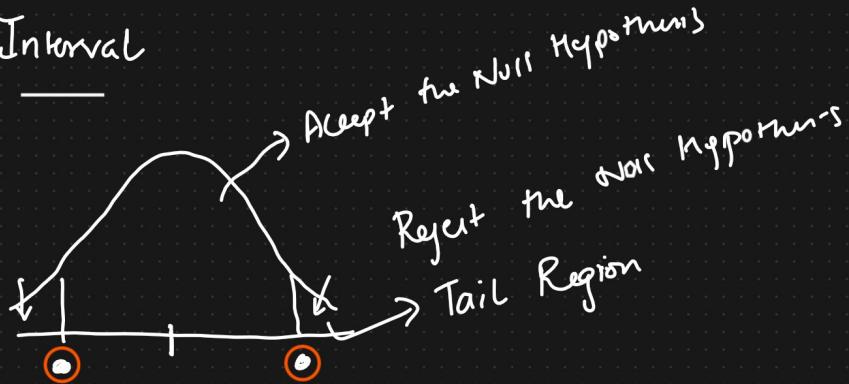
$$\frac{0.45}{2} = 0.225$$



Real World Project

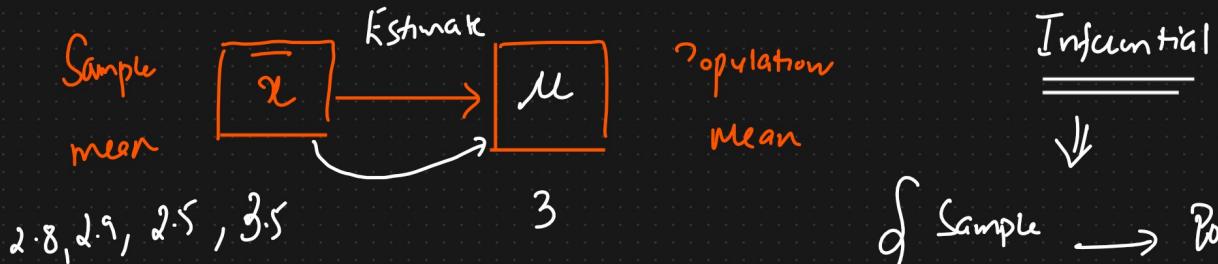


① Confidence Interval



Point Estimate

{ The value of any statistic that estimates the value of a parameter is called Point Estimate.

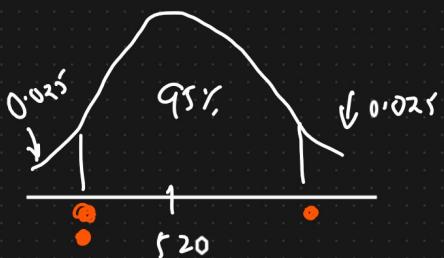


Confidence Interval

t test Point Estimate \pm Margin of Error \Rightarrow Population.

- Q) On the quant test of CAT Exam, the standard deviation is known to be 100. A sample of 25 test takers has a mean of 520. Construct 95% CI about the mean?

$$\text{Ans) } \sigma = 100 \quad n = 25 \quad \bar{x} = 520 \quad (\cdot I = 95\%) \quad \alpha = 0.05$$



① Population std is given {Z score} \rightarrow Z-table

Point Estimate \pm Margin of Error \Rightarrow C.I. =

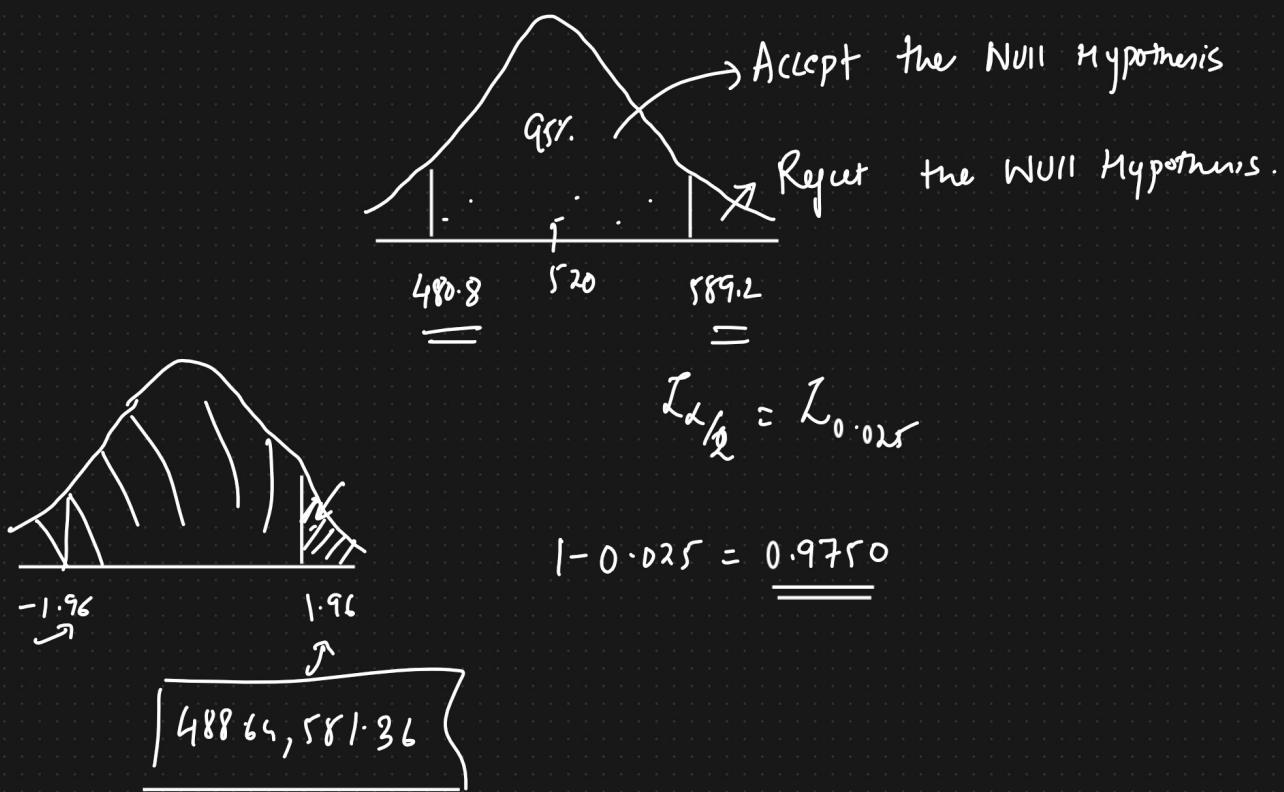
$$\bar{x} \pm Z_{\alpha/2} \left[\frac{\sigma}{\sqrt{n}} \right] \rightarrow \text{Standard Error}$$

$$\text{Lower fence C.I.} = \bar{x} - Z_{\alpha/2} \left[\frac{\sigma}{\sqrt{n}} \right] \Rightarrow Z_{0.05} = 1.96$$

$$\text{Higher fence C.I.} = \bar{x} + Z_{\alpha/2} \left[\frac{\sigma}{\sqrt{n}} \right]$$

$$\text{Lower fence} = 520 - (1.96) \times \frac{100}{\sqrt{25}} = 520 - (1.96) \times 20 = 480.8$$

$$\text{Higher fence} = 520 + (1.96) \times 20 = 559.2$$



- ④ On the quant test of CAT exam, a sample of 25 test-takers has a mean of 520 with a sample standard deviation of 80. Construct 95% C.I about the mean? 2

$$\text{Ans) } \bar{x} = 520 \quad S = 80 \quad f = 0.05 \quad n = 25$$

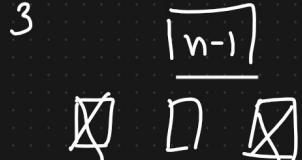
t -test $\Rightarrow t$ - table { Because population Sd is not given }

$$\bar{x} \pm t_{\alpha/2} \left(\frac{S}{\sqrt{n}} \right) \rightarrow \text{Standard Error}$$

$$t_{0.025}$$

t -test

$$\textcircled{1} \text{ Degree of freedom} = n-1 = 25-1 = 24 \quad \underline{\underline{=}}$$



3 people

$$\bar{x} \pm 2.064 \left(\frac{80}{5} \right) \Rightarrow 486.976 \leftrightarrow 553.024$$

- (f) Type 1 and Type 2 Error.
- (g) One Tailed vs 2 Tailed Test

Type 1 and Type 2 Error

Reality Check

$H_0 \Rightarrow$ Coin is Fair

① Null Hypothesis is True or Null

$H_1 \Rightarrow$ Coin is not Fair

Hypothesis is False

Outcome 1:

Decision of Experiments?

We reject the Null Hypothesis Null Hypothesis is True or False.

in reality if it is false \rightarrow Yes



Null Hypothesis



$H_0 \rightarrow$ The Criminal is not guilty

$H_1 \rightarrow$ " " is guilty

Outcome 2:

We reject the Null Hypothesis

when in reality it is true \Rightarrow No \Rightarrow Type 1 Error X

Outcome 3:

We accept the Null Hypothesis, \Rightarrow Type 2 Error X

When in reality it is false

Confusion Matrix

Outcome 4: We accept the Null Hypothesis

when in reality it is True



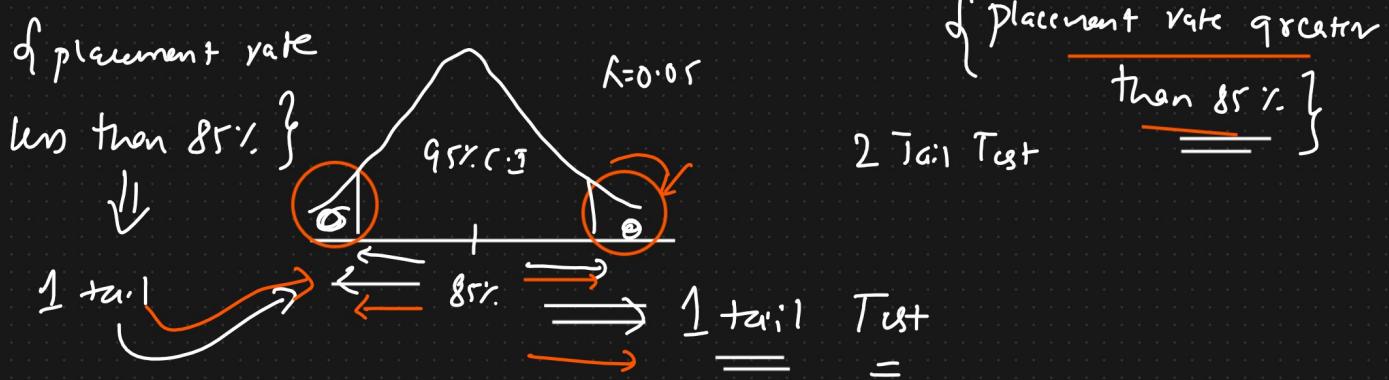
$\begin{bmatrix} \downarrow \\ \text{Cancer} \\ \text{True} \end{bmatrix} \rightarrow \underline{\text{Not Cancer}}$

{ \rightarrow Stock market is going to crash }

② 1 Tail and 2 Tail Test

Eg: College is Karnataka has an 85% placement rate. A new college was recently opened and it was found that a sample of 150 students had a placement rate of 88%. With a standard deviation of 4%. Does this college has a different placement rate?

$$\alpha = 0.05 \Rightarrow 95\% \text{ C.I} \rightarrow 85\%$$



- ① Z test Hypothesis Testing
 - ② J Test Hypothesis Testing
 - ③ Significance value of P value.
 - ④ ANOVA TEST
 - ⑤ CHI SQUARE TEST
 - ⑥ Practical
- Saturday $\frac{10 \text{ min probability}}{\text{EDA} \rightarrow 3-4 \text{ projects}}$ Sunday
- Machine Learning

Statistics

{ 11:30 - 12pm }

- ① Covariance
- ② Pearson Correlation Coefficient
- ③ Spearman Rank Correlation Coefficient
- ④ CHI SQUARE TEST
- ⑤ ANNOVA (F-Test)

✓ practicals
✓

Covariance

$x \uparrow \quad y \uparrow$

$\downarrow \quad \quad \quad \downarrow$
 $x =$

$y =$

{ quantity the relationship

between $x \& y$ }

$x \uparrow \quad y \downarrow$

-

-

$x \downarrow \quad y \uparrow$

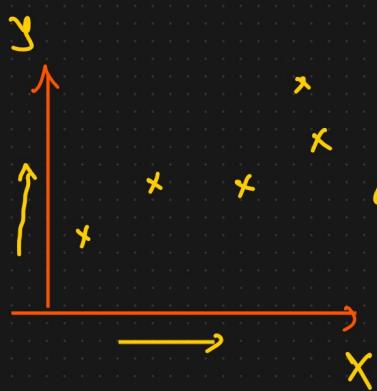
-

-

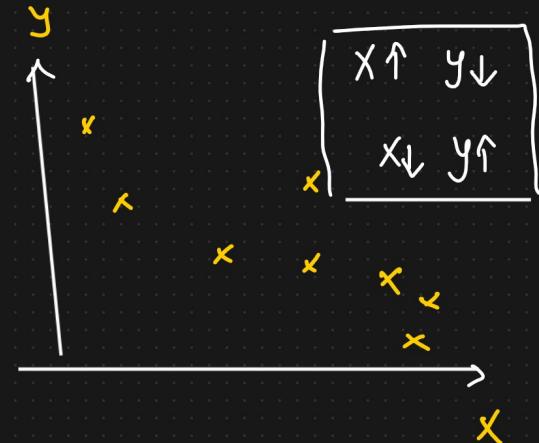
$x \downarrow \quad y \downarrow$

-

-



$\left\{ \begin{array}{l} x \uparrow \quad y \uparrow \\ x \downarrow \quad y \downarrow \end{array} \right.$



$$\text{Cov}_{x,y} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{N-1} \Leftrightarrow \text{Var}_x(x) = \frac{\sum (x_i - \bar{x})^2}{N-1}$$

$\text{Cov}(x, y)$

\Downarrow

$$\text{Cov}(x, x) = \frac{\sum (x_i - \bar{x})^2}{N-1}$$

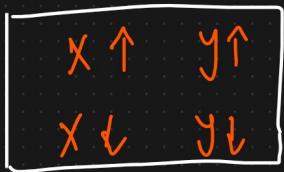
$$= \frac{\sum (x_i - \bar{x}) \times (x_i - \bar{x})}{N-1}$$

$$\text{Var}(x) = \sum_{i=1}^n \frac{(x_i - \bar{x})^2}{n-1} \Rightarrow \sum_{i=1}^n \frac{(x_i - \bar{x})(x_i - \bar{x})}{n-1}$$

↓

$$\text{Cov}(x, x) = \sum_{i=1}^n \frac{(x_i - \bar{x})(x_i - \bar{x})}{n-1}$$

+ve $\Rightarrow \Rightarrow \Rightarrow \Rightarrow$
 \Rightarrow Positively Correlation

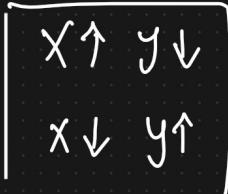


$$\text{Cov}(x, y) = \sum_{i=1}^n \frac{(x_i - \bar{x})(y_i - \bar{y})}{n-1} \quad \left. \right\}$$

$$\left. \begin{array}{l} \\ = (2-4)(3-5) + (4-4)(5-5) \\ \quad + (6-4)(7-5) \end{array} \right. \overline{x} = 4 \quad \overline{y} = 5$$

2

$$= \frac{(-2)(-2) + 0 + (2)(2)}{2} = \frac{8}{2} = 4$$



\Rightarrow -ve Correlation \Rightarrow -ve value.

Disadvantage Covariance

$\text{Cov}(x, y) \Rightarrow$ +ve value
 or -ve value

↓

Relationship $\left[\begin{matrix} -1 & \rightarrow & 1 \end{matrix} \right]$

$$\text{Cov}(x, y) = 500$$

$$\text{Cov}(y, z) = 600$$

Limit	-400
f 500	-300
-400	f 1000

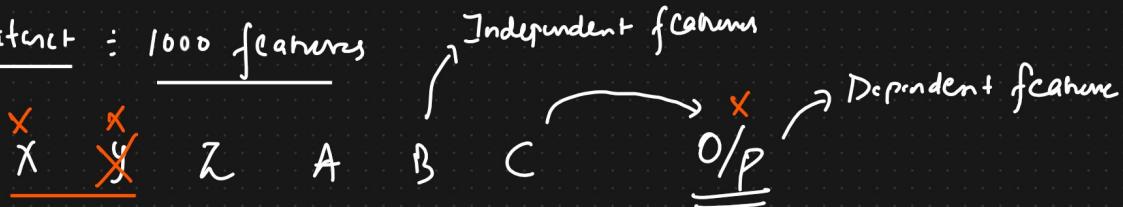
$\boxed{\infty}$

② Pearson Correlation Coefficient

$$r_{x,y} = \frac{\text{Cov}(x, y)}{\sigma_x \sigma_y} \quad [-1 \text{ to } 1]$$

The more the value towards 1 more the it is correlated

Dataset : 1000 features



+ve correlated

$$x, y \Rightarrow 99\% \quad \underline{=}$$

$$\underline{90\%} \quad \underline{0.9}$$

-ve correlation

↓
Keep it

③ Spearman Rank Correlation

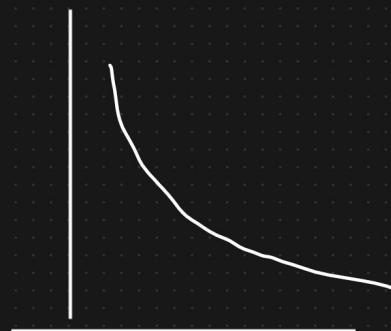
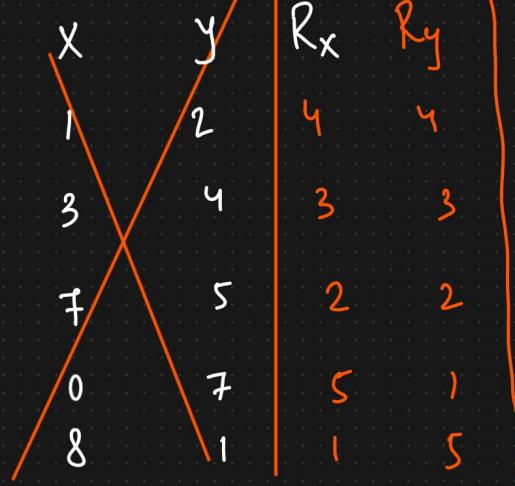
$$r_s = \frac{\text{Cov}(R(x), R(y))}{\sqrt{R(x)} \sqrt{R(y)}}$$

Marks



Spearman Rank

$$\text{Corr} = \underline{1}$$



$$\underline{-1}$$

(f) Chi Square

The Chi Square Test claims about population proportions.

It is a non parametric test that is performed on categorical (nominal or ordinal) data.

- f) In the 2000 U.S Census, the ages of individuals in a small town were found to be the following.

↓	↓	↓
<18	18-35	>35
20%	30%	50%

In 2010, ages of $n=500$ individuals were sampled. Below are the results

<18	18-35	>35
121	288	91

Using $\alpha = 0.05$, would you conclude the the population distribution of ages has changed in the last 10 years?

Ans)

Expected	<18	18-35	>35
20%	30%	50%	$95\% \text{ C.I}$

$n=500$

Observed : 121 288 91

Expected 100 150 250

① H_0 = the data meets the expected distribution
 H_1 = the data does not meet the expected distn

② State Alpha $\therefore \alpha = 0.05$

③ Calculate the degree of freedom

$$df = n - 1 = 3 - 1 = 2 \Rightarrow 3 \text{ categories.}$$

④ Decision Chi Square Table.

If χ^2 is greater $\underline{\underline{5.99}}$ than, Reject H_0

⑤ Calculate Chi square Test

$$\chi^2 = \sum \frac{(f_o - f_e)^2}{f_e} = \frac{(121 - 100)^2}{100} + \frac{(288 - 150)^2}{150} + \frac{(91 - 250)^2}{250} \\ \chi^2 = 232.494$$

$232.494 > 5.99$ Reject the null hypothesis.

⑥ A school principal would like to know which days of the week students are most likely to be absent. The principal expect the students will be absent equally during the 5-day school week. The principal selects a random sample of 100 teachers asking them which day of the week they had the highest number of

Student absences. The Observed and expected results are shown in the table below. Based on these results, do the days for the highest number of absences occur with equal frequencies (use 95% C.I.)

	Monday	Tuesday	Wednesday	Thursday	FRIDAY
Observed	23	16	14	19	28
Expected	20	20	20	20	20.

$$\text{Ans} = \frac{6.3}{\text{---}} \quad \left\{ \begin{array}{l} \text{Accept the Null Hypothesis} \\ \text{---} \end{array} \right\}$$

Practicals + EDA + Feature Engineering }

Statistics

- ① ANOVA (F-Test) → 1 hour }
 ② RDA → { Solve Some Examples } ↘

ANOVA : { Analysis of Variance }

ANOVA IS a statistical method used to compare the means of 2 or more group

ANOVA :

① Factors ② Levels
 (variables)
Medicine { Dosage } Anxiety reducing { Gender }

0mg	50mg	100mg
\bar{x}	\bar{x}	\bar{x}

factor : Dosage

9	6	3
---	---	---

levels : 0mg, 50mg,

8	6	2
---	---	---

100mg

7	6	2
---	---	---

8	7	3
---	---	---

8	8	3
---	---	---

M = F =

Types of ANOVA

One Way ANOVA : One factor with at least 2 levels, levels are independent.

② Repeated Measures ANOVA - One factor with at least 2 levels, but levels are dependent

Factor	Running Kms
Levles	1
	2

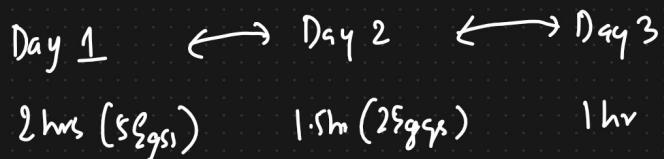


Ques. Study hours of KARTIK

④ Factorial ANOVA



Gym



⑤ Factorial ANOVA : Two or more factor (each of which with atleast 2 levels), levels can be either independent, dependent or both (mixed)

↓ factor

Eq	↓ factor	Day 1 Day 2 Day 3		
		Day 1	Day 2	Day 3
Men	9	7	4	
	8	6	3	
Women	7	5	2	
	8	7	3	
	8	8	4	
	9	7	3	

One Way ANOVA (F -test) \Rightarrow Inferential stats



Comparing means of 2 or more groups

- A) Researchers want to test a new anxiety medication. They split participants into 3 conditions (0mg, 50mg, 100mg), then ask them to rate their anxiety level on scale of 1-10. Are there any differences between the 3 conditions using $\alpha=0.05$?

0mg	50mg	100mg
9	7	4
8	6	3
7	6	2
8	7	3
8	8	4
9	7	3
8	6	2

① $H_0 = \mu_{0\text{mg}} = \mu_{50\text{mg}} = \mu_{100\text{mg}}$ }
 $H_1 = \text{not all } \mu's \text{ are equal}$ }

② State α and C.I

$$\alpha = 0.05 \quad C.I = 95\%$$

③ Calculate the Degree of freedom

$$\rightarrow df_{\text{Between}} = a - 1 = 3 - 1 = 2$$

$$\rightarrow df_{\text{Within}} = N - a = 21 - 3 = 18$$

$$\rightarrow df_{\text{Total}} = N - 1 = 21 - 1 = 20$$

Statistics

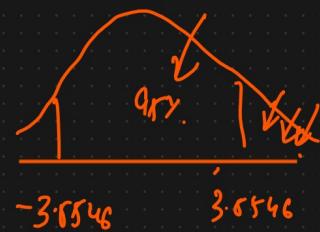
$$N = 21 \quad n = 7$$

$$a = 3 \rightarrow \{ \text{No. of levels} \}$$

④ State Decision Rule

$$df_{\text{Between}} = a - 1 = 3 - 1 = 2 \quad \{(2, 18)\}$$

$$df_{\text{Within}} = N - a = 21 - 3 = 18$$



If F test is greater than 3.8846, Reject the Null Hypothesis

If F test is less than -3.8846 " " " "

⑤ Calculate F Test Statistics

$$F_{\text{test}} = \frac{MS_{\text{between}}}{MS_{\text{within}}} = \frac{49.34}{0.57} =$$

	SS	df	MS	F Test
Between	98.67	2	49.34	86.56
Within	10.29	18	0.57	
Total	108.96	20		

$$SS_{\text{between}} = \frac{\sum (\sum a_i)^2}{n} \quad \overline{T^2} \leftarrow \quad N=21 \quad n=7 \text{ //} \\ T^2 = [57 + 47 + 21]^2 \\ = (125)^2$$

$$\begin{aligned} \sum (\sum a_i)^2 &= (9+8+7+8+8+9+8)^2 + (7+6+6+7+8+7+6)^2 \\ &\quad + (4+3+2+3+4+3+2)^2 \\ &= 57^2 + 47^2 + 21^2 \end{aligned}$$

$$SS_{\text{Between}} = \frac{57^2 + 47^2 + 21^2}{7} - \frac{125^2}{21} = 98.67 = .$$

$$\textcircled{2} \quad SS_{\text{within}} = \sum y^2 - \frac{\sum (\sum a_i)^2}{n}$$

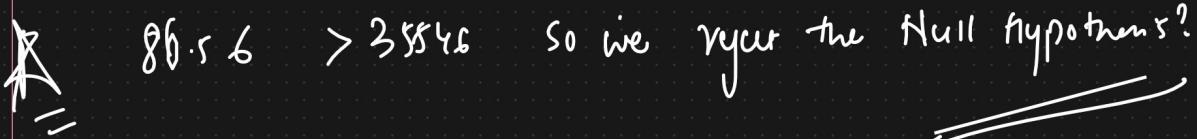
$$\left. \begin{array}{l} P=0.48 \\ d.f.=0.05 \end{array} \right\} = \sum y^2 - \left[\frac{57^2 + 47^2 + 21^2}{7} \right] = 10.29$$

$$\sum y^2 = 9^2 + 8^2 + 7^2 + 8^2 + 8^2 + 9^2 + \dots + 2^2 = 853$$

$\frac{0.75 > 0.05}{\Downarrow}$

Final Conclusion

Accept

 $86.56 > 35846$ So we reject the Null hypothesis?

$$\left. \begin{array}{l} H_0: \mu = \text{Some value} \\ H_1: \mu \neq \text{Some value} \end{array} \right\} \rightarrow 95\% \text{ C.I}$$

Virginia

=

=

Pctz1 width

-

-

-

-

-

-

-

-

-

-

-

-

$$\rightarrow H_0 = \mu_{\text{virgin}} = \mu_{\text{swiss}} = \mu_{\text{...}}.$$

$H_1 = \cdot \neq \text{p-value.} \neq \text{reject the Null Hypothesis}$

$$0.0118 \quad 0.0228 < 0.05 \quad 1 - 0.025 = 0.975$$

$$0.0118$$

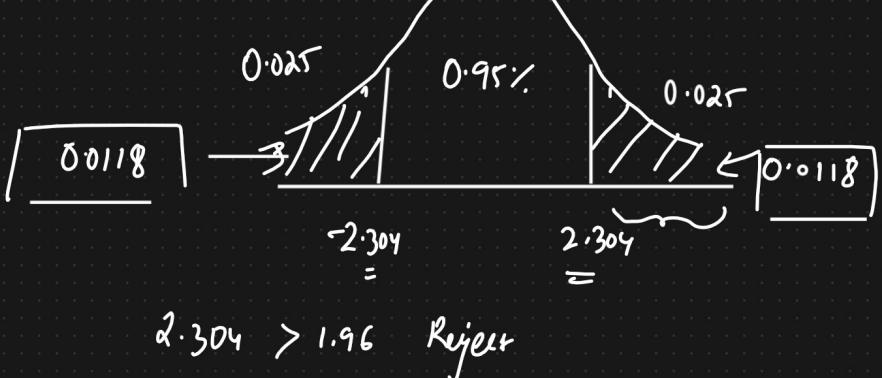
$$0.0228$$

$$0.0228$$

$$d =$$

$$Z_{\text{test statistic}}$$

Z_{test}



$$Z = \frac{\bar{x} - \mu}{\sigma / \sqrt{n}}$$

$$Z = 2.304$$

$2.304 > 1.96 \text{ Reject}$

MongoDB

https://github.com/Chandan220698/Ineuron-DataScience/blob/d527d6cbab1ad6f2403c8bbd5f1e0ff95ec4a0ea/Database%20Assignment/Carbon%20Nanotube%20Task/Carbon_Nanotube%20Task.ipynb

<https://github.com/pallavi176/MongoDBTask.git>

<https://github.com/pk1308/mongodbtask/blob/main/worksheet.ipynb>

<https://github.com/pk1308/mongodbtask>

https://github.com/ckunalgit/live_class/blob/main/FSDS_Assignment_MongoDB.ipynb

https://github.com/sayansaha934/WeeklyTask/tree/main/carbon_nanotube-mongodb

drive link:

<https://drive.google.com/drive/folders/1GHvoRnI4EAOfLW3XKEDbNVYZQy1nFQ8F?usp=sharing>

<https://github.com/LijiAlex/MongoDBImplementationDemo>

<https://github.com/hemraj-shaqawal/FileParserForCarbonNanoApp.git>

https://github.com/pushpavj/Class_assignements/tree/main/MogoDB-project

<https://github.com/sthirumoorthi/Projects/tree/main/Python/MongoDB%20Bulk%20Data%20Load>

https://github.com/madhura-tk/tkmresp/tree/main/carbon_nanotube

Github Link for Code - https://github.com/jithinjk116/iNeuron/blob/main/Tasks_21_02_2022.ipynb

Github link showing the log - https://github.com/jithinjk116/iNeuron/blob/main/carbon_nanotubes.log

https://github.com/AAKGH/AK_Ineuron/tree/main/Bag_Of_Words_Task

https://github.com/nirajkumarm Jain/iNeuron_Task/tree/main/Mongo_DB_task

https://github.com/pushpavj/Class_assignements/tree/main/MogoDB-project

SQLite

<https://github.com/Chandan220698/Ineuron-DataScience/blob/c53c79398b2f0286b3d22f01094a88ed59ec78c7/Database%20Assignment/SQLite%20Task/SQLite%20Task.ipynb>

<https://github.com/pallavi176/SqliteTask.git>

https://github.com/umaretiya/iN_Assignment/blob/main/Assignments_Sun_200220222.ipynb

<https://github.com/pk1308/sqlite/blob/main/worksheet.ipynb>

<https://github.com/pk1308/sqlite>

https://github.com/ckunalgit/live_class/blob/main/FSDS_Assignment_SQLlite3.ipynb

Github Link for Code - https://github.com/jithinjk116/iNeuron/blob/main/Tasks_21_02_2022.ipynb

Github link showing the log - https://github.com/jithinjk116/iNeuron/blob/main/carbon_nanotubes.log

https://github.com/AAKGH/AK_Ineuron/tree/main/Bag_Of_Words_Task

20 Question Task

<https://github.com/pallavi176/FullStackDataScience/blob/main/Task.ipynb>

Bank task

https://github.com/Abhikudale/DataFrame_1/blob/main/Bank%20DataFrame

https://github.com/madhura-tk/ineuron/blob/main/Mar_5_pandas_task/Bank.ipynb

https://github.com/MohamedFaaLil/fsds_exercise_pandas_2/blob/master/bank_data_exercise.ipynb

https://github.com/Manikumar1991/Ineuron_inclass_assig/blob/main/Bank.ipynb

<https://github.com/Chandan220698/Ineuron-DataScience/blob/c58868a8c43c710f95ba784fdc27b16f4253dac/Live%20Class/Data%20Analysis/Live%20Class%20Assignment/Task%203%20-%20Bank.ipynb>

Titanic task

https://github.com/madhura-tk/ineuron/blob/main/Mar_5_pandas_task/titanic.ipynb

<https://github.com/Poornimarthy/titanic-data-task>

https://github.com/ShubhamIad1/Python-Assignments/blob/main/Titanic_Data_oprtations.ipynb

https://github.com/MohamedFaalil/fsds_exercise_pandas_2/blob/master/titanic_data_exercise.ipynb

https://github.com/Manikumar1991/Ineuron_inclass_assig/blob/main/Titanicdataset.ipynb

<https://github.com/Chandan220698/Ineuron-DataScience/blob/c58868a8c43c710f95ba784bfdc27b16f4253dac/Live%20Class/Data%20Analysis/Live%20Class%20Assignment/Task%20%20-%20Titanic%20Excercise.ipynb>

Pallavi task:<https://github.com/pallavi176/Pandas/tree/main/Task>

Both thing in the same link

Vaibhav

task:https://github.com/vaibhavyaramwar/Python_Program_Challenge/tree/main/Pandas

<https://github.com/rkbh1718/pandas-task-05-march/blob/main/pandas%20task%20on%20bank%20dataset%20and%20tanic%20dataset.ipynb>

[https://github.com/Ashutosh0428/pandas_assignment/blob/main/Untitled8%20\(1\).ipynb](https://github.com/Ashutosh0428/pandas_assignment/blob/main/Untitled8%20(1).ipynb)

https://github.com/krishnavizster/iNeuron-Tasks/blob/main/TASK2_IN_PANDAS_5_3_2022.ipynb

vignesh:https://github.com/vtech20/ineuron_assignments/blob/main/ineuron-pandas-assignment.ipynb

https://github.com/Ansfarheen/homework/blob/main/Pandas_Homework_5March22.ipynb

https://github.com/Shilpaltnal23/ClassAssignments/blob/main/05-03-22_Pandas_Assignment.ipynb_shilpa

<https://github.com/Mukesh-areo/Ineuron-tasks/blob/main/Pandas%20task%20.ipynb>

Mukesh

<https://github.com/pallavianand90/Pandas-Assignment>

<https://github.com/vedjangid19/iNeuron-Class-Task/blob/main/bank/05-Mar-2022-Pandas-class-assinement.ipynb>

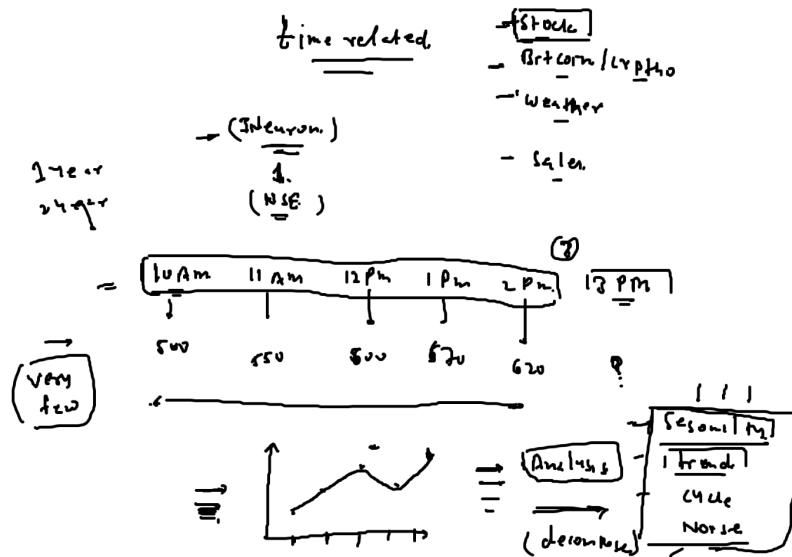
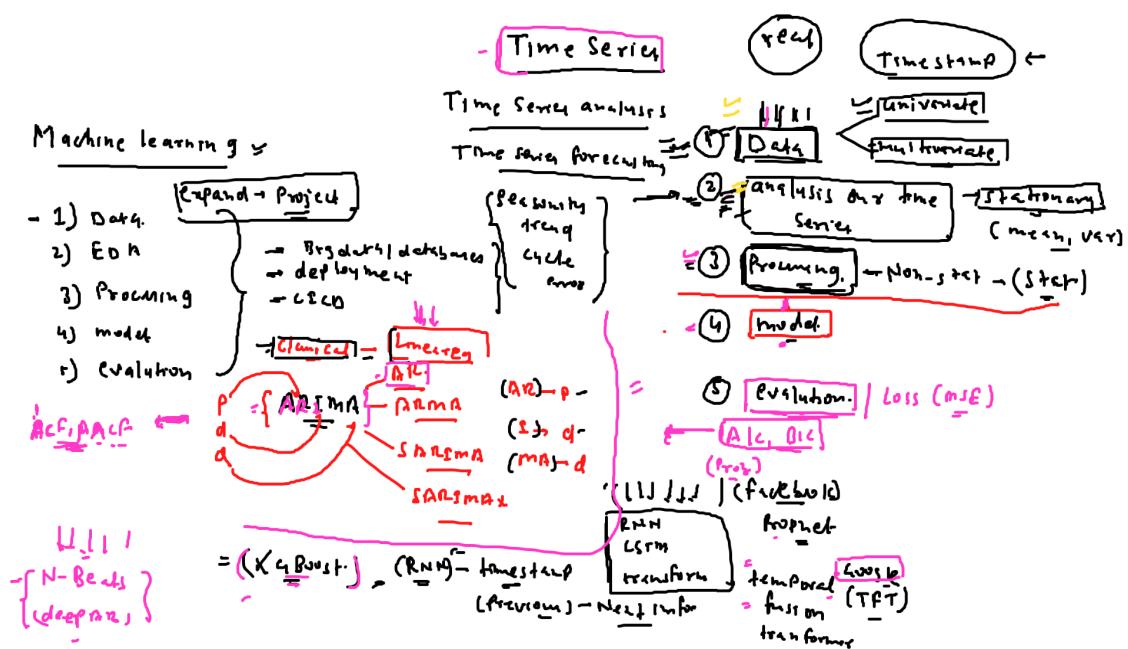
<https://github.com/ritikavijay/my-python-work/blob/4bac4522e61825a7d94c92f184a78e891f14b5e3/pandaschallenge1.ipynb>

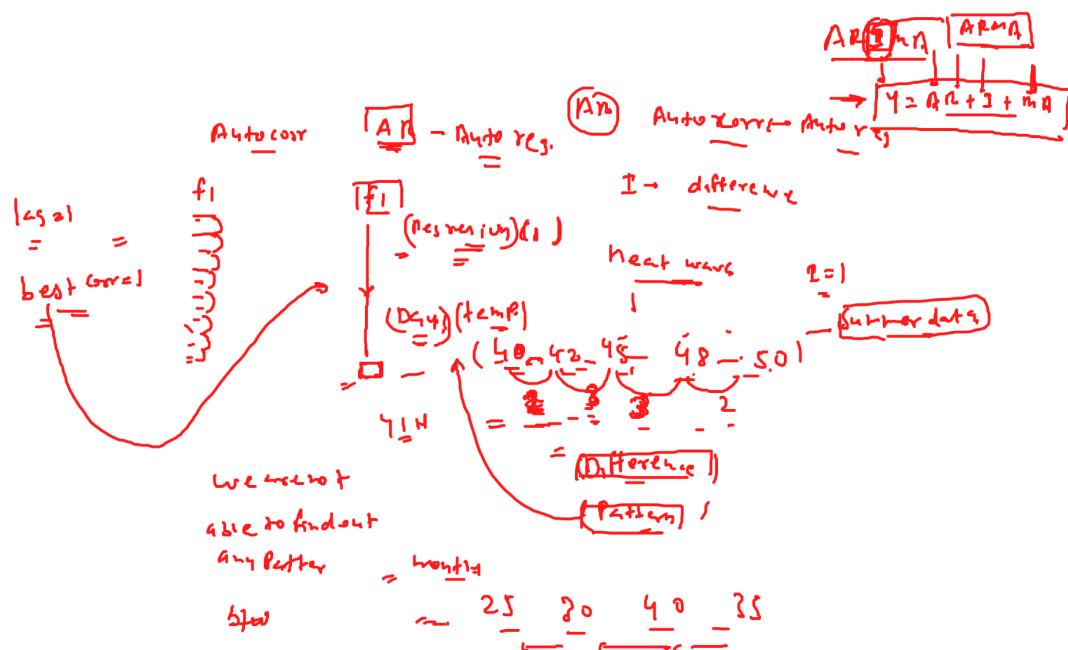
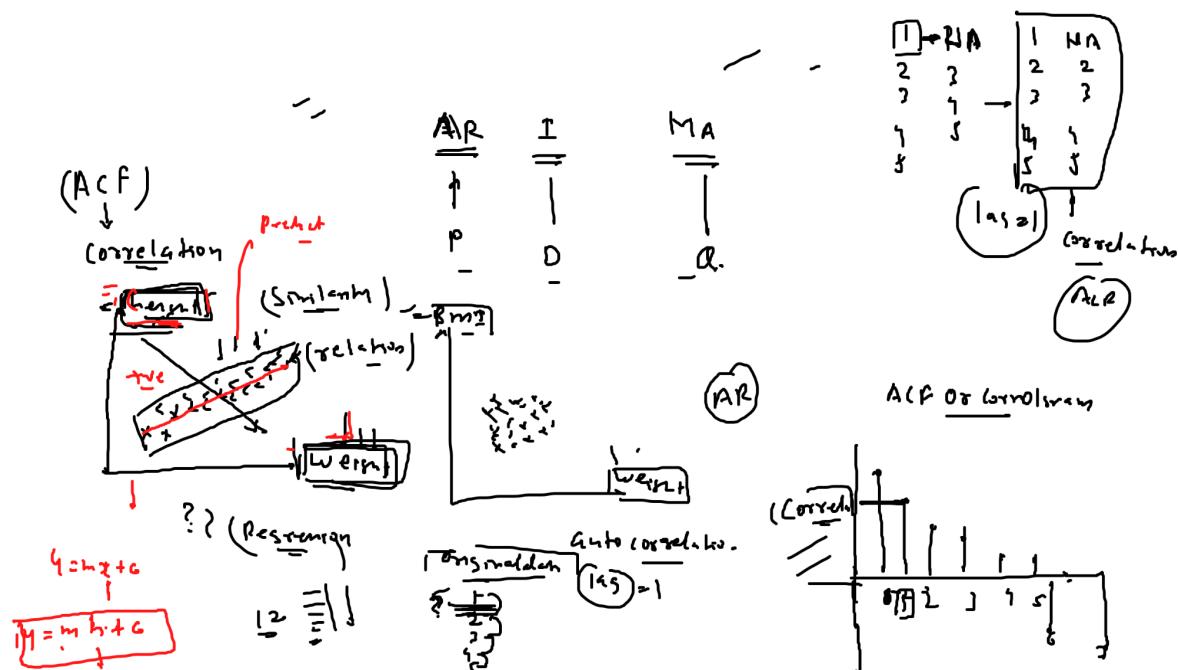
<https://github.com/dembasiby/pandas1>

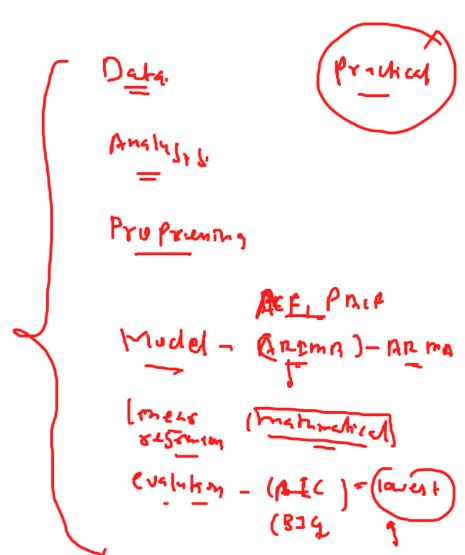
<https://github.com/kinghini/Pandas-hw2.git>

https://github.com/siridi10/Data_Science_Assignment/blob/main/Live%20Class/Data%20Analysis/Pandas%203.ipynb--srividika

https://github.com/codeofelango/Class-Task-pandas-day-3/blob/main/March%206th%20Class%20Task%20pandas_day_3.ipynb--elango







Time Series Analysis

Problem Statement

- PH, a tractor and farm equipment manufacturing company, was established a few years after World War II.
- The company has shown a consistent growth in its revenue from tractor sales since its inception. However, over the years the company has struggled to keep it's inventory and production cost down because of variability in sales and tractor demand.

Problem Statement

- The management at PH is under enormous pressure from the shareholders and board to reduce the production cost.
- Additionally, they are also interested in understanding the impact of their marketing and farmer connect efforts towards overall sales.

Problem Statement

- They have hired us as a data science and predictive analytics consultant.
- We will develop an ARIMA model to forecast sale / demand of tractor for next 3 years.
- Additionally, We will also investigate the impact of marketing program on sales by using an exogenous variable ARIMA model.

Problem Statement

- An endogenous variable is one that **is influenced** by other factors in the system. flower growth is affected by sunlight and is therefore endogenous.

Exogenous variables...

- are fixed when they enter the model.
- are taken as a “given” in the model.
- influence endogenous variables in the model.
- are not determined by the model.
- are not explained by the model.

Problem Statement

- As a part of the project, one of the production units we are analyzing is based in South East Asia.
- This unit is completely independent and caters to neighboring geographies. This unit is just a decade and a half old. In 2014 , they captured 11% of the market share, a 14% increase from the previous year.

Problem Statement

- However, being a new unit they have very little bargaining power with their suppliers to implement Just-in-Time (JiT) manufacturing principles that have worked really well in PH's base location.
- Hence, they want to be on top of their production planning to maintain healthy business margins.

Problem Statement

- Monthly sales forecast is the first step we have suggested to this unit towards effective inventory management.
- The MIS team shared the month on month (MoM) sales figures (number of tractors sold) for the last 12 years

Time Series Analysis

- Time series analysis is extensively used to forecast company sales, product demand, stock market trends, agricultural production etc.
- The fundamental idea for time series analysis is to decompose the original time series (sales, stock market trends, etc.) into several independent components.

Time Series Analysis

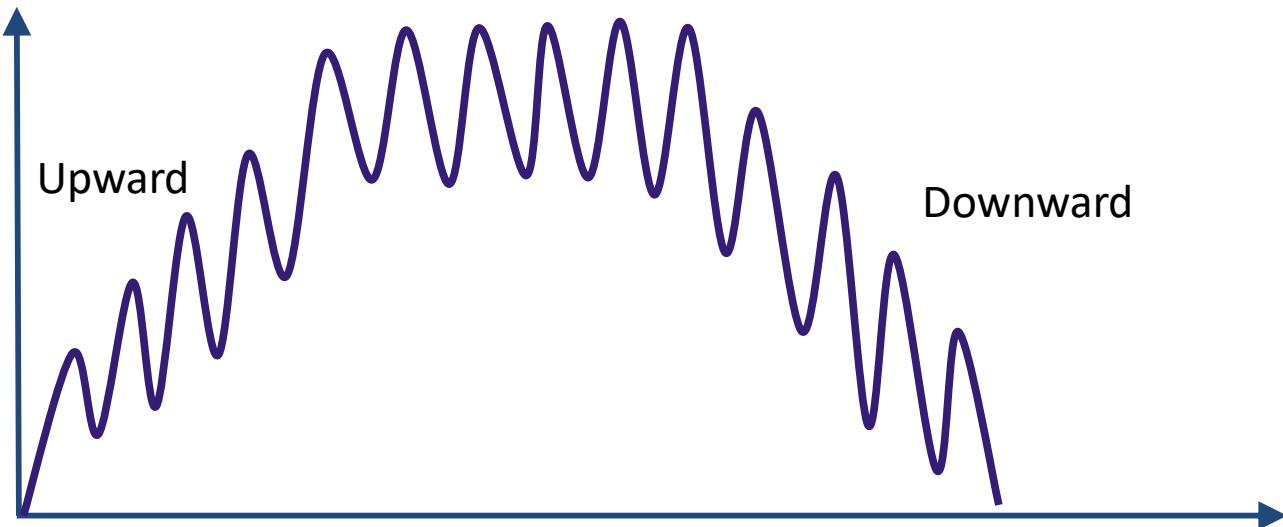
- Typically, business time series are divided into the following four components:
- **Trend** – overall direction of the series i.e. upwards, downwards etc.
- **Seasonality** – monthly or quarterly patterns
- **Cycle** – long-term business cycles, they usually come after 5 or 7 years
- **Irregular remainder** – random noise left after extraction of all the components

Time Series Analysis

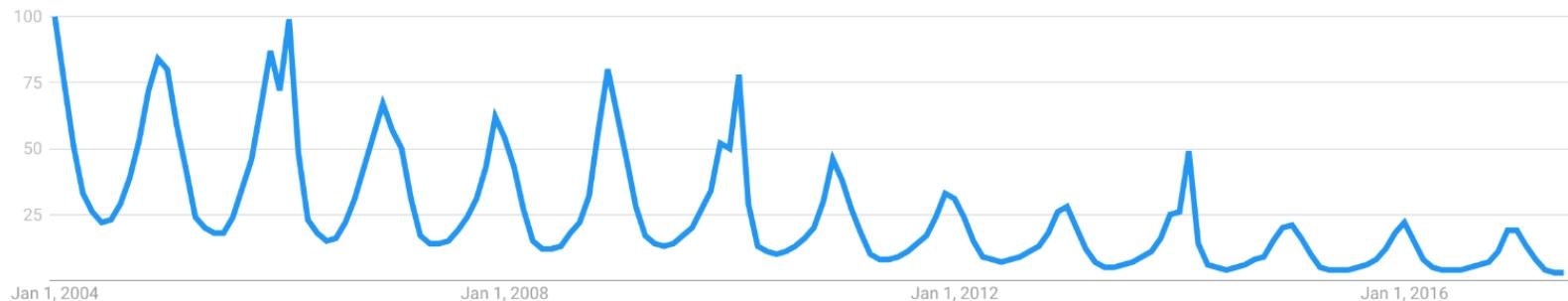
- Interference of these components produces the final series.
- Why decomposing the original / actual time series into components?
- It is much easier to forecast the individual regular patterns produced through decomposition of time series than the actual series.

- Trends

Horizontal/Stationary



- Seasonality - Repeating trends



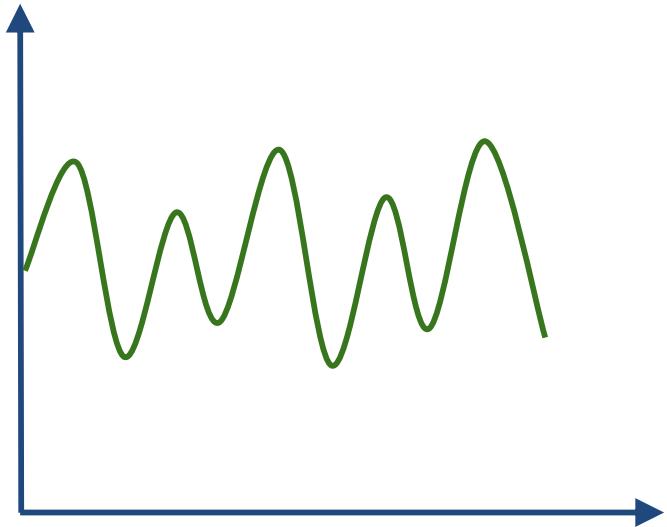
Google Trends - “Snowboarding”

- Cyclical - Trends with no set repetition.

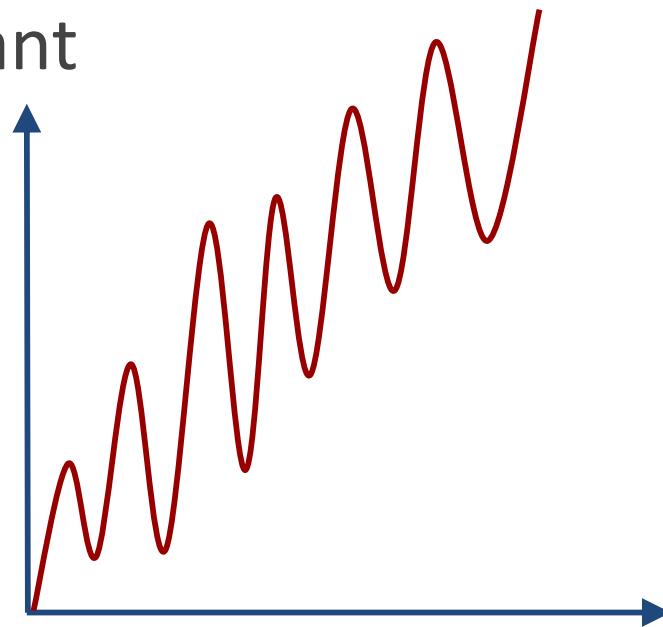


- Stationary vs Non-Stationary Data
 - To effectively use ARIMA, we need to understand Stationarity in our data.
 - So what makes a data set Stationary?
 - A Stationary series has constant mean and variance over time.

- Mean needs to be constant

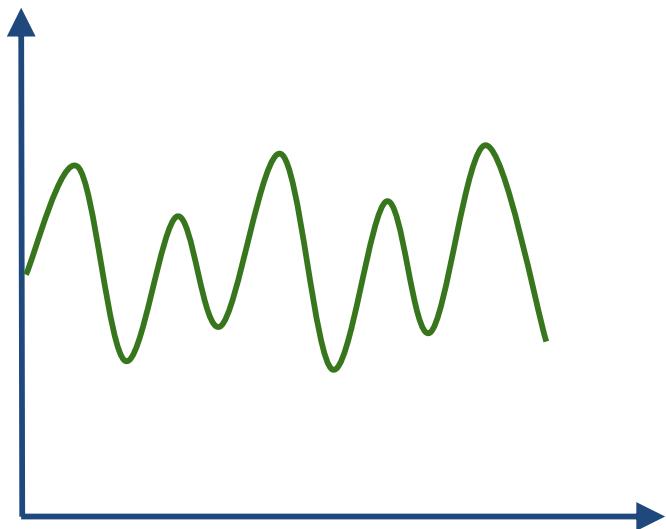


Stationary

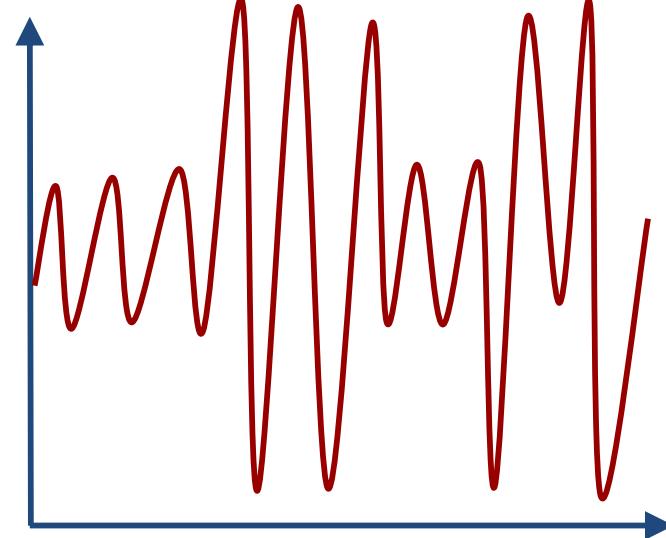


Non-Stationary

- Variance should not be a function of time



Stationary



Non-Stationary

- A Stationary data set will allow our model to predict that the mean and variance will be the same in future periods.

- There are also mathematical tests you can use to test for stationarity in your data.
- A common one is the Augmented Dickey–Fuller test

- If we've determined your data is not stationary (either visually or mathematically), we will then need to transform it to be stationary in order to evaluate it and what type of ARIMA terms you will use.

- One simple way to do this is through “differencing”.

Original Data

Time 1	10
Time 2	12
Time 3	8
Time 4	14
5	

First Difference

Time 1	NA
Time 2	2
Time 3	-4
Time 4	6
Time 5	-7

Second Difference

Time 1	NA
Time 2	NA
Time 3	-6
Time 4	10
Time 5	-13

- You can continue differencing until you reach stationarity (which you can check visually and mathematically)
- Each differencing step comes at the cost of losing a row of data.

- For seasonal data, we can also difference by a season.
- For example, if we had monthly data with yearly seasonality, we could difference by a time unit of 12, instead of just 1.

- With our data now stationary it is time the p,d,q terms and how we choose them.
- A big part of this are AutoCorrelation Plots and Partial AutoCorrelation Plots.

Trend

- From the plots it is obvious that there is some kind of increasing trend in the series along with seasonal variation.
- Stationarity is a vital assumption we need to verify if our time series follows a stationary process or not.

Trend

- We can do by
 - Plots: review the time series plot of our data and visually check if there are any obvious trends or seasonality
 - Statistical tests: use statistical tests to check if the expectations of stationarity are met or have been violated.

Trend using MAs

- Moving averages over time
 - One way to identify a trend pattern is to use moving averages over a specific window of past observations.
 - This smoothens the curve by averaging adjacent values over the specified time horizon (window).

Seasonality

- People tend to go on vacation mainly during summer holidays.
- At some time periods during the year people tend to use aircrafts more frequently. We can check the hypothesis of a seasonal effect

Noise

- To understand the underlying pattern in the number of international airline passengers, we assume a multiplicative time series decomposition model
- Purpose is to understand underlying patterns in temporal data to use in more sophisticated analysis like Holt-Winters seasonal method or ARIMA.

Noise

- Noise - is the residual series left after removing the trend and seasonality components

Stationarize a Time series

- Before models forecasting can be applied, the series must be transformed into a stationary time series.
- The Augmented-Dickey Fuller Test can be used to test whether or not a given time series is stationary.

Stationarize a Time series

- If the test statistic is smaller than the critical value, the hypothesis is rejected, the series would be stationary, and no further transformations of the data would be required.

Residuals Serial Correlation

- When the residuals (errors) in a time series are correlated with each other it is said to exhibit serial correlation.
- Autocorrelation is a better measurement for the dependency structure, because the autocovariacne will be affected by the underlying units of measurement for the observation.

White Noise & ACF & PACF

- Random process is white noise process
- Errors are serially uncorrelated if they are independent and identically distributed (iid).
- It is important because if a time series model is successful at capturing the underlying process, residuals of the model will be iid and resemble a white noise process.

White Noise

- Part of time series analysis is simply trying to fit a model to a time series such that the residual series is indistinguishable white noise.

ACF & PACF

- The plots of the Autocorrelation function (ACF) and the Partial Autorrelation Function (PACF) are the two main tools to examine the time series dependency structure.
- The ACF is a function of the time displacement of the time series itself.
- It is the similarity between observations as a function of the time lag between them.

PACF

- The PACF is the conditional correlation between two variables under the assumptions that the effects of all previous lags on the time series are known.

Random Walk

- What is special about the random walk is, that it is non-stationary, that is, if a given time series is governed by a random walk process it is unpredictable.
- It has high ACF for any lag length
- The normal QQ plot and the histogram indicate that the series is not normally distributed

Random Walk

- The random walk is a first order autoregressive process that is, this causes the process to be non-stationary.
- The process can be made stationary

Auto Regressive Model – AR(p)

- The random walk process belongs to a more general group of processes, called autoregressive process
- The current observation is a linear combination of past observations.
- An AR(1) time series is one period lagged weighted version of itself.

The Moving Average Model - MA(q)

- The moving average model MA(q) assumes that the observed time series can be represented by a linear combination of white noise error terms.
- The time series will always be stationary.

ARIMA Forecasting

- An autoregressive integrated moving average (ARIMA) model is an generalization of an autoregressive moving average (ARMA) model.
- Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting).

ARIMA Forecasting

- ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity.
- There are three parameters (p, d, q) that are used to parametrize ARIMA models. Hence, an ARIMA model is denoted as $\text{ARIMA}(p, d, q)$
- Each of these three parts is an effort to make the time series stationary, i. e. make the final residual a white noise pattern.

Box-Jenkins Approach to non-Seasonal ARIMA Modeling

- In time series analysis, the Box–Jenkins method,[1] named after the statisticians George Box and Gwilym Jenkins, applies autoregressive moving average (ARMA) or autoregressive integrated moving average (ARIMA) models to find the best fit of a time-series model to past values of a time series.

Box-Jenkins Approach to non-Seasonal ARIMA Modeling

- The original model uses an iterative three-stage modeling approach:
- Model identification and model selection:

Box-Jenkins Approach to non-Seasonal ARIMA Modeling

- Making sure that the variables are stationary, identifying seasonality in the dependent series (seasonally differencing it if necessary), and using plots of the autocorrelation and partial autocorrelation functions of the dependent time series to decide which (if any) autoregressive or moving average component should be used in the model.

Box-Jenkins Approach to non-Seasonal ARIMA Modeling

- Parameter estimation using computation algorithms to arrive at coefficients that best fit the selected ARIMA model.
- Model checking by testing whether the estimated model conforms to the specifications of a stationary univariate process.

Box-Jenkins Approach to non-Seasonal ARIMA Modeling

- The residuals should be independent of each other and constant in mean and variance over time.
- If the estimation is inadequate, we have to return to step one and attempt to build a better model.

Optimal Parameter Selection

- To fit the time series data to a seasonal ARIMA model with parameters $\text{ARIMA}(p, d, q)(P, D, Q)s$ the optimal parameters need to be found first.
- This is done via grid search, the iterative exploration of all possible parameters constellations.

Optimal Parameter Selection

- Depending on the size of the model parameters $\$(p, d, q)(P, D, Q)s\$$ this can become an extremely costly task with regard to computation. We start of by generating all possible parameter constellation we'd like to evaluate.

Akaike Information Criterion (AIC).

- For all possible parameter constellations from both lists pdq and seasonal_pdq the algorithm will create a model and eventually pick the best one to proceed.
- The best model is chosen based on the Akaike Information Criterion (AIC).

Akaike Information Criterion (AIC).

- The Akaike information criterion (AIC) is a measure of the relative quality of statistical models for a given set of data.
- Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. Hence, AIC provides a means for model selection.
- It measures the trade-off between the goodness of fit of the model and the complexity of the model (number of included and estimated parameters).

One step ahead prediction

- The `get_prediction` and `conf_int` methods calculate predictions for future points in time for the previously fitted model and the confidence intervals associated with a prediction, respectively.
- The `dynamic=False` argument causes the method to produce a one-step ahead prediction of the time series

MSE

- To quantify the accuracy between model fit and true observations we use the mean squared error (MSE).
- The MSE computes the squared difference between the true and predicted value.

Out of sample Prediction

- To put the model to the real test with a 24-month-head prediction.
- This requires to pass the argument `dynamic=False` when using the `get_prediction` method."

Long term forecasting

- Finally, a 10 year ahead forecast, leveraging a seasonal ARIMA model trained on the complete time series y .
- Grid search found the best model to be of form SARIMAX(2, 1, 3)(1, 2, 1)12 for the data vector y .

Tractor Sales & Marketing Analysis

PH Trend - Time Series Decomposition

- Remove wrinkles from our time series using moving average.
- Moving average of different time periods i.e. 4,6,8, and 12 months

PH Tractor - Dicky Fuller Test on the timeseries

- Run the Dicky Fuller Test on the timeseries and
- Verify the null hypothesis that the TS is non-stationary.

PH Tractor Seasonality Time Series Decomposition 1/2

- The first thing to do is to see how number of tractors sold vary on a month on month basis. .
- A stacked annual plot to observe seasonality
- A box plot

PH Tractor Seasonality Time Series Decomposition 2/2

- The tractor sales have been increasing without fail every year.
- July and August are the peak months for tractor sales and the variance and the mean value in July and August are also much higher than any of the other months.
- We can see a seasonal cycle of 12 months where the mean value of each month starts with a increasing trend in the beginning of the year and drops down towards the end of the year.
- We can see a seasonal effect with a cycle of 12 months.

PH Tractor Irregular Remainder 1/4

- To decipher underlying patterns in tractor sales, we build a multiplicative time series decomposition model
- The primary purpose is to understand underlying patterns in temporal data to use in more sophisticated analysis like Holt-Winters seasonal method or ARIMA.

PH Tractor Irregular Remainder 2/4

- Key observations:
- 1) Trend: 12-months moving average looks similar to a straight line hence we could use linear regression to estimate the trend in this data.
- 2) Seasonality: seasonal plot displays a fairly consistent month-on-month pattern. The monthly seasonal components are average values for a month after removal of trend. Trend is removed from the time series

PH Tractor Irregular Remainder 3/4

- Key observations:
- 3) Irregular Remainder (random): is the residual left in the series after removal of trend and seasonal components.
- The expectations from remainder component is that it should look like a white noise i.e. displays no pattern at all.

PH Tractor Irregular Remainder 4/4

- Key observations:
- However, for our series residual displays some pattern with high variation on the edges of data i.e. near the beginning (2004-07) and the end (2013-14) of the series.

PH ARIMA Modeling 1/4

- ARIMA is a combination of 3 parts i.e. AR (AutoRegressive), I (Integrated), and MA (Moving Average). A convenient notation for ARIMA model is ARIMA(p,d,q).
- Here p,d, and q are the levels for each of the AR, I, and MA parts.
- Each of these three parts is an effort to make the final residuals display a white noise pattern (or no pattern at all).

PH ARIMA Modeling 2/4

- In each step of ARIMA modeling, time series data is passed through these 3 parts like a sugar cane through a sugar cane juicer to produce juice-less residual. The sequence of three passes for ARIMA analysis is as following:
- 1st Pass of ARIMA to Extract Juice / Information
- Integrated (I) – subtract time series with its lagged series to extract trends from the data

PH ARIMA Modeling 3/4

- In this pass of ARIMA juicer, we extract trend(s) from the original time series data.
- Differencing is one of the most commonly used mechanisms for extraction of trends. Here, the original series is subtracted with it's lagged series e.g. November's sales values are subtracted with October's values to produce trend-less residual series.

PH ARIMA Modeling 4/4

- The formulae for different orders of differencing in the plot a time series data with a linearly upward trend is displayed.
- Adjacent to that plot is the 1st order differenced plot for the same data.
- We can notice after 1st order differencing, trend part of the series is extracted and the difference data (residual) does not display any trend.

2nd Pass of ARIMA 1/2

- AutoRegressive (AR) – extract the influence of the previous periods' values on the current period
- After the time series data is made stationary through the integrated (I) pass, the AR part of the ARIMA juicer gets activated.
- As the name auto-regression suggests, here we try to extract the influence of the values of previous periods on the current period e.g. the influence of the September and October's sales value on the November's sales.

2nd Pass of ARIMA 2/2

- This is done through developing a regression model with the time lagged period values as independent or predictor variables.
- AR model of order 1 i.e. $p=1$ or $\text{ARIMA}(1,0,0)$ is represented

3rd Pass of ARIMA 1/2

- Moving Average (MA) – extract the influence of the previous period's error terms on the current period's error
- Finally, the last component of ARIMA i.e. MA involves finding relationships between the previous periods' error terms on the current period's error term.

3rd Pass of ARIMA 2/2

- Moving Average (MA) part of ARIMA is developed with the simple multiple linear regression values with the lagged error values as independent or predictor variables.
- MA model of order 1 i.e. $q=1$ or ARIMA(0,0,1) is represented

White Noise & ARIMA 1/3

- White noise is a funny thing, if we look at it for long we will start seeing some false patterns. This is because the human brain is wired to find patterns, and at times confuses noises with signals.
- The biggest proof of this is how people lose money every day on the stock market.

White Noise & ARIMA 2/3

- This is the reason why we need a mathematical or logical process to distinguish between a white noise and a signal (juice / information).
- Consider the simulated white noise

White Noise & ARIMA 3/3

- Key observations:
 - If we stare at the graph for a reasonably long time we may start seeing some false patterns
 - A good way to distinguish between signal and noise is ACF (AutoCorrelation Function). This is developed by finding the correlation between a series of its lagged values..

ACF Plot 1/2

- Key observations:
 - We could see that for lag = 0 the ACF plot has the perfect correlation i.e. $\rho = 1$. because any data with itself will always have the perfect correlation
 - However as expected, our white noise doesn't have a significant correlation with its historic values ($lag \geq 1$).
 -

ACF Plot 2/2

- Key observations:
 - The dotted horizontal lines in the plot show the threshold for the insignificant region i.e. for a significant correlation the vertical bars should fall outside the horizontal dotted lines.

Step 2: Difference data to make data stationary on mean (remove trend) 1/2

- Key Observations:
 - The above series is not stationary on variance i.e. variation in the plot is increasing as we move towards the right of the chart.
 - We need to make the series stationary on variance to produce reliable forecasts through ARIMA models.

Step 2: Difference data to make data stationary on mean (remove trend) 2/2

- Key Observations:
 - The tractor sales has an upward trend for tractors sales and there is also a seasonal component.
 - Make the series stationary by removing the upward trend through 1st order differencing of the series

Step 3: log transform data to make data stationary on variance

- One of the best ways to make a series stationary on variance is through transforming the original series through log transform.
- Log transform original tractor sales series it to make it stationary on variance.
- This series is not stationary on mean since we are using the original data without differencing. But now the series looks stationary on variance.

Step 4: Difference log transform data to make data stationary on both mean and variance 1/2

- Look at the differenced plot for log transformed series to reconfirm if the series is actually stationary on both mean and variance.

Step 4: Difference log transform data to make data stationary on both mean and variance 2/2

- Key Observations:
 - This series looks stationary on both mean and variance.
 - This also gives us the clue that I or integrated part of our ARIMA model will be equal to 1 as 1st difference is making the series stationary.

Step 5: Plot ACF and PACF to identify potential AR and MA model 1/2

- Key Observations:
- Since, there are enough spikes in the plots outside the insignificant zone (dotted horizontal lines) we can conclude that the residuals are not random.
- This implies that there is patterns or information available in residuals to be extracted by AR and MA models.

Step 5: Plot ACF and PACF to identify potential AR and MA model 2/2

- Key Observations:
 - Also, there is a seasonal component available in the residuals at the lag 12 (represented by spikes at lag 12).
 - Since we are analyzing monthly data that tends to have seasonality of 12 months because of patterns in tractor sales.

Step 6: Identification of best fit ARIMA model 1/2

- In order to fit the time series data with a seasonal ARIMA model, we need to first find the values of ARIMA(p,d,q)(P,D,Q)s that optimize a metric of interest such as AIC or BIC.
- We generate combination of p,d and q to select the optimal parameter values for our ARIMA(p,d,q)(P,D,Q)s time series model.

Step 6: Identification of best fit ARIMA model 2/2

- This technique is known as "grid search" where we iteratively explore different combinations of parameters.
- For each such combination of parameters, we try to fit a new seasonal ARIMA model with the SARIMAX() function from the statsmodels module and assess AIC or BIC score.
- The model with the best score wins and the parameters for that model are the optimal parameters.

AIC & BIC 1/4

- The best fit model is selected based on Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC) values.
- The idea is to choose a model with minimum AIC and BIC values.
- Akaike Information Criterion (AIC) - AIC is an effort to balance the model between goodness-of-fit and number of parameters used in the model.

AIC & BIC 2/4

- Bayesian Information Criterion (BIC) is another variant of AIC and is used for the same purpose of best fit model selection.
- For the best possible model selection, we look at AIC, BIC, and AICc (AIC with sample correction) if all these values are minimum for a given model.

AIC & BIC 3/4

- As expected, our model has I (or integrated) component equal to 1.
- This represents differencing of order 1. There is additional differencing of lag 12 in the above best fit model.

AIC & BIC 4/4

- Moreover, the best fit model has MA value of order 1.
Also, there is seasonal MA with lag 12 of order 1.

Step 7: Predict sales on in-sample date using the best fit ARIMA model

- The next step is to predict tractor sales for in-sample data and find out how close is the model prediction on the in-sample data to the actual truth.

Step 8: Forecast sales using the best fit ARIMA model 1/2

- The next step is to predict tractor sales for next 3 years i.e. for 2015, 2016, and 2017 through the above model.
- Get forecast 36 steps (3 years) ahead in future

Step 8: Forecast sales using the best fit ARIMA model 2/2

- Key Observations:
 - A short-term forecasting model, say a couple of business quarters or a year, is usually a good idea to forecast with reasonable accuracy.
 - A long-term model like the one above needs to evaluated on a regular interval of time (say 6 months).
 - The idea is to incorporate the new information available with the passage of time in the model.

Step 9: Plot ACF and PACF for residuals of ARIMA 1/6

- Plot ACF and PACF for residuals of ARIMA model to ensure no more information is left for extraction
- Finally, create an ACF and PACF plot of the residuals of our best fit
- ARIMA model i.e. ARIMA(0,1,1)(1,0,1)[12].

Step 9: Plot ACF and PACF for residuals of ARIMA 2/6

- Key Observations:
 - We need to ensure that the residuals of our model are uncorrelated and normally distributed with zero-mean.
 - If it is not that it signifies that the model can be further improved and we repeat the process with the residuals.

Step 9: Plot ACF and PACF for residuals of ARIMA 3/6

- Key Observations:
 - In this case, our model diagnostics suggests that the model residuals are normally distributed based on the following:
 - The KDE plot of the residuals on the top right is almost similar with the normal distribution.

Step 9: Plot ACF and PACF for residuals of ARIMA 4/6

- Key Observations:
 - The qq-plot on the bottom left shows that the ordered distribution of residuals (blue dots) follows the linear trend of the samples taken from a standard normal distribution with $N(0, 1)$.
 - This is a strong indication that the residuals are normally distributed.

Step 9: Plot ACF and PACF for residuals of ARIMA 5/6

- Key Observations:
- The residuals over time (top left plot) don't display any obvious seasonality and appear to be white noise.
 - This is confirmed by the autocorrelation (i.e. correlogram) plot on the bottom right, which shows that the time series residuals have low correlation with lagged versions of itself.

Step 9: Plot ACF and PACF for residuals of ARIMA 6/6

- Key Observations:
- Those observations coupled with the fact that there are no spikes outside the insignificant zone for both ACF and PACF plots lead us to conclude that residuals are random with no information or patterns in them and our model produces a satisfactory fit that could help us understand our time series data and forecast future values.
- It means that our ARIMA model is working fine.

Sales & Marketing: Regression with ARIMA Errors

1/5

- For the last 4 years, PH tractors is running an expensive marketing and farmer connect program to boost their sales.
- They are interested in learning the impact of this program on overall sales.
- As a data science consultant we are helping them with this effort.
- This is a problem that requires a thorough analysis followed by creative solutions and scientific monitoring mechanism.

Sales & Marketing: Regression with ARIMA Errors

2/5

- We will build models based on regression with ARIMA errors and compare them with the pure play ARIMA model.
- This analysis will provide some clues towards effectiveness of the marketing program.
- However, this analysis will not be conclusive for finding shortcomings and enhancements for the program which will require further analysis and creative solutions.

Sales & Marketing: Regression with ARIMA Errors

3/5

- Key Observations:
- This looks promising with quite a high correlation coefficient ($\rho > 0.8$).
- However, there is a danger in analyzing non-stationary time series data.
- Since two uncorrelated series can display high correlation because of time series trend in data.

Sales & Marketing: Regression with ARIMA Errors

4/5

- Key Observations:
 - In this case, PH is a growing company and the latent factor is 'growth' of the company.
 - Hence both its sales and marketing expenses can be on an upward curve independent of each other.

Sales & Marketing: Regression with ARIMA Errors

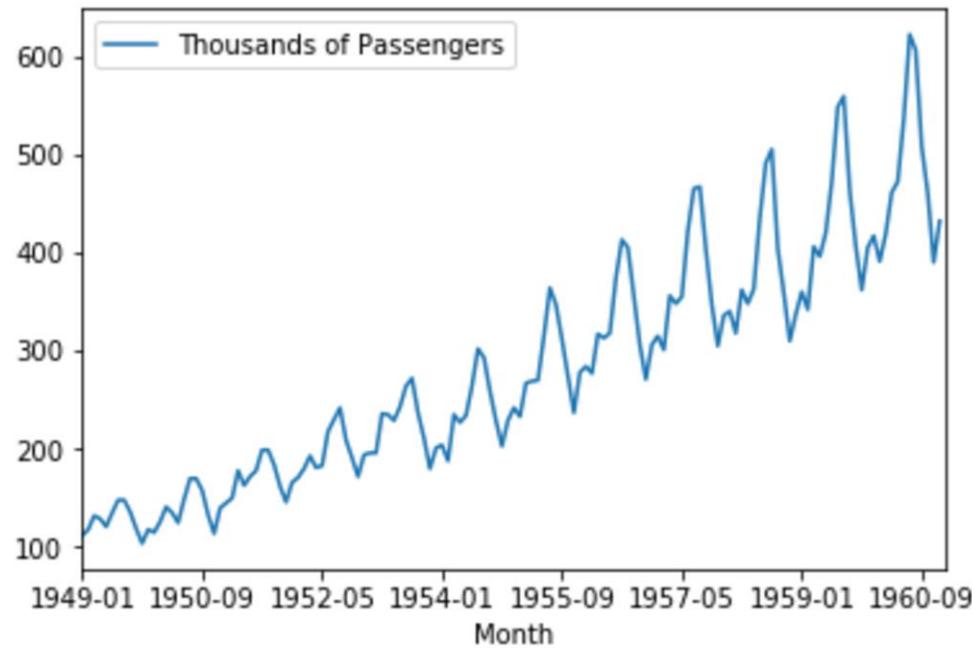
5/5

- Key Observations:
 - To investigate that a better way is to find the correlation between stationary data obtained through differencing of marketing expenditure and the tractor sales data individually.
 - The correlation plot for stationary data:

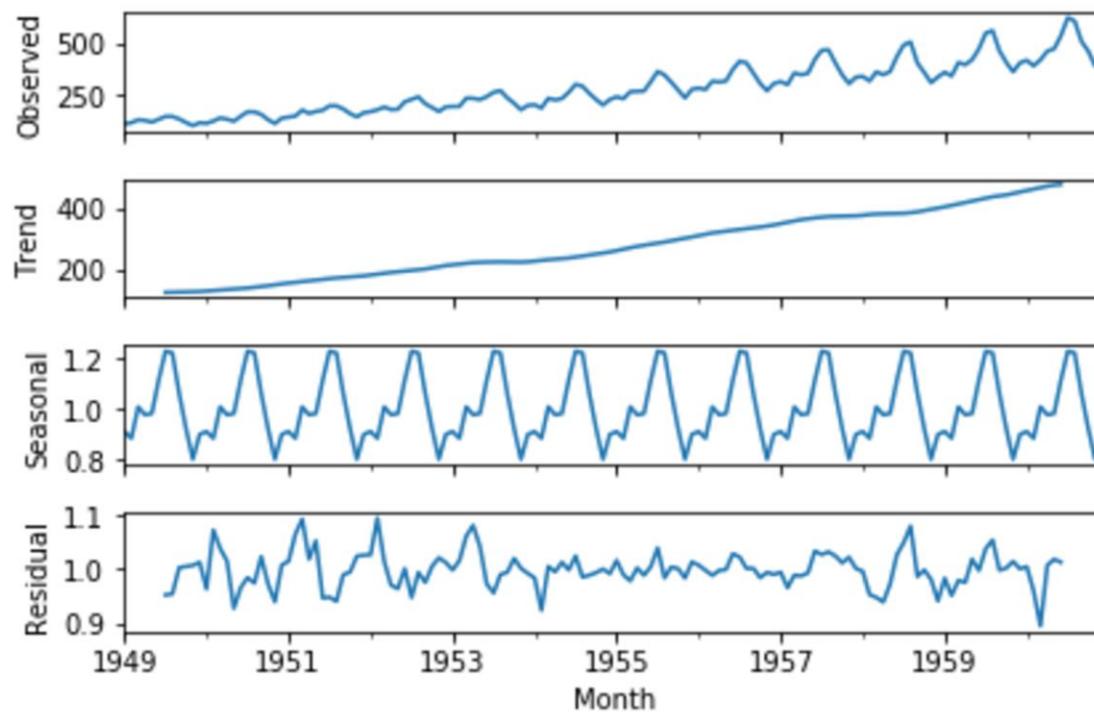
ETS Models

- ETS Models (Error-Trend-Seasonality)
 - Exponential Smoothing
 - Trend Methods Models
 - ETS Decomposition

- ETS Decomposition - Airline Passengers



- ETS Decomposition - Airline Passengers



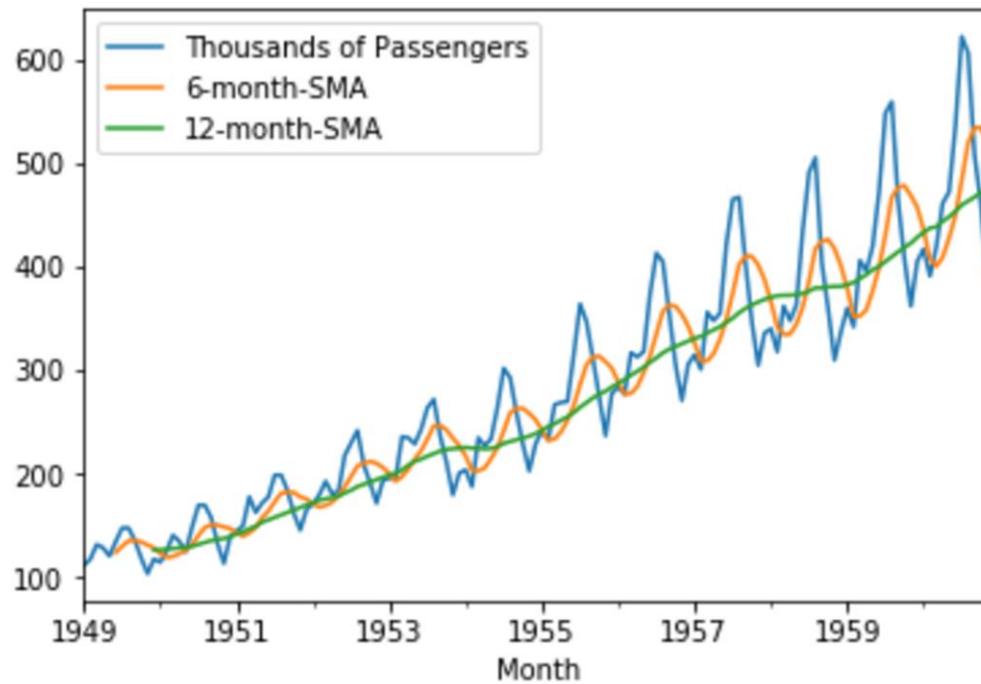
- Time Series Decomposition with ETS (Error-Trend-Seasonality).
- Visualizing the data based off its ETS is a good way to build an understanding of its behaviour.

- ETS (Error-Trend-Seasonality) Models will take each of those terms for “smoothing” and may add them, multiply them, or even just leave some of them out.
- Based off these key factors, we can try to create a model to fit our data.

- Time Series Decomposition with ETS (Error-Trend-Seasonality).
- Visualizing the data based off its ETS is a good way to build an understanding of its behaviour.

EWMA Models

- SMA - Simple Moving Averages



- EWMA- Exponentially Weighted Moving Averages
- Basic SMA has some "weaknesses".
 - Does not really inform you about possible future behaviour, all it really does is describe trends in your data.

- EWMA- Exponentially Weighted Moving Averages
- Basic SMA has some "weaknesses".
 - To help fix some of these issues, we can use an EWMA (Exponentially-weighted moving average).

- EWMA will allow us to reduce the lag effect from SMA and it will put more weight on values that occurred more recently (by applying more weight to the more recent values, thus the name).

ARIMA Models

- AutoRegressive Integrated Moving Average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model.

- Both of those models (ARIMA and ARMA) are fitted to time series data either to better understand the data or to predict future points in the series (forecasting).

- ARIMA (Autoregressive Integrated Moving Averages)
 - Non-seasonal ARIMA
 - Seasonal ARIMA

- ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity.

- Non-seasonal ARIMA models are generally denoted ARIMA(p,d,q) where parameters p, d, and q are non-negative integers.

- Parts of ARIMA model
- AR (p): Autoregression
 - A regression model that utilizes the dependent relationship between a current observation and observations over a previous period

- Parts of ARIMA model
- I (d): Integrated.
 - Differencing of observations (subtracting an observation from an observation at the previous time step) in order to make the time series stationary.

- Parts of ARIMA model
- MA (q): Moving Average.
 - A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

AutoCorrelation Plots

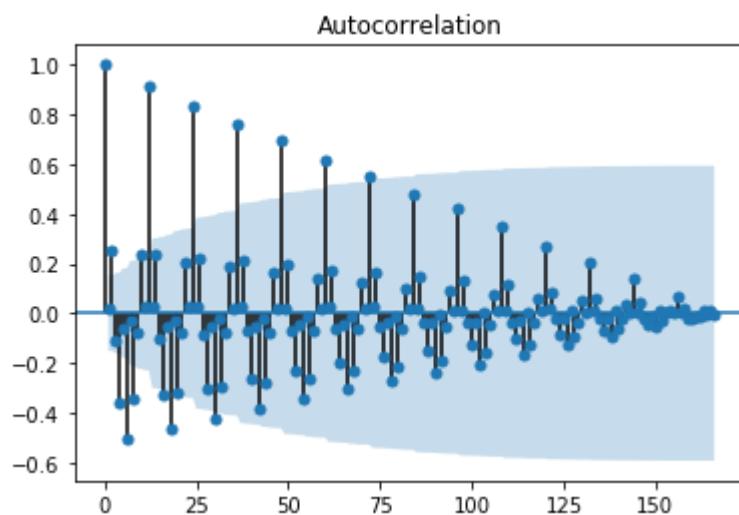
- An autocorrelation plot (also known as a Correlogram) shows the correlation of the series with itself, lagged by x time units.
- So the y axis is the correlation and the x axis is the number of time units of lag.

- Imagine taking your time series of length T , copying it, and deleting the first observation of copy 1 and the last observation of copy 2.

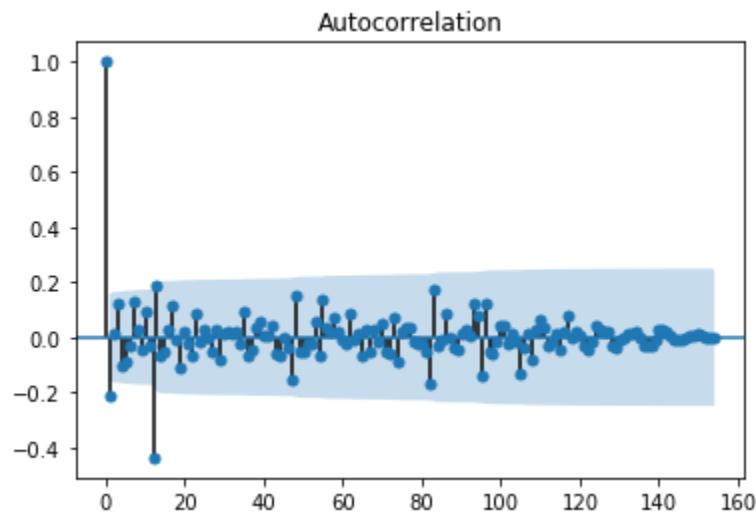
- Now you have two series of length $T-1$ for which you calculate a correlation coefficient.
- This is the value of the vertical axis at $x=1$ in your plots.

- It represents the correlation of the series lagged by one time unit.
- You go on and do this for all possible time lags x and this defines the plot.

- Gradual Decline



- Sharp Drop-off



- In general you would use either AR or MA, using both is less common.
- When actually applying the AR and MA terms, you will set values of p or q.

- If the autocorrelation plot shows positive autocorrelation at the first lag (lag-1), then it suggests to use the AR terms in relation to the lag

- If the autocorrelation plot shows negative autocorrelation at the first lag, then it suggests using MA terms.
- This will allow you to decide what actual values of p,d, and q to provide your ARIMA model.

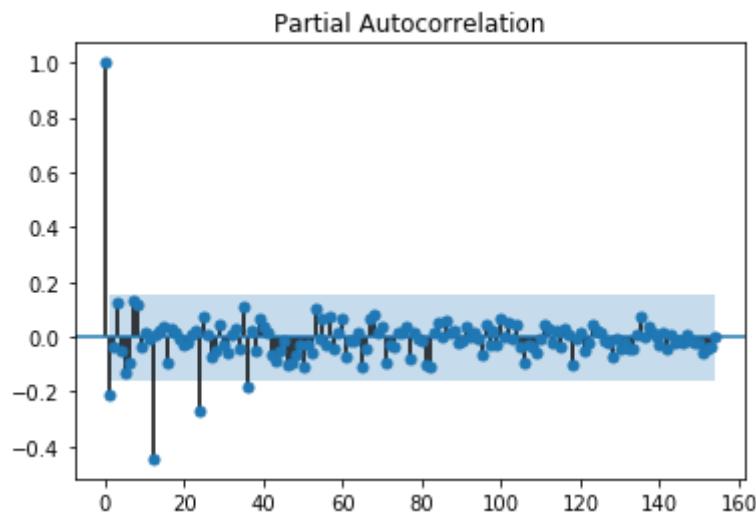
- p: The number of lag observations included in the model.
- d: The number of times that the raw observations are differenced
- q: The size of the moving average window, also called the order of moving average.

- There are also partial autocorrelation plots!

- In general, a partial correlation is a conditional correlation.
- It is the correlation between two variables under the assumption that we know and take into account the values of some other set of variables.

- For instance, consider a regression context in which y = response variable and x_1 , x_2 , and x_3 are predictor variables.
- The partial correlation between y and x_3 is the correlation between the variables determined taking into account how both y and x_3 are related to x_1 and x_2 .

- an example of what the plot can look like:



- Typically a sharp drop after lag "k" suggests an AR-k model should be used.
- If there is a gradual decline, it suggests an MA model.

- Identification of an AR model is often best done with the PACF.
- Identification of an MA model is often best done with the ACF rather than the PACF.

- Finally once you've analyzed your data using ACF and PACF you are ready to begin to apply ARIMA or Seasonal ARIMA, depending on your original data.
- We will provide the p,d, and q terms for the model.

- An ARIMA will then take three terms p,d, and q.
(We'll see this in the coding example)
- For seasonal ARIMA there will be an additional set of P,D,Q terms that we will see.

dplyr

a grammar of
data manipulation

Group	dose 1	dose 2
A	3	3
A	4	5
B	3	1
B	1	3
C	1	3
C	2	2

A 3 3

A 4 5

B 3 1

B 1 3

C 1 3

C 2 2

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

n	min	mean	max
6	4	5.2	9

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9

Group	Total
A	15

B	3	1	4
B	1	3	4

B	8
---	---

C	1	3	4
C	2	2	4

C	8
---	---

n	min	mean	max
6	4	5.2	9

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

Group	Sum
A	6
A	9
B	4
B	4
C	4
C	4

Group	Sum
A	6
C	4
C	4

tbl
%>%



database



American Airlines



American Eagle



FRONTIER
AIRLINES

jetBlue
AIRWAYS®

SkyWest
AIRLINES*

 **DELTA**



tbl

select
filter
arrange
mutate
summarize

Group	first dose	second dose	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

select

Group	Sum
A	6
A	9
B	4
B	4
C	4
C	4

select

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

filter

Group	dose 1	dose 2	Sum
A	3	3	6
C	1	3	4
C	2	2	4

filter

Group	dose 1	dose 2	Sum
C	1	3	4
A	3	3	6
A	4	5	9
B	3	1	4
B	2	3	4
C	5	2	4

arrange

Group	dose 1	dose 2	Sum
C	1	3	4
B	2	3	4
B	3	1	4
A	3	3	6
A	4	5	9
C	5	2	4

arrange

Group	dose 1	dose 2
A	3	3
A	4	5
B	3	1
B	1	3
C	1	3
C	2	2

mutate

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

mutate

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

summarise

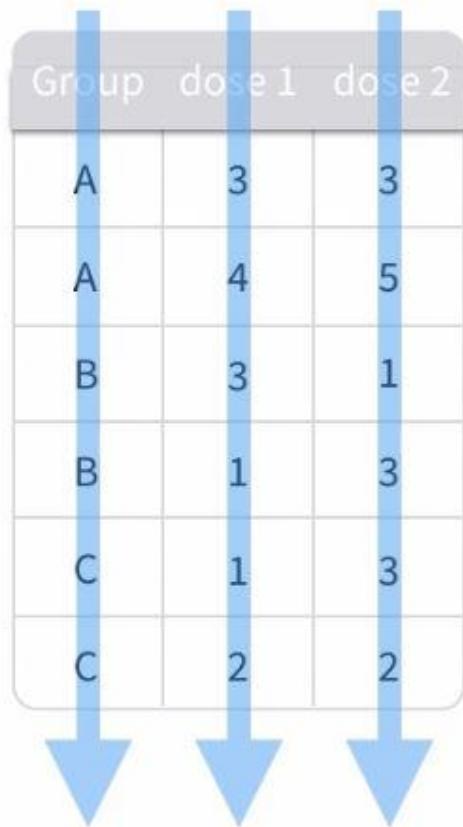
Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

n	min	mean	max
6	4	5.2	9

summarise

variables

Group	dose 1	dose 2
A	3	3
A	4	5
B	3	1
B	1	3
C	1	3
C	2	2



variables

observations

	Group	dose 1	dose 2	
	A	3	3	→
	A	4	5	→
	B	3	1	→
	B	1	3	→
	C	1	3	→
	C	2	2	→

select

mutate

filter

arrange

summarize

select

mutate

variables

filter

arrange

summarize

select

mutate

variables

filter

arrange

observations

summarize

select

mutate

variables

filter

arrange

observations

summarize

groups

Year	ActualElapsedTime
Month	AirTime
DayofMonth	ArrDelay
DayofWeek	DepDelay
DepTime	Origin
ArrTime	Dest
UniqueCarrier	Distance
FlightNum	TaxiIn
TailNum	TaxiOut
	Cancelled
	CancellationCode
	Diverted

Year	ActualElapsedTime
Month	AirTime
DayofMonth	ArrDelay
DayofWeek	DepDelay
DepTime	Origin
ArrTime	Dest
UniqueCarrier	Distance
FlightNum	TaxiIn
TailNum	TaxiOut
	Cancelled
	CancellationCode
	Diverted

tbl

columns to
select

```
select(df, Group, Sum)
```

tbl

columns to
select

select(df, Group, Sum)

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

tbl

columns to
select

select(df, Group, Sum)

Group	Sum
A	6
A	9
B	4
B	4
C	4
C	4

Group	dose 1	dose 2
A	3	3
A	4	5
B	3	1
B	1	3
C	1	3
C	2	2

mutate

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

mutate

length x width x height =

length	width	height
2	3	3
2	4	5
3	3	1
1	1	3

length x width x height = volume

length	width	height	volume
2	3	3	18
2	4	5	40
3	3	1	9
1	1	3	3

length x width x height = volume

mass / volume =

mass	volume
50	10
45	15
35	10

length x width x height = volume

mass / volume = density

mass	volume	density
50	10	5
45	15	3
35	10	3.5

```
mutate(h1, loss = ArrDelay - DepDelay)
```

tbl

```
mutate(h1, loss = ArrDelay - DepDelay)
```

tbl

new column
name

```
mutate(h1, loss = ArrDelay - DepDelay)
```

tbl

new column
name

=

expression

```
mutate(h1, loss = ArrDelay - DepDelay)
```

tbl

new column
name

=

expression

```
mutate(h1, loss = ArrDelay - DepDelay)
```

Year	ArrDelay	DepDelay	loss
2011	-10	0	-10
2011	-9	1	-10
2011	-8	-8	0
2011	3	3	0
2011	-3	5	-8
2011	-7	-1	-6

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

filter

Group	dose 1	dose 2	Sum
A	3	3	6
C	1	3	4
C	2	2	4

filter

tbl

logical test

```
filter(hflights, Cancelled == 1)
```

tbl

logical test

```
filter(hflights, Cancelled == 1)
```

Year	Cancelled	Dest
2011	0	DFW
2011	1	DFW
2011	0	ELP
2011	1	ELP

tbl

logical test

```
filter(hflights, Cancelled == 1)
```

Year	Cancelled	Dest
2011	1	DFW
2011	1	ELP

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	2	3	4
C	1	3	4
C	5	2	4

arrange

Group	dose 1	dose 2	Sum
C	1	3	4
B	2	3	4
B	3	1	4
A	3	3	6
A	4	5	9
C	5	2	4

arrange

tbl

column
name

```
arrange(hflights, DepDelay)
```

Year	DepDelay	Dest
2011	-2	DFW
2011	3	DFW
2011	0	ELP
2011	10	ELP

tbl

column
name

```
arrange(hflights, DepDelay)
```

Year	DepDelay	Dest
2011	-2	DFW
2011	0	ELP
2011	3	DFW
2011	10	ELP

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

summarise

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

n	min	mean	max
6	4	5.2	9

summarise

tbl

new column
name

=

expression

```
summarise(df, sum = sum(A),  
          avg = mean(B),  
          var = var(B))
```

tbl

new column
name

=

expression

```
summarise(df, sum = sum(A),  
          avg = mean(B),  
          var = var(B))
```

A	B	C
105	6	20
108	3	18
144	3	7
132	5	8

tbl

new column
name

=

expression

```
summarise(df, sum = sum(A),  
          avg = mean(B),  
          var = var(B))
```

A	B	C
105	6	20
108	3	18
144	3	7
132	5	8

sum	avg	var
489	4.25	44.92

A	B	C
105	6	20
108	3	18
144	3	7
132	5	8



summarise

sum	avg	var
489	4.25	44.92

```
a1 <- select(a, X, Y, Z)
a2 <- filter(a1, X > Y)
a3 <- mutate(a2, Q = X + Y + Z)
a4 <- summarise(a3, all = sum(Q))
```

```
a1 <- select(a, X, Y, Z)
a2 <- filter(a1, X > Y)
a3 <- mutate(a2, Q = X + Y + Z)
a4 <- summarise(a3, all = sum(Q))
```

```
summarise(  
  mutate(  
    filter(  
      select(a, X, Y, Z),  
      X > Y),  
      Q = X + Y + Z),  
    all = sum(Q))  
)
```

%>%

magrittr

some
object

pipe

some
function

arguments

```
object %>% function(____, arg2, arg3, ...)
```

```
summarise(  
  mutate(  
    filter(  
      select(a, X, Y, Z),  
      X > Y),  
      Q = X + Y + Z),  
  all = sum(Q))  
)
```

```
a %>%  
  select(X, Y, Z) %>%  
  filter(X > Y) %>%  
  mutate(Q = X + Y + Z) %>%  
  summarise(all = sum(Q))
```

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

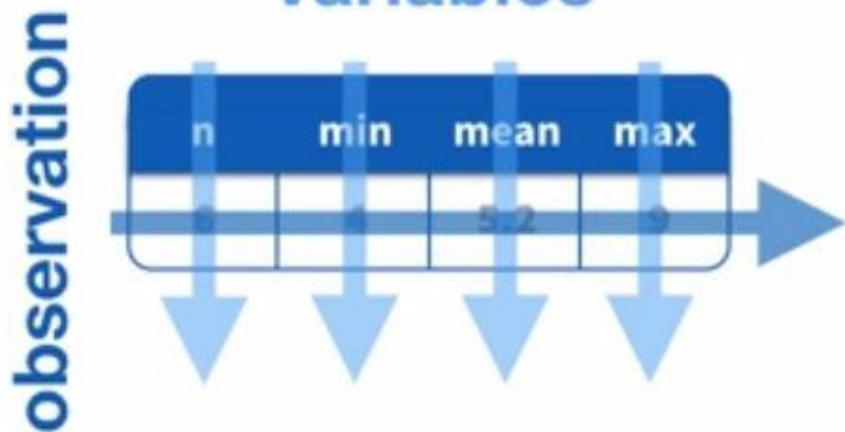


n	min	mean	max
6	4	5.2	9

summarise

Group	dose 1	dose 2	Sum
A	3	3	6
A	4	5	9
B	3	1	4
B	1	3	4
C	1	3	4
C	2	2	4

variables



Group	dose 1	dose 2	Sum	
A	3	3	6	
A	4	5	9	



Group	Total
A	15

B	3	1	4	
B	1	3	4	



B	8
---	---

C	1	3	4	
C	2	2	4	



C	8
---	---

n	min	mean	max
6	4	5.2	9

group_by

tbl

column to
group by

group_by(df, Group)

tbl

column to
group by

```
group_by(df, Group)
```

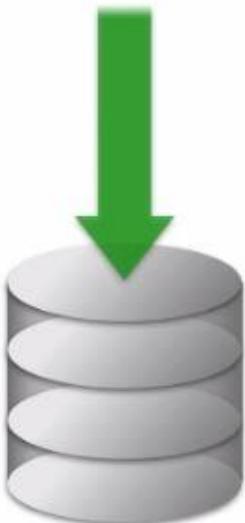
```
df %>%
```

```
  group_by(Group)
```

data frame

data table

database



database

WHAT IS TEXT MINING ?

- An exploration and analysis of textual (natural language) data to identify facts, relationships and assertions
- Process of analysing collections of textual materials in order to capture key concepts and themes and uncover hidden relationships and trends
- Examples:
 - ✓ Classical Spam Filtering
 - ✓ Review Analysis
 - ✓ Tweets Analysis

WHAT IS MISUNDERSTOOD AS TEXT MINING ?

- Information Retrieval
 - ✓ Information Retrieval is a domain that deals with most effective ways of retrieving information according to user needs and it is more concerned with search engine
- Examples: Search Engines like Google

WHAT IS MISUNDERSTOOD AS TEXT MINING ?

- Information Extraction
 - ✓ Information Extraction (IE) is about locating specific items in textual data
 - ✓ In information extraction we just extract the already known information which was not properly formatted
- Example: Home Address

WHAT IS MISUNDERSTOOD AS TEXT MINING ?

- Natural Language Processing (NLP)
 - ✓ NLP is based on latent features of the text and uses those in its methods and text mining is based on observed features
 - ✓ NLP considers the context in the text, while text mining does not

TEXT MINING SOURCES

- Comments (Social Networking Sites)
- Tweets
- Sales Reports
- Emails
- Blogs
- Word Documents

TYPES OF DOCUMENTS IN TEXT MINING

- Structured Documents
 - ✓ Survey Forms
 - ✓ Claims
- Semi Structured Documents
 - ✓ Job Listings
 - ✓ Invoices
- Unstructured (Free Format) Documents
 - ✓ Blogs

TEXT MINING PROCESS (STEPS)

- Given Data (Text)
- Text Preprocessing
- Feature Generation/Extraction
- Feature Selection
- Text Mining Methods
- Results Evaluation

TEXT PREPROCESSING

- The main idea is to extract (identify) the words from the texts by removing unnecessary data
- Words can be extracted by using following two techniques:
 - ✓ Lexical Analysis also known as Tokenization
 - ✓ Syntactical Analysis also known as part of speech tagging (using Brill POS Tagger)

TEXT PREPROCESSING

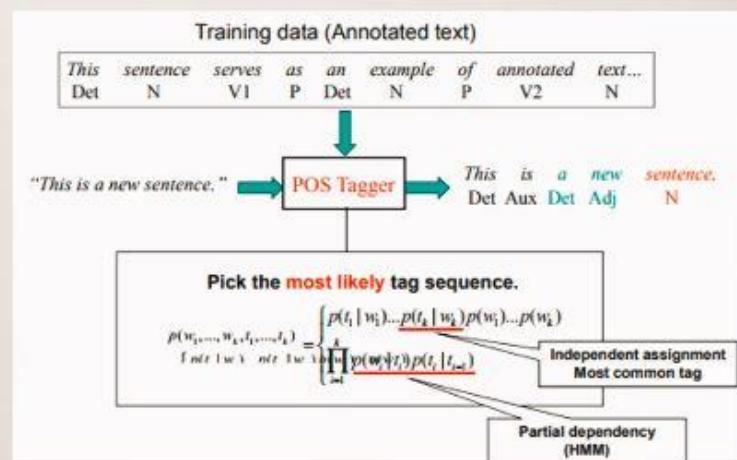
- Lexical Analysis:
 - ✓ Can be done using delimiters or by using a dictionary
- Easy to do, but sometimes not efficient
 - ✓ For example: swimming pool
- Lexical Analysis using Dictionary

TEXT PREPROCESSING

- Syntactic Analysis:

- ✓ Can be done using dictionary or by using a POS TAGGER

- Using a POS Tagger:



Taken from: ChengXiang Zhai

FEATURE EXTRACTION

- The main goal in this step is to extract good subset of words to represent the text documents in the collection
- Can be achieved by following methods:
 - ✓ Stop Word Elimination (remove uninformative words)
 - ✓ Stemming and Lemmatization

FEATURE EXTRACTION

- Stop Word Elimination:
- Can be done by using following two methods
 - ✓ Using a already maintained list of stop words
 - ✓ Bu using some statistical approach

FEATURE EXTRACTION

- Stemming:
 - ✓ To cut the end of words by some heuristic based process
- Examples:
 - ✓ Shoe and Shoes
 - ✓ Story and Stories

FEATURE EXTRACTION

- Lemmatization:
 - ✓ Another efficient way to reduce the vocabulary size and this is done by using a dictionary
- Examples:
 - ✓ Walking and Walk
 - ✓ Good and Better

WEIGHTING MODELS

- Main goal is to convert bag of word to vector space model

$$d_i = (w_{i1}, w_{i2}, \dots, w_{it}) \in \mathbf{R}^t$$

- w_{ij} is the weight of j^{th} term in document d_i , where j belongs to the index composed of t terms.
- Approaches:
 - ✓ Boolean Model
 - ✓ Term Frequency (TF)
 - ✓ Term Frequency Inverse Document Frequency (TFIDF)

TERM FREQUENCY MODEL

- This model takes into account the number of occurrences of words in the document

$$d_i = (w_{i1}, w_{i2}, \dots, w_{it}) \in \mathbf{R}^t$$

w_{ij} : the number of times jth term occurs in document d_i

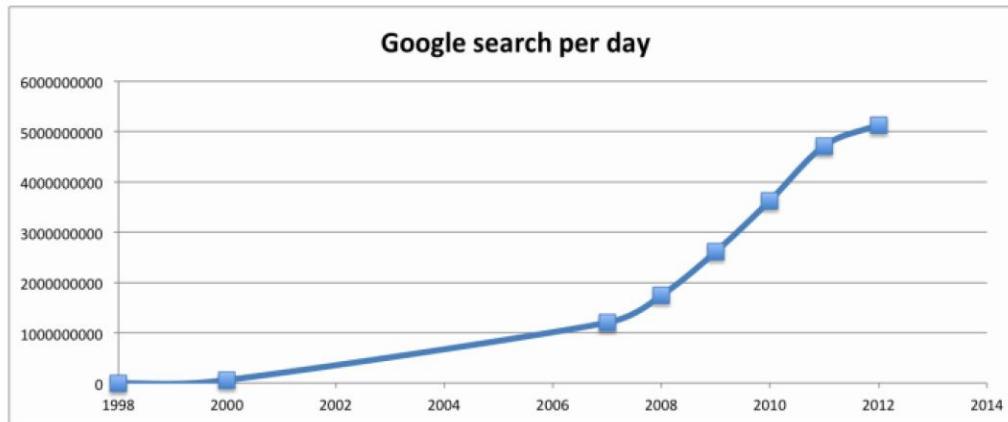
$$w_{ij} = tf_{ij}$$

- Example:
 - ✓ I need to practice, if I want to pass
 - ✓ D=[2, 1, 2, 1, 1, 1, 1]

Tf-Idf and Cosine similarity

- In the year **1998** Google handled **9800** average search queries every day.
- In **2012** this number shot up to **5.13 billion** average searches per day.

The graph given below shows this astronomical growth.



Diploma in Data Science & Big Data Analytics

Naresh IT Opposite Satyam Theatre, Ameerpet, Hyderabad 040-2374666, 23734842

Google Search- How it works?

Let's consider 3 documents to show how this works. Take some time to go through them.

- **Document 1:** The game of life is a game of everlasting learning
- **Document 2:** The unexamined life is not worth living
- **Document 3:** Never stop learning

Let us imagine that you are doing a search on these documents with the following query:

"life learning"

The query is a **free text query**.

It means a query in which the terms of the query are typed freeform into the search interface, without any connecting search operators.

Diploma in Data Science & Big Data Analytics

Naresh IT Opposite Satyam Theatre, Ameerpet, Hyderabad 040-2374666, 23734842

Step 1: Term Frequency (TF)

Term Frequency also known as TF measures the number of times a term (word) occurs in a document.

Given below are the terms and their frequency on each of the document.

TF for Document 1

Document1	the	game	of	life	is	a	everlasting	learning
Term Frequency	1	2	2	1	1	1	1	1

TF for Document 2

Document2	the	unexamined	life	is	not	worth	living
Term Frequency	1	1	1	1	1	1	1

TF for Document 3

Document3	never	stop	learning
Term Frequency	1	1	1

Step 1 (continued): Normalized Term Frequency (TF)

- In reality each document will be of different size.
- On a large document the frequency of the terms will be much higher than the smaller ones.
- Hence we need to normalize the document based on its size.
- A simple trick is to divide the term frequency by the total number of terms.

e.g:

In Document 1 the term game occurs two times.

The total number of terms in the document is 10.

Hence the normalized term frequency is $2 / 10 = 0.2$.

Diploma in Data Science & Big Data Analytics

Naresh IT Opposite Satyam Theatre, Ameerpet, Hyderabad 040-2374666, 23734842

Step 1 (continued): Normalized Term Frequency (

Given below are the normalized term frequency for all the documents.

Normalized TF for Document 1

Document1	the	game	of	life	is	a	everlasting	learning
Normalized TF	0.1	0.2	0.2	0.1	0.1	0.1	0.1	0.1

Normalized TF for Document 2

Document2	the	unexamined	life	is	not	worth	living
Normalized TF	0.142857	0.142857	0.142857	0.142857	0.142857	0.142857	0.142857

Normalized TF for Document 3

Document3	never	stop	learning
Normalized TF	0.333333	0.333333	0.333333

Step 2: Inverse Document Frequency (IDF)

The main purpose of doing a search is to find out relevant documents matching the query. In the first step all terms are considered equally important.

- Certain terms that **occur too frequently have little power** in determining the relevance.
- Solution: **Weigh down the effects of too frequently occurring terms.**
- The terms that **occur less in the document can be more relevant.**
- Solution: **Weigh up the effects of less frequently occurring terms.**

Logarithms helps us to solve this problem.

Given below is the IDF for terms occurring in all the documents. Since the terms: the, life, is, learning occurs in 2 out of 3 documents they have a lower score compared to the other terms that appear in only one document.

Step 2 (continued): Inverse Document Frequency (IDF)

Computing IDF for the term **game**:

$IDF(\text{game}) = 1 + \log_e(\text{Total Number Of Documents} / \text{Number Of Documents with term } \text{game} \text{ in it})$

There are 3 documents in all = Document1, Document2, Document3

The term game appears in Document1

$$\begin{aligned} IDF(\text{game}) &= 1 + \log_e(3 / 1) \\ &= 1 + 1.098726209 \\ &= 2.098726209 \end{aligned}$$

Step 2 (continued): Inverse Document Frequency (IDF)

Terms	IDF
the	1.405507153
game	2.098726209
of	2.098726209
life	1.405507153
is	1.405507153
a	2.098726209
everlasting	2.098726209
learning	1.405507153
unexamined	2.098726209
not	2.098726209
worth	2.098726209
living	2.098726209
never	2.098726209
stop	2.098726209

Diploma in Data Science & Big Data Analytics

Naresh IT Opposite Satyam Theatre, Ameerpet, Hyderabad 040-2374666, 23734842

Step 3: TF * IDF

Remember we are trying to find out relevant documents for the query: **life learning**

- For each term in the query multiply its normalized term frequency with its IDF on each document.



- Given below is TF * IDF calculations for **life** and **learning** in all the documents.

	Document1	Document2	Document3
life	0.140550715	0.200786736	0
learning	0.140550715	0	0.468502384

Step 4 (Continued): Vector Space Model – Cosine Simi

- From each document we derive a vector.
- The set of documents in a collection then is viewed as a set of vectors in a vector space. Each term will have its own axis.
- Using the formula given below we can find out the similarity between any two documents.

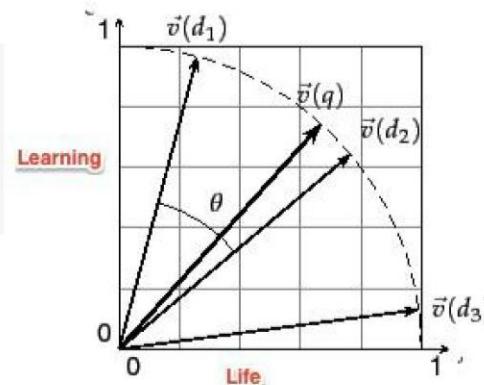
Cosine Similarity (d_1, d_2) = Dot product(d_1, d_2) / $\|d_1\| * \|d_2\|$

Dot product (d_1, d_2) = $d_1[0] * d_2[0] + d_1[1] * d_2[1] * \dots * d_1[n] * d_2[n]$

$\|d_1\| = \text{square root}(d_1[0]^2 + d_1[1]^2 + \dots + d_1[n]^2)$

$\|d_2\| = \text{square root}(d_2[0]^2 + d_2[1]^2 + \dots + d_2[n]^2)$

- Vectors deals only with numbers. In this example we are dealing with text documents.
- We used **TF and IDF** to convert text into numbers so that it can be represented by a vector.



Diploma in Data Science & Big Data Analytics

Naresh IT Opposite Satyam Theatre, Ameerpet, Hyderabad 040-2374666, 23734842

Step 4 (Continued): Vector Space Model – Cosine Similarity

The query entered by the user can also be represented as a vector.

We will calculate the TF*IDF for the query

	TF	IDF	TF*IDF
life	0.5	1.405507153	0.702753576
learning	0.5	1.405507153	0.702753576

Note:

The cosine value is always between -1 and 1 : the cosine of a small angle is near 1 , and the cosine of a large angle near 180 degrees is close to -1 . This is good, because small angles should map to high similarity, near 1 , and large angles should map to near -1 .

Step 4 (Continued): Vector Space Model – Cosine Similarity

Let us now calculate the cosine similarity of the query and Document1.

```
Cosine Similarity(Query, Document1) = Dot product(Query, Document1) / ||Query|| * ||Document1||

Dot product(Query, Document1)
= ((0.702753576) * (0.140550715) + (0.702753576)*(0.140550715))
= 0.197545035151

||Query|| = sqrt((0.702753576)2 + (0.702753576)2) = 0.993843638185

||Document1|| = sqrt((0.140550715)2 + (0.140550715)2) = 0.198768727354

Cosine Similarity(Query, Document1) = 0.197545035151 / (0.993843638185) * (0.198768727354)
= 0.197545035151 / 0.197545035151
= 1
```

Given below is the similarity scores for all the documents and the query

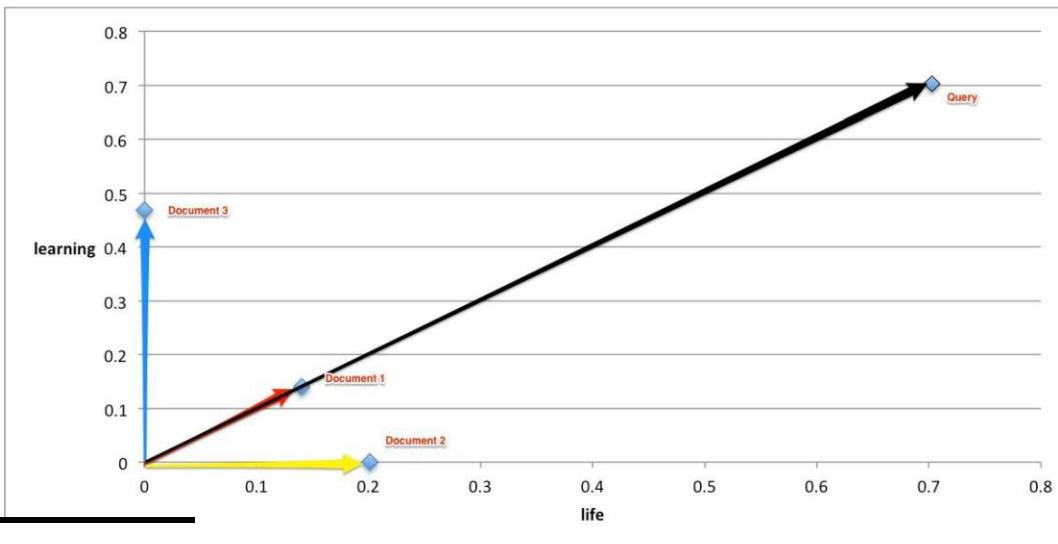
	Document1	Document2	Document3
Cosine Similarity	1	0.707106781	0.707106781

Diploma in Data Science & Big Data Analytics

Naresh IT Opposite Satyam Theatre, Ameerpet, Hyderabad 040-2374666, 23734842

Step 4 (Continued): Vector Space Model – Cosine Sim

- ✓ Below is the plot of vector values for the query and documents in 2-dimensional space of life and learning.
- ✓ Document1 has the highest score of 1.
- ✓ This is not surprising as it has both the terms **life** and **learning**.



DIMENSION REDUCTION

- The main goal in this step is to reduce the size of vocabulary to avoid overfitting (curse of dimensionality)
- Can be achieved by following methods:
 - ✓ Document Frequency Thresholding
 - ✓ χ^2 statistic
 - ✓ PCA (Principal Component Analysis)
 - ✓ Latent semantic Analysis (LSA)/LSI

TEXT MINING APPLICATIONS/TASKS

- Keyword Based Association Analysis:

“In keyword based association analysis, we collect a set of keywords that occur frequently together and then we try to find out the relationship”

- Examples:
 - ✓ Document Tagging

TEXT MINING APPLICATIONS/TASKS

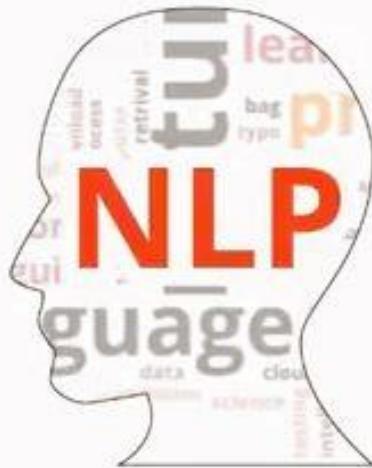
- Text mining as classification:

“We try to automatically classify large collection of documents in predefined categories”

- Classification can be accomplished by various techniques:
 - ✓ K neighbor KNN
 - ✓ Decision Trees
 - ✓ Neural Networks (Deep Learning)
 - ✓ SVM (Support Vector Machines)
 - ✓ Naïve Bayes Classifier (Probabilistic Model)

Natural Language Processing (NLP)

Natural Language Processing is an automated way to understand and analyze natural human languages and extract information from such data by applying machine algorithms.



Machine learning
algorithms

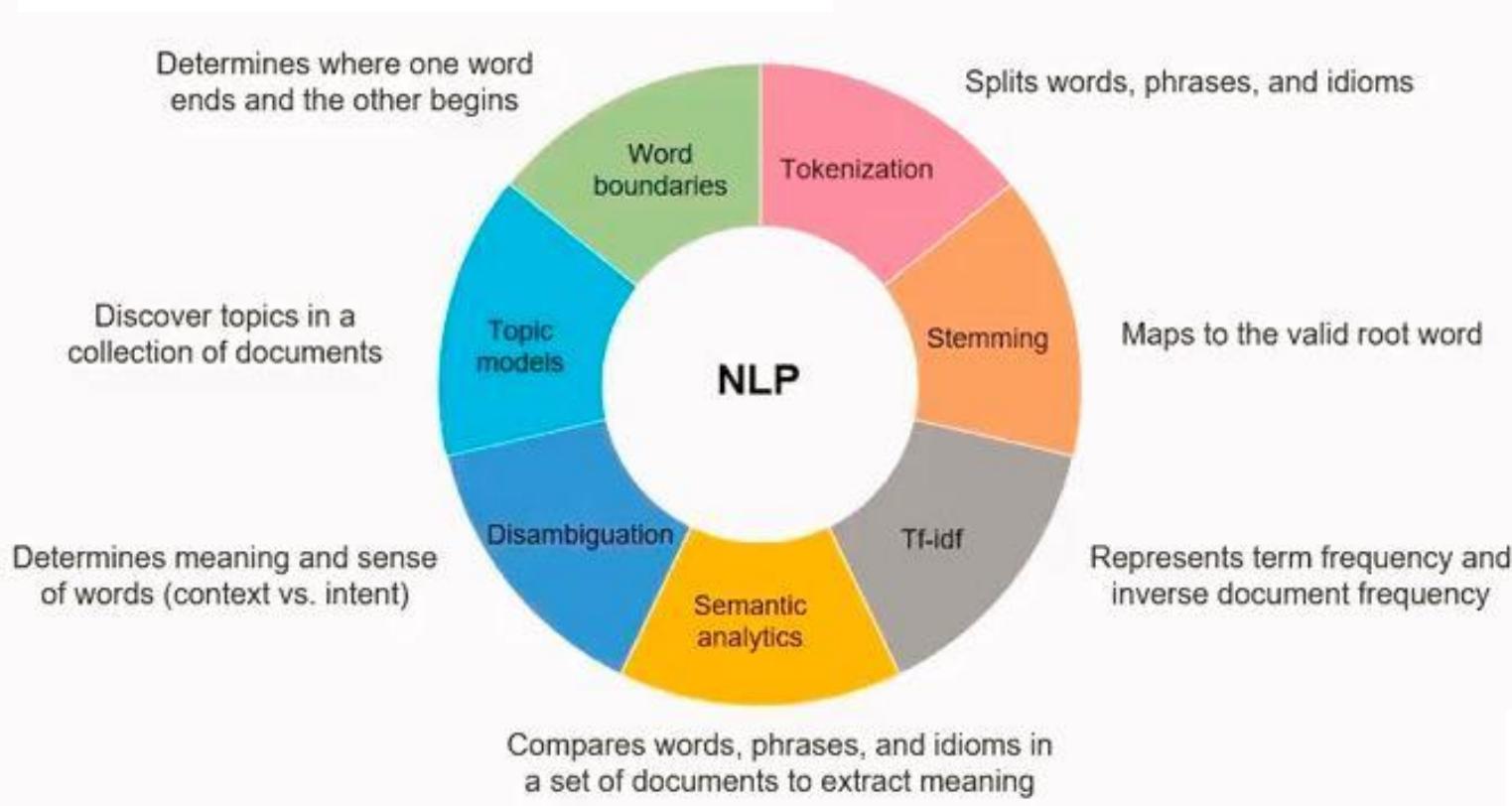
Why Natural Language Processing

- Analyzing tons of data
- Identifying various languages and dialects
- Applying quantitative analysis
- Handling ambiguities

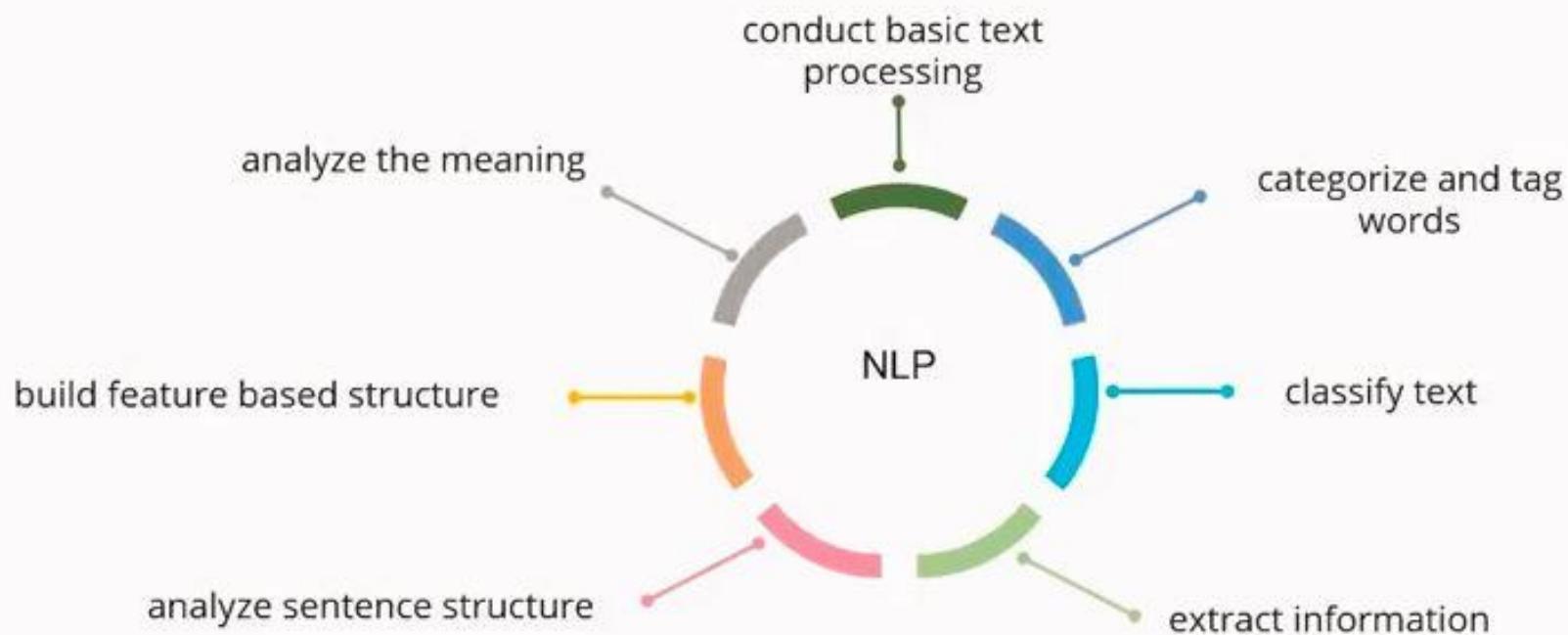


NLP Terminology

NLP terminologies:

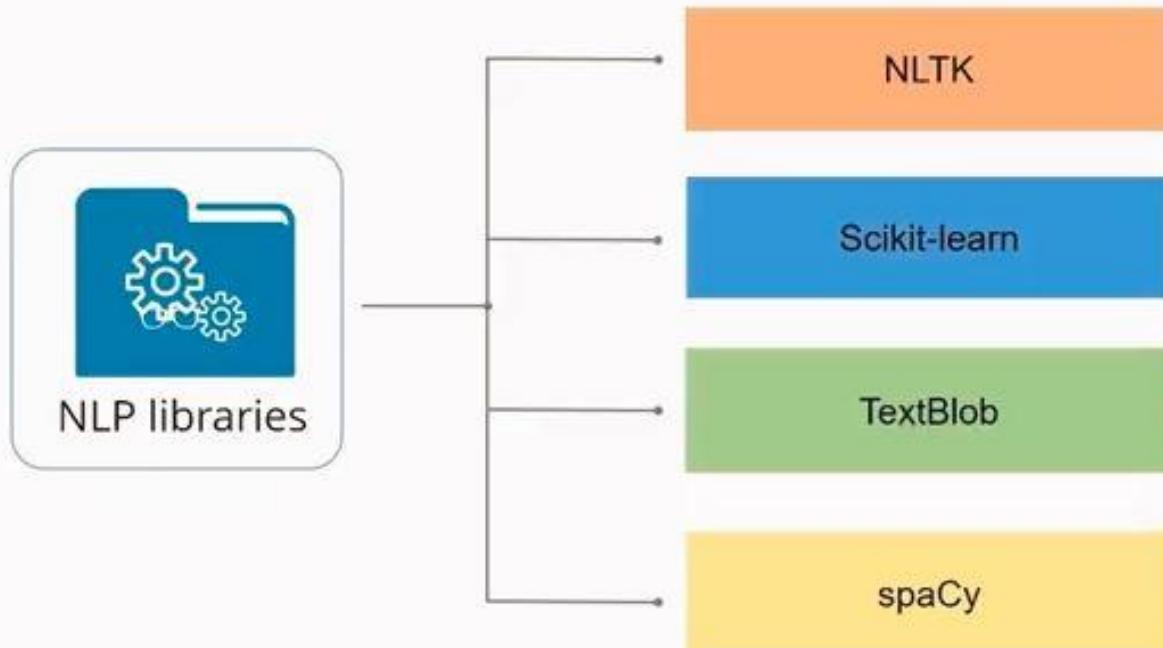


The NLP Approach for Text Data

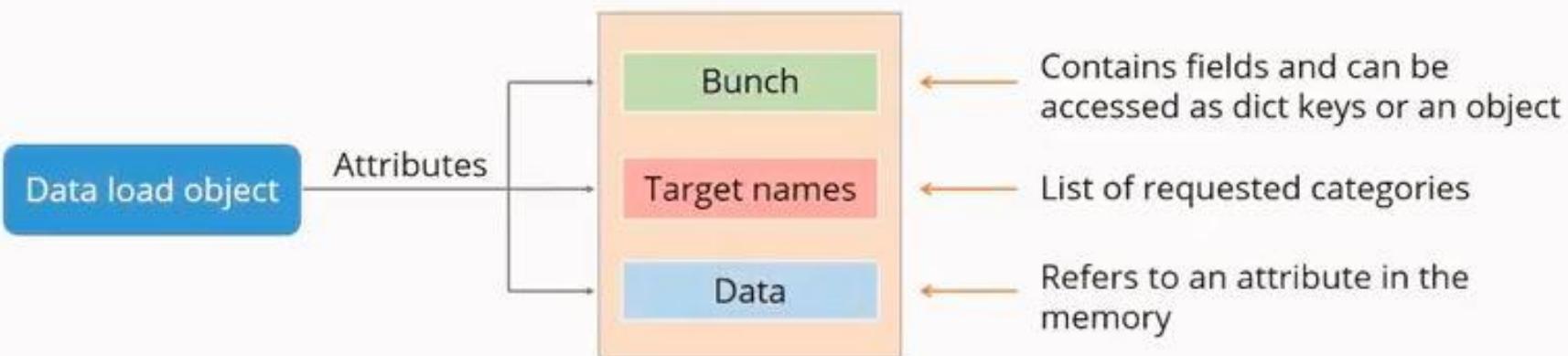


Major NLP Libraries

The major NLP libraries used in Python are:



Modules to Load Content and Category



Feature Extraction

Feature extraction is a technique to convert the content into the numerical vectors to perform machine learning.



Text feature extraction

For example: Large datasets or documents

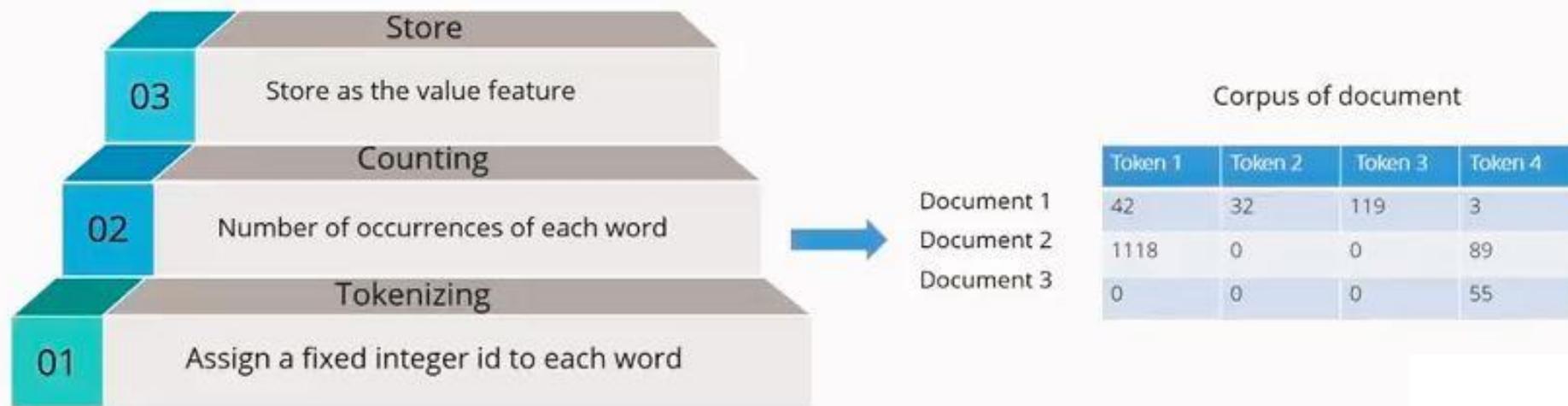


Image feature extraction

For example: Patch extraction,
hierarchical clustering

Bag of Words

Bag of words is used to convert text data into numerical feature vectors with a fixed size.



Model Training

An important task in model training is to identify the right model for the given dataset. The choice of model completely depends on the type of dataset.



Supervised

Models predict the outcome of new observations and datasets, and classify documents based on the features and response of a given dataset

Example: Naïve Bayes, SVM, linear regression, K-NN neighbors



Unsupervised

Models identify patterns in the data and extract its structure. They are also used to group documents using clustering algorithms.

Example: K-means

Naïve Bayes Classifier

Advantages

- It is efficient as it uses limited CPU and memory.
- It is fast as the model training takes less time.

Uses

- Naïve Bayes is used for sentiment analysis, email spam detection, categorization of documents, and language detection.
- Multinomial Naïve Bayes is used when multiple occurrences of the words matter.

Bayes' theorem (too) simplified

- Probability of an event A = $P(A)$ is between 0 and 1
- Bayes' theorem gives the conditional probability of an event A given event B has already occurred.

$$P(A/B) = P(A \text{ intersect } B) * P(A) / P(B)$$

- Example

- There are 100 patients
- Probability of a patient having diabetes is $P(A) = .2$
- Probability of patient having diabetes (A) given that the patient's age is > 50 (B) is $P(A/B) = .4$

Naïve Bayes Classification

- Application of Bayes' theorem to ML
- The target variable becomes event A
- The predictors become events B₁ – B_n
- We try to find P(A / B₁-B_n)

Age	BMI	Is Diabetic	
24	22	N	Probability of Is Diabetic = Y given that Age = 24 and BMI = 22
41	36	Y	Probability of Is Diabetic – Y given that Age = 41 and BMI = 36

Model building and prediction

- The model generated stores the conditional probability of the target for every possible value of the predictor.

Salary	Overall	Age						Gender	
		1 to 20	20 to 30	30 to 40	40 to 50	50 to 60	60 to 100	Female	Male
< 50K	.75	0.1	0.3	0.25	0.17	0.1	0.08	0.39	0.61
> 50K	.25	0.03	0.08	0.3	0.32	0.2	0.07	0.15	0.85
Overall		.08	.24	.26	.21	.12	.08	.33	.67

- When a new prediction needs to be done, the conditional probabilities are applied using Bayes' formula to find the probability
 - To predict for Age = 25
 - $P(\text{Salary} < 50K / \text{Age}=25) = 0.3 * 0.75 / 0.24 = \sim 0.92$
 - $P(\text{Salary} > 50K / \text{Age}=25) = 0.08 * 0.25 / 0.24 = \sim 0.08$

Summary – Naïve Bayes

Advantages

- Simple and fast
- Works well with noisy and missing data
- Provides probabilities of the result
- Very good with categorical data

Shortcomings

- Limited Accuracy
- Assumes predictors are independent
- Not good with large numeric features

Used in

- Medical diagnosis
- Spam filtering
- Document classification

Text Summarization

A NLP based approach

Text Summarization -Techniques

- A simple Natural Language Processing based approach
- A Deep NLP based approach

Text Summarization – Sample paragraph

Thank you all so very much. Thank you to the Academy. Thank you to all of you in this room. I have to congratulate the other incredible nominees this year. *The Revenant* was the product of the tireless efforts of an unbelievable cast and crew.

Text Summarization – Tokenization

1. Thank you all so very much.
2. Thank you to the Academy.
3. Thank you to all of you in this room.
4. I have to congratulate the other incredible nominees this year.
5. *The Revenant* was the product of the tireless efforts of an unbelievable cast and crew.

Text Summarization – Preprocess

1. thank you all so very much
2. thank you to the academy
3. thank you to all of you in this room
4. i have to congratulate the other incredible nominees this year
5. *the revenant* was the product of the tireless efforts of an unbelievable cast and crew

Text Summarization – Histogram

Word	Count	Word	Count	Word	Count
thank	3	in	1	revenant	1
you	4	this	2	was	1
all	2	room	1	product	1
so	1	i	1	tireless	1
very	1	have	1	efforts	1
much	1	congratulate	1	an	1
to	3	other	1	unbelievable	1
the	5 ✓	incredible	1	cast	1
academy	1.	nominees	1	and	1
of	3	year	1	crew	3

Text Summarization – Weighted Histogram

Word	Weight	Word	Weight	Word	Weight
thank	3/5	in	1/5	revenant	1/5
you	4/5	this	2/5	was	1/5
all	2/5	room	1/5	product	1/5
so	1/5	i	1/5	tireless	1/5
very	1/5	have	1/5	efforts	1/5
much	1/5	congratulate	1/5	an	1/5
to	3/5	other	1/5	unbelievable	1/5
the	5/5	incredible	1/5	cast	1/5
academy	1/5	nominees	1/5	and	1/5
of	3/5	year	1/5	crew	1/5

Maximum is
5

Text Summarization – Weighted Histogram

Word	Weight	Word	Weight	Word	Weight
thank	0.5	in	0.2	revenant	0.2
you	0.8	this	0.4	was	0.2
all	0.4	room	0.2	product	0.2
so	0.2	i	0.2	tireless	0.2
very	0.2	have	0.2	efforts	0.2
much	0.2	congratulate	0.2	an	0.2
to	0.6	other	0.2	unbelievable	0.2
the	1	incredible	0.2	cast	0.2
academy	0.2	nominees	0.2	and	0.2
of	0.6	year	0.2	crew	0.2

Text Summarization – Sentence Scores

thank	0.5
you	0.8
all	0.4
so	0.2
very	0.2
much	0.2
	2.3

Text Summarization – Sentence Scores

Sentence	Score
thank you all so very much	2.3
thank you to the academy	3.1
thank you to all of you in this room	4.3
i have to congratulate the other incredible nominees this year	3.4
<i>the revenant</i> was the product of the tireless efforts of an unbelievable cast and crew	6.2

Text Summarization – Sort by Score

Sentence	Score
<i>the revenant</i> was the product of the tireless efforts of an unbelievable cast and crew .	6.2
thank you to all of you in this room	4.3
i have to congratulate the other incredible nominees this year	3.4
thank you to the academy	3.1
thank you all so very much	2.3

Text Summarization – Pickle N Largest

Sentence	Score
<i>the revenant</i> was the product of the tireless efforts of an unbelievable cast and crew	6.2
thank you to all of you in this room	4.3

Text Summarization – Summary done

The Revenant was the product of the tireless efforts of an unbelievable cast and crew.

Thank you to all of you in this room.

Word2Vec

Introduction to the Word2Vec

BOW, TFIDF - Problems

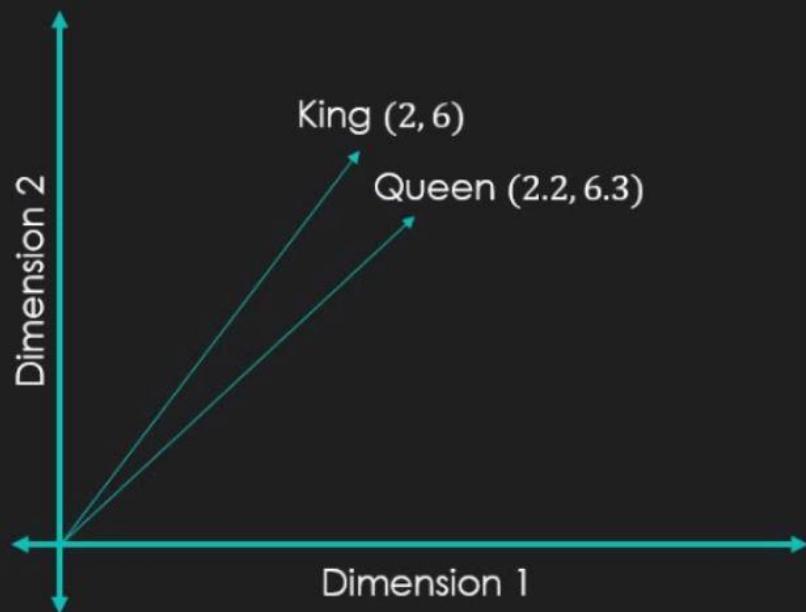
- Semantic information of the words is not stored. Even in TF-IDF model we only give more importance to the uncommon words.
- There's a chance of overfitting the model. Overfitting a scenario when model performs very well with your dataset but fails miserably when applied to any new dataset.

Word2Vec – The solution

- In this model, each word is represented as vector of 32 or more dimension instead of a single number.
- Relation between different words is preserved.

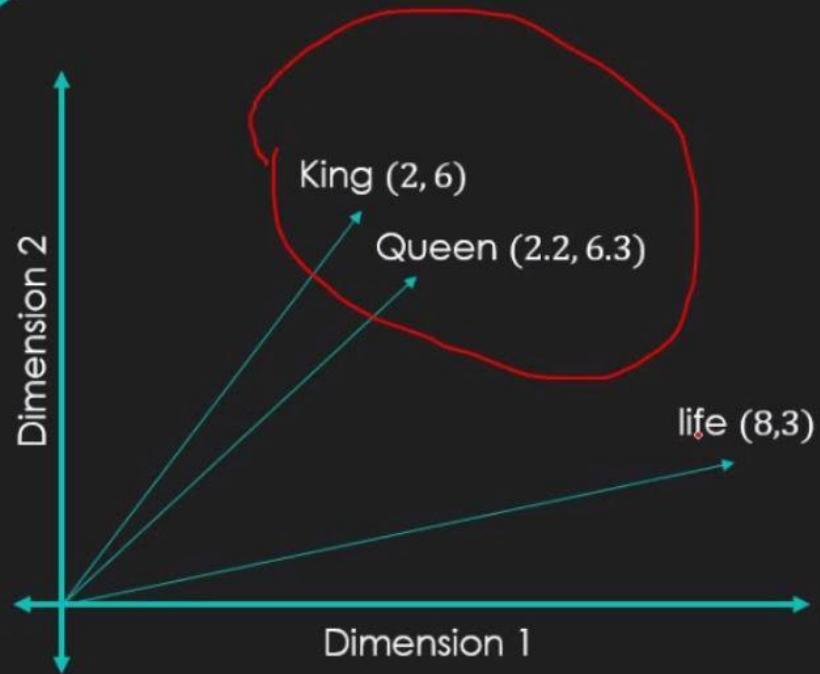
Word2Vec – Graphical Representation

2 dimensional



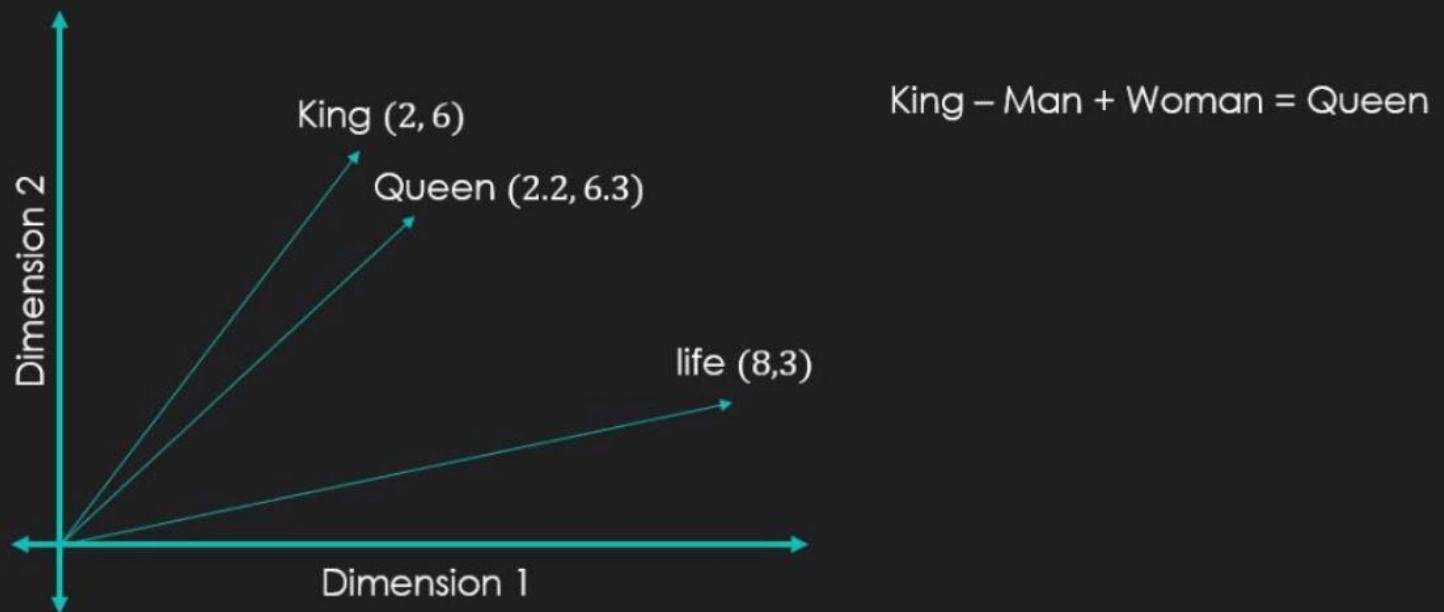
Word2Vec – Graphical Representation

2 dimensional



Word2Vec – Graphical Representation

2 dimensional



Word2Vec – Extracting sentence meaning

“Sachin Tendulkar is the Roger Federer of Cricket”

Word2Vec – Extracting sentence meaning

“Sachin Tendulkar is the Roger Federer of Cricket”

Roger Federer – tennis + cricket = Sachin Tendulkar

Word2Vec – Steps to build the model

- Scrape through a **huge** dataset like the whole Wikipedia.
- Create a matrix with all the unique words in the dataset.
The matrix represents the occurrence relation between
the words.
- Split the matrix into two thin matrices.
- We have the model.

Word2Vec – Sample Dataset

going
to
today
i
am
it
is
rain
not
outside

“it is going to rain today”
“today i am not going outside”

“i am going to watch the season premiere”

Word2Vec – Steps to build the model

Word2Vec – Word Matrix Formation

Words	going	to	today	i	am	it	is	rain	not	outside
going	3	2	2	2	2	1	1	1	1	1
to	2	2	1	1	1	1	1	1	0	0
today	2	1	2	1	1	1	1	1	1	1
i	2	1	1	2	2	0	0	0	1	1
am	2	1	1	2	2	0	0	0	1	1
it	1	1	1	0	0	1	1	1	0	0
is	1	1	1	0	0	1	1	1	0	0
rain	1	1	1	0	0	1	1	1	0	0
not	1	0	1	1	1	0	0	0	1	1
outside	1	0	1	1	1	0	0	0	1	1

Word2Vec – Splitting into smaller matrices

Words	Dimension 1	Dimension 2
going		
to		
today		
i		
am		
it		
is		
rain		
not		
outside		

Word2Vec – Splitting into smaller matrices

Words	Dimension 1	Dimension 2
going		
to		
today		
i		
am		
it		
is		
rain		
not		
outside		

Words	going	to	today	i	am	it	is	rain	not	outside
Dimension 1										
Dimension 2										

$$A * A^T$$

•

Word2Vec – Word Vectors

Words	Dimension 1	Dimension 2
going		
to		
today		
i		
am		
it		
is		
rain		
not		
outside		

Word2Vec – Word Vectors

Words	Dimension 1	Dimension 2
going	$X_{1\text{going}}$	$X_{2\text{going}}$
to	$X_{1\text{to}}$	$X_{2\text{to}}$
today	$X_{1\text{today}}$	$X_{2\text{today}}$
i	X_{1i}	X_{2i}
am	$X_{1\text{am}}$	$X_{2\text{am}}$
it	$X_{1\text{it}}$	$X_{2\text{it}}$
is	$X_{1\text{is}}$	$X_{2\text{is}}$
rain	$X_{1\text{rain}}$	$X_{2\text{rain}}$
not	$X_{1\text{not}}$	$X_{2\text{not}}$
outside	$X_{1\text{outside}}$	$X_{2\text{outside}}$

Word2Vec – Word Vectors

going = ($X_{1\text{going}}$, $X_{2\text{going}}$, ..., $X_{300\text{going}}$)

Time	Temperature
5:00 am	59 °F
6:00 am	59 °F
7:00 am	58 °F
8:00 am	58 °F
9:00 am	60 °F
10:00 am	62 °F
11:00 am	64 °F
12:00 pm	66 °F
1:00 pm	67 °F
2:00 pm	69 °F
3:00 pm	71 °F
4:00 pm	71 °F
5:00 pm	71 °F
6:00 pm	69 °F
7:00 pm	68 °F
8:00 pm	65 °F
9:00 pm	64 °F

yyyy:mm:dd hh:mm:ss

augmented dickey fuller

Time	Temperature	cloud cover	dew point	humidity	wind
5:00 am	59 °F	97%	51 °F	74%	8 mph SSE
6:00 am	59 °F	89%	51 °F	75%	8 mph SSE
7:00 am	58 °F	79%	51 °F	76%	7 mph SSE
8:00 am	58 °F	74%	51 °F	77%	7 mph S
9:00 am	60 °F	74%	51 °F	74%	7 mph S
10:00 am	62 °F	74%	52 °F	70%	8 mph S
11:00 am	64 °F	76%	52 °F	65%	8 mph SSW
12:00 pm	66 °F	80%	52 °F	60%	8 mph SSW
1:00 pm	67 °F	78%	52 °F	58%	10 mph SW
2:00 pm	69 °F	71%	52 °F	54%	10 mph SW
3:00 pm	71 °F	75%	52 °F	52%	11 mph SW
4:00 pm	71 °F	78%	52 °F	52%	11 mph SW
5:00 pm	71 °F	78%	52 °F	52%	12 mph SW
6:00 pm	69 °F	78%	52 °F	54%	11 mph SW
7:00 pm	68 °F	87%	53 °F	60%	12 mph SW
8:00 pm	65 °F	100%	54 °F	66%	11 mph SSW
9:00 pm	64 °F	100%	55 °F	72%	13 mph SSW

Variable y1	Variable y2
$y1_{t-n}$	$y2_{t-n}$
...	...
$Y1_{t-2}$	$Y2_{t-2}$
$Y1_{t-1}$	$Y2_{t-1}$
$y1_t$	$y2_t$

$$y_1(t) = a_1 + w_{111} * y_1(t-1) + w_{112} * y_2(t-1) + e_1(t-1)$$

$$y_2(t) = a_2 + w_{211} * y_1(t-1) + w_{222} * y_2(t-1) + e_2(t-1)$$

$$y(t) = a + w^*y(t-1) + e$$

Akaike Information Criterion

The Akaike Information Criterion, or AIC for short, is a method for scoring and selecting a model.

It is named for the developer of the method, Hirotugu Akaike, and may be shown to have a basis in information theory and frequentist-based inference.

This is derived from a frequentist framework, and cannot be interpreted as an approximation to the marginal likelihood.

The AIC statistic is defined for logistic regression as follows (taken from "The Elements of Statistical Learning"):

- $AIC = -2\ln(\text{likelihood}) + 2*k,$
- $AIC = -2/N * LL + 2 * k/N$

Where N is the number of examples in the training dataset, LL is the log-likelihood of the model on the training dataset, and k is the number of parameters in the model.

The score, as defined above, is minimized, e.g. the model with the lowest AIC is selected.

Bayesian Information Criterion

The Bayesian Information Criterion, or BIC for short, is a method for scoring and selecting a model.

It is named for the field of study from which it was derived: Bayesian probability and inference. Like AIC, it is appropriate for models fit under the maximum likelihood estimation framework.

The BIC statistic is calculated for logistic regression as follows (taken from "The Elements of Statistical Learning"):

- $BIC = -2 * LL + \log(N) * k$

Where $\log()$ has the base-e called the natural logarithm, LL is the log-likelihood of the model, N is the number of examples in the training dataset, and k is the number of parameters in the model.

The score as defined above is minimized, e.g. the model with the lowest BIC is selected.

Arima:<https://towardsdatascience.com/arima-simplified-b63315f27cbc>

In [1]:

```
1 from housing.util.util import load_object
```

In [4]:

```
1 model_fp=r"D:\Project\machine_learning_project\saved_models\20220709135900\model.pkl
```

In [5]:

```
1 model=load_object(file_path=model_fp)
```

In [9]:

```
1 import pandas as pd
```

In [7]:

```
1 model.preprocessing_object.transform()
```

Out[7]:

```
ColumnTransformer(transformers=[('num_pipeline',
                                 Pipeline(steps=[('imputer',
                                                 SimpleImputer(strategy
='median'))),
                                 ('feature_generator',
                                  FeatureGenerator(columns
=['longitude',
 'latitude',
 'housing_median_age',
 'total_rooms',
 'total_bedrooms',
 'population',
 'households',
 'median_income']))),
                                 ('scaler', StandardScaler
())]),
                   ['longitude', 'latitude', 'housing_median
_age',
 'total_rooms', 'total_bedrooms', 'popula
tion',
 'households', 'median_income']),
                   ('cat_pipeline',
                    Pipeline(steps=[('impute',
                                     SimpleImputer(strategy
='most_frequent')),
                                     ('one_hot_encoder',
                                      OneHotEncoder()),
                                     ('scaler',
                                      StandardScaler(with_mean
=False))]),
                     ['ocean_proximity'])])
```

In [10]:

```
1 data_fp=r"D:\Project\machine_learning_project\housing\artifact\data_ingestion\2022-0
```

In [11]:

```
1 df=pd.read_csv(data_fp)
```

In [26]:

```
1 pred=model.predict(df)
2 pred.shape,df.shape
```

Out[26]:

((4128,), (4128, 10))

In [16]:

```
1 true=df.median_house_value
```

In [18]:

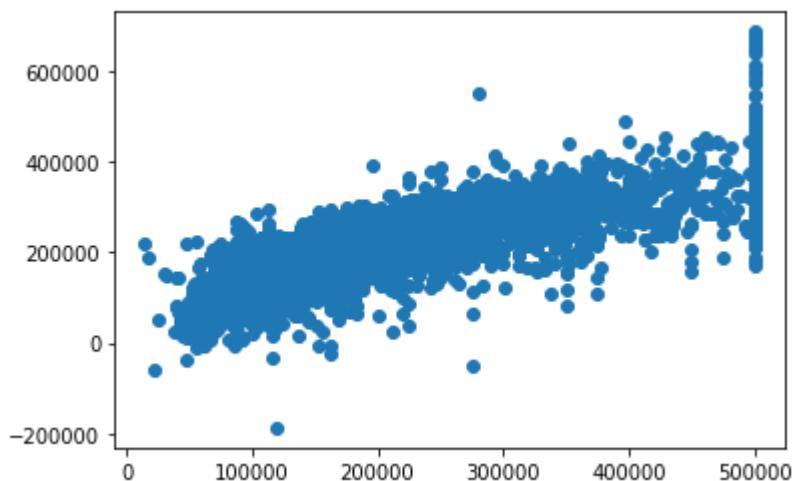
```
1 import matplotlib.pyplot as plt
```

In [22]:

```
1 plt.plot(true,pred,"o")
```

Out[22]:

[<matplotlib.lines.Line2D at 0x1b92e9ce2e8>]



In [24]:

```
1 true.shape,pred.shape
```

Out[24]:

((4128,), (4128,))

In [25]:

```
1 df.shape
```

Out[25]:

(4128, 10)

In [1]:

```
1 class A:  
2     pass
```

In [2]:

```
1 A.a=3
```

In [3]:

```
1 A.a
```

Out[3]:

3

In [4]:

```
1 setattr(A, 'b', 6)
```

In [5]:

```
1 A.b
```

Out[5]:

6

In [7]:

```
1 from sklearn.linear_model import LinearRegression
```

In [8]:

```
1 lr=LinearRegression()
```

In [9]:

```
1 lr.fit_intercept=True
```

In [12]:

```
1 setattr(lr, 'fit_intercept', False)
```

In [13]:

```
1 lr.fit_intercept
```

Out[13]:

False

In [1]:

```
1 from threading import Thread
```

In [8]:

```
1 import time
```

In [23]:

```
1 class DemoThread(Thread):
2     def __init__(self,*args,**kwargs):
3         super().__init__(daemon=False,name="demo_thread")
4
5     def print_msg(self):
6         time.sleep(10)
7         print("I was called by thread")
8
9     def run(self,):
10        self.print_msg()
11
12
```

In [24]:

```
1 a=DemoThread()
```

In [25]:

```
1 a.start()
2 print("I am wating for print msg to complete")
3 a.join()
```

I am wating for print msg to complete
I was called by thread

In [26]:

```
1 a.print_msg()
2 print("I am wating for print msg to complete")
3
```

I was called by thread
I am wating for print msg to complete

In [10]:

```
1 import os
2 os.chdir(os.pardir)
```

In [11]:

```
1 config_file_path=os.path.join(os.getcwd(),"config","config.yaml")
```

In [12]:

```
1 os.getcwd()
```

Out[12]:

```
'd:\\Project\\machine_learning_project'
```

In [13]:

```
1 from housing.pipeline.pipeline import Pipeline
```

In [14]:

```
1 from housing.config.configuration import Configuartion
```

In [15]:

```
1 config = Configuartion(config_file_path=config_file_path)
```

In []:

```
1
```

In [16]:

```
1 config.get_training_pipeline_config()
```

Out[16]:

```
TrainingPipelineConfig(artifact_dir='d:\\Project\\machine_learning_project\\notebook\\housing\\artifact')
```

In [17]:

```
1 pipeline = Pipeline(config=config)
```

In [18]:

```
1 pipeline.experiment_file_path
```

Out[18]:

```
'd:\\Project\\machine_learning_project\\notebook\\housing\\artifact\\experiment\\experiment.csv'
```

In [13]:

```
1 Pipeline.experiment
```

Out[13]:

```
Experiment(experiment_id=None, initialization_timestamp=None, artifact_time_stamp=None, running_status=None, start_time=None, stop_time=None, execution_time=None, message=None, experiment_file_path=None, accuracy=None, is_model_accepted=None)
```

In [14]:

```
1 pipeline1 = Pipeline(config=config)
```

In [17]:

```
1 pipeline1.experiment
```

Out[17]:

```
Experiment(experiment_id='cd4fec88-b4c9-4ba3-9e1c-9b73d150b426', initialization_timestamp='2022-07-10-12-09-35', artifact_time_stamp='2022-07-10-12-09-35', running_status=True, start_time=datetime.datetime(2022, 7, 10, 12, 50, 512645), stop_time=None, execution_time=None, message='Pipeline has been started.', experiment_file_path='d:\\Project\\machine_learning_project\\notebook\\housing\\artifact\\experiment\\experiment.csv', accuracy=None, is_model_accepted=None)
```

In [16]:

```
1 pipeline1.start()
```

In [19]:

```
1 pipeline.experiment
```

Out[19]:

```
Experiment(experiment_id='cd4fec88-b4c9-4ba3-9e1c-9b73d150b426', initialization_timestamp='2022-07-10-12-09-35', artifact_time_stamp='2022-07-10-12-09-35', running_status=True, start_time=datetime.datetime(2022, 7, 10, 12, 50, 512645), stop_time=None, execution_time=None, message='Pipeline has been started.', experiment_file_path='d:\\Project\\machine_learning_project\\notebook\\housing\\artifact\\experiment\\experiment.csv', accuracy=None, is_model_accepted=None)
```

In [21]:

```
1 Pipeline.experiment.experiment_file_path
```

Out[21]:

```
'd:\\Project\\machine_learning_project\\notebook\\housing\\artifact\\experiment\\experiment.csv'
```

In [22]:

```
1 pipeline.start()
```

In [25]:

```
1 pipeline.run_pipeline()
```

Out[25]:

```
Experiment(experiment_id='cd4fec88-b4c9-4ba3-9e1c-9b73d150b426', initialization_timestamp='2022-07-10-12-09-35', artifact_time_stamp='2022-07-10-12-09-35', running_status=True, start_time=datetime.datetime(2022, 7, 10, 12, 50, 512645), stop_time=None, execution_time=None, message='Pipeline has been started.', experiment_file_path='d:\\Project\\machine_learning_project\\notebook\\housing\\artifact\\experiment\\experiment.csv', accuracy=None, is_model_accepted=None)
```

In [1]:

```
1 from housing.pipeline.pipeline import Pipeline
```

In []:

```
1 pipeline()
```

In [20]:

```
1 Pipeline.get_experiments_status(limit=10)
```

Out[20]:

	experiment_id	artifact_time_stamp	running_status	start_time	stop_time	execution_time
0	cd4fec88-b4c9-4ba3-9e1c-9b73d150b426	2022-07-10-09-35	True	2022-07-10 12:12:50.512645		NaN

In [1]:

```
1 file_path= r"D:\\Project\\machine_learning_project\\housing\\artifact\\data_ingestion\\2022\\train.csv"
```

In [2]:

```
1 import pandas as pd
```

In [3]:

```
1 df=pd.read_csv(file_path)
```

In [4]:

```
1 df.shape
```

Out[4]:

```
(4128, 10)
```

In [5]:

```
1 from housing.entity.housing_predictor import HousingPredictor
```

In [6]:

```
1 housing_predictor = HousingPredictor(model_dir=r"D:\Project\machine_learning_project")
```

In [8]:

```
1 pred=housing_predictor.predict(df)
```

In [11]:

```
1 df.shape
```

Out[11]:

(4128, 10)

In [10]:

```
1 pred.shape
```

Out[10]:

(4128,)

In [17]:

```
1 signle_record= pd.DataFrame(df[:1].to_dict())
```

In [18]:

```
1 housing_predictor.predict(signle_record)
```

Out[18]:

array([424327.43173199])

In [19]:

```
1 housing_predictor.get_latest_model_path()
```

Out[19]:

'D:\\\\Project\\\\machine_learning_project\\\\saved_models\\\\20220710130438\\\\model.pkl'

In []:

```
1
```


In [1]:

```
1 import os  
2  
3 os.chdir(os.pardir)
```

In [2]:

```
1 from housing.pipeline.pipeline import Pipeline
```

In [3]:

```
1 p=Pipeline()
```

In [4]:

```
1 p
```

Out[4]:

```
<Pipeline(pipeline, initial)>
```

In [5]:

```
1 p.start()
```

```
Exception in thread pipeline:  
Traceback (most recent call last):  
  File "/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningP  
roject/machine_learning_project/housing/pipeline/pipeline.py", line 183, i  
n save_experiment  
    { "experiment_file_path": os.path.basename(experiment_dict["experiment  
_file_path"])),  
  File "/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningP  
roject/machine_learning_project/venv/lib/python3.7 posixpath.py", line 14  
6, in basename  
    p = os.fspath(p)  
TypeError: expected str, bytes or os.PathLike object, not list
```

The above exception was the direct cause of the following exception:

```
Traceback (most recent call last):  
  File "/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningP  
roject/machine_learning_project/housing/pipeline/pipeline.py", line 132, i  
n run_pipeline  
    self.save_experiment()  
  File "/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningP  
roject/machine_learning_project/housing/pipeline/pipeline.py", line 196, i  
n save_experiment  
    raise HousingException(e,sys) from e  
housing.exception.HousingException:  
    Error occurred in script:  
    [ /home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearnin  
gProject/machine_learning_project/housing/pipeline/pipeline.py ] at  
        try block line number: [183] and exception block line number: [19  
6]  
        error message: [expected str, bytes or os.PathLike object, not lis  
t]
```

The above exception was the direct cause of the following exception:

```
Traceback (most recent call last):  
  File "/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningP  
roject/machine_learning_project/venv/lib/python3.7/threading.py", line 91  
7, in _bootstrap_inner  
    self.run()  
  File "/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningP  
roject/machine_learning_project/housing/pipeline/pipeline.py", line 173, i  
n run  
    raise e  
  File "/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningP  
roject/machine_learning_project/housing/pipeline/pipeline.py", line 171, i  
n run  
    self.run_pipeline()  
  File "/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningP  
roject/machine_learning_project/housing/pipeline/pipeline.py", line 167, i  
n run_pipeline  
    raise HousingException(e, sys) from e  
housing.exception.HousingException:  
    Error occurred in script:  
    [ /home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearnin  
gProject/machine_learning_project/housing/pipeline/pipeline.py ] at  
        try block line number: [132] and exception block line number: [16  
7]  
        error message: [  
        Error occurred in script:
```

```
[ /home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearnin
gProject/machine_learning_project/housing/pipeline/pipeline.py ] at
    try block line number: [183] and exception block line number: [19
6]
        error message: [expected str, bytes or os.PathLike object, not lis
t]
    ]
```

In [1]:

```
1 f="/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningProject/machine_
```

In [2]:

```
1 from housing.util.util import read_yaml_file
```

In [4]:

```
1 import json
```

In [9]:

```
1 json.dump(read_yaml_file(f),open("sample.json","w"),indent=4)
```

In [11]:

```
1 data="""{
2     "grid_search": {
3         "class": "GridSearchCV",
4         "module": "sklearn.model_selection",
5         "params": {
6             "cv": 4,
7             "verbose": 2
8         }
9     },
10    "model_selection": {
11        "module_0": {
12            "class": "LinearRegression",
13            "module": "sklearn.linear_model",
14            "params": {
15                "fit_intercept": true
16            },
17            "search_param_grid": {
18                "fit_intercept": [
19                    true
20                ]
21            }
22        },
23        "module_1": {
24            "class": "RandomForestRegressor",
25            "module": "sklearn.ensemble",
26            "params": {
27                "min_samples_leaf": 2
28            },
29            "search_param_grid": {
30                "min_samples_leaf": [
31                    2
32                ],
33                "n_estimators": [
34                    10
35                ]
36            }
37        }
38    }
39 }"""
```

In [12]:

```
1 import json
```

In [32]:

```
1 json.loads(a)

-----
-
JSONDecodeError                                     Traceback (most recent call las
t)
/tmp/ipykernel_13318/2214753276.py in <module>
----> 1 json.loads(a)

~/iNeuron_Private_Intelligence_Limited/MachineLearningProject/machine_lear
ning_project/venv/lib/python3.7/json/__init__.py in loads(s, encoding, cl
s, object_hook, parse_float, parse_int, parse_constant, object_pairs_hook,
**kw)
  346         parse_int is None and parse_float is None and
  347         parse_constant is None and object_pairs_hook is None a
nd not kw):
--> 348         return _default_decoder.decode(s)
  349     if cls is None:
  350         cls = JSONDecoder

~/iNeuron_Private_Intelligence_Limited/MachineLearningProject/machine_lear
ning_project/venv/lib/python3.7/json/decoder.py in decode(self, s, _w)
  335
  336     """
--> 337     obj, end = self.raw_decode(s, idx=_w(s, 0).end())
  338     end = _w(s, end).end()
  339     if end != len(s):

~/iNeuron_Private_Intelligence_Limited/MachineLearningProject/machine_lear
ning_project/venv/lib/python3.7/json/decoder.py in raw_decode(self, s, id
x)
  353         obj, end = self.scan_once(s, idx)
  354     except StopIteration as err:
--> 355         raise JSONDecodeError("Expecting value", s, err.value)
from None
  356     return obj, end

JSONDecodeError: Expecting value: line 1 column 246 (char 245)
```

In [17]:

```
1 data="""{'grid_search': {'class': 'GridSearchCV',
 2   'module': 'sklearn.model_selection',
 3   'params': {'cv': 4, 'verbose': 2}},
 4   'model_selection': {'module_0': {'class': 'LinearRegression',
 5     'module': 'sklearn.linear_model',
 6     'params': {'fit_intercept': True},
 7     'search_param_grid': {'fit_intercept': [True]}},
 8   'module_1': {'class': 'RandomForestRegressor',
 9     'module': 'sklearn.ensemble',
10     'params': {'min_samples_leaf': 2},
11     'search_param_grid': {'min_samples_leaf': [2], 'n_estimators': [10]}}}}"""
```

In [33]:

```
1 a=data.replace("",'').replace("\n",'')
```

In [27]:

```
1 len(data[245])
```

Out[27]:

1

In [37]:

```
1
```

Expecting value: line 1 column 246 (char 245)

In []:

```
1
```

In [1]:

```
1 from collections import namedtuple
```

1. Download url
2. Download folder (compressed file)
3. Extract folder (extracted file))
4. Train dataset folder
5. Test dataset folder

In [3]:

```
1 DataIngestionConfig=namedtuple("DataIngestionConfig",
2 ["dataset_download_url","tgz_download_dir","raw_data_dir","ingested_train_dir","inge
```

In [5]:

```
1 data_ingestion_config = DataIngestionConfig(dataset_download_url='asfasdf',
2 tgz_download_dir='asdasd',
3 raw_data_dir='asdas',
4 ingested_train_dir='asdbfk',
5 ingested_test_dir='sadnjk'
6 )
```

In [6]:

```
1 data_ingestion_config
```

Out[6]:

```
DataIngestionConfig(dataset_download_url='asfasdf', tgz_download_dir='asdasd', raw_data_dir='asdas', ingested_train_dir='asdbfk', ingested_test_dir='sadnjk')
```

In [7]:

```
1 ("sdfjnksdf","sdwjkuf","asdfasd","wsdfbkiasd")
```

Out[7]:

```
('sdfjnksdf', 'sdwjkuf', 'asdfasd', 'wsdfbkiasd')
```

In [1]:

```
1 import yaml
```

In [2]:

```
1 import os
```

In [3]:

```
1 os.getcwd()
```

Out[3]:

```
'd:\\Project\\machine_learning_project\\notebook'
```

In [4]:

```
1 os.chdir("d:\\Project\\machine_learning_project")
```

In [5]:

```
1 os.getcwd()
```

Out[5]:

```
'd:\\Project\\machine_learning_project'
```

In []:

```
1
```

In [6]:

```
1 config_file_path=os.path.join("config","config.yaml")
```

In [7]:

```
1 config_file_path
```

Out[7]:

```
'config\\config.yaml'
```

In [8]:

```
1 os.path.exists(config_file_path)
```

Out[8]:

```
True
```

In [17]:

```
1 os.getcwd()
```

Out[17]:

```
'd:\\Project\\machine_learning_project'
```

In [18]:

```
1 config_info=None
2 with open(config_file_path, "rb") as yaml_file:
3     config_info=yaml.safe_load(yaml_file)
4
```

In [20]:

```
1 config_info["data_ingestion_config"]
```

Out[20]:

```
{'dataset_download_url': 'https://raw.githubusercontent.com/ageron/handson-
ml/master/datasets/housing/housing.tgz',
'raw_data_dir': 'raw_data',
'tgz_download_dir': 'tgz_data',
'ingested_dir': 'ingested_data',
'ingested_train_dir': 'train',
'ingested_test_dir': 'test'}
```

In [21]:

```
1 def read_yaml_file(file_path:str)->dict:
2     """
3         Reads a YAML file and returns the contents as a dictionary.
4         file_path: str
5     """
6     try:
7         with open(file_path, 'rb') as yaml_file:
8             return yaml.safe_load(yaml_file)
9     except Exception as e:
10        raise e
```

In [23]:

```
1 config =read_yaml_file(config_file_path)
```

In [2]:

```
1 from housing.constant import *
```

In [26]:

```
1 TRAINING_PIPELINE_CONFIG_KEY
```

Out[26]:

```
'training_pipeline_config'
```

In [29]:

```
1 config[TRAINING_PIPELINE_CONFIG_KEY]
```

Out[29]:

```
{'pipeline_name': 'housing', 'artifact_dir': 'artifact'}
```

In [28]:

```
1 config[TRAINING_PIPELINE_CONFIG_KEY][TRAINING_PIPELINE_NAME_KEY]
```

Out[28]:

```
'housing'
```

In [31]:

```
1 training_pipeline_config = config[TRAINING_PIPELINE_CONFIG_KEY]
2 artifact_dir = os.path.join(ROOT_DIR,
3     training_pipeline_config[TRAINING_PIPELINE_NAME_KEY],
4     training_pipeline_config[TRAINING_PIPELINE_ARTIFACT_DIR_KEY]
5 )
```

In [33]:

```
1 ROOT_DIR
```

Out[33]:

```
'd:\\Project\\machine_learning_project'
```

In [34]:

```
1 training_pipeline_config[TRAINING_PIPELINE_NAME_KEY]
```

Out[34]:

```
'housing'
```

In [35]:

```
1 training_pipeline_config[TRAINING_PIPELINE_ARTIFACT_DIR_KEY]
```

Out[35]:

```
'artifact'
```

In [32]:

```
1 artifact_dir
```

Out[32]:

```
'd:\\Project\\machine_learning_project\\housing\\artifact'
```

In [10]:

```
1 from housing.config.configuration import Configuartion
```

In [13]:

```
1 os.getcwd()
```

Out[13]:

```
'd:\\Project\\machine_learning_project'
```

In [12]:

```
1 config = Configuartion(config_file_path="d:\\Project\\machine_learning_project\\conf
```

In [29]:

```
1 training_pipeline_config=config.get_training_pipeline_config()
```

In [31]:

```
1 artifact_dir = training_pipeline_config.artifact_dir
```

In [32]:

```
1 DATA_INGESTION_ARTIFACT_DIR
```

Out[32]:

'data_ingestion'

In [33]:

```
1 CURRENT_TIME_STAMP
```

Out[33]:

'2022-06-25-12-58-04'

In [11]:

```
1 from housing.constant import *
```

In [23]:

```
1 data_ingestion_info=config.config_info[DATA_INGESTION_CONFIG_KEY]
```

In [24]:

```
1 data_ingestion_info
```

Out[24]:

```
{'dataset_download_url': 'https://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing/housing.tgz',
'raw_data_dir': 'raw_data',
'tgz_download_dir': 'tgz_data',
'ingested_dir': 'ingested_data',
'ingested_train_dir': 'train',
'ingested_test_dir': 'test'}
```

In [26]:

```
1
```

Out[26]:

'dataset_download_url'

In [6]:

```
1 from housing.constant import DATA_INGESTION_CONFIG_KEY
```

In [7]:

```
1 DATA_INGESTION_CONFIG_KEY
```

Out[7]:

```
'data_ingestion_config'
```

In [27]:

```
1 data_ingestion_info[DATA_INGESTION_DOWNLOAD_URL_KEY]
```

Out[27]:

```
'https://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing/housing.tgz'
```

In [28]:

```
1 data_ingestion_info
```

Out[28]:

```
{'dataset_download_url': 'https://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing/housing.tgz',
 'raw_data_dir': 'raw_data',
 'tgz_download_dir': 'tgz_data',
 'ingested_dir': 'ingested_data',
 'ingested_train_dir': 'train',
 'ingested_test_dir': 'test'}
```

In [14]:

```
1 config.get_data_ingestion_config()
```

```
-  
AttributeError Traceback (most recent call last)  
t)  
d:\Project\machine_learning_project\housing\config\configuration.py in get  
_data_ingestion_config(self)  
    32         )  
---> 33     data_ingestion_info = self.config_info[DATA_INGESTION_  
CONFIG_KEY]  
    34
```

AttributeError: 'dict' object has no attribute 'config_info'

The above exception was the direct cause of the following exception:

```
HousingException Traceback (most recent call last)  
t)  
~\AppData\Local\Temp\ipykernel_14708\2719117554.py in <module>  
----> 1 config.get_data_ingestion_config()  
  
d:\Project\machine_learning_project\housing\config\configuration.py in get  
_data_ingestion_config(self)  
    66         return data_ingestion_config  
    67     except Exception as e:  
---> 68         raise HousingException(e,sys) from e  
    69  
    70     def get_data_validation_config(self) -> DataValidationConfig:
```

HousingException: Error occurred in script: [d:\Project\machine_learning_project\housing\config\configuration.py] at line number: [68] error message: ['dict' object has no attribute 'config_info']

In []:

```
1
```

In [2]:

```
1 from housing.config.configuration import Configuartion
```

In [3]:

```
1 config = Configuartion(config_file_path="d:\\Project\\machine_learning_project\\conf
```

In [4]:

```
1 config.get_data_ingestion_config()
```

Out[4]:

```
DataIngestionConfig(dataset_download_url='https://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing/housing.tgz', tgz_download_dir='d:\\Project\\machine_learning_project\\notebook\\housing\\artifact\\data_ingestion\\2022-06-25-13-25-32\\tgz_data', raw_data_dir='d:\\Project\\machine_learning_project\\notebook\\housing\\artifact\\data_ingestion\\2022-06-25-13-25-32\\raw_data', ingested_train_dir='d:\\Project\\machine_learning_project\\notebook\\housing\\artifact\\data_ingestion\\2022-06-25-13-25-32\\ingested_data\\train', ingested_test_dir='d:\\Project\\machine_learning_project\\notebook\\housing\\artifact\\data_ingestion\\2022-06-25-13-25-32\\ingested_data\\test')
```

Type *Markdown* and *LaTeX*: α^2

In []:

```
1
```

In [2]:

```
1 url= 'https://raw.githubusercontent.com/ageron/handson-ml/master/datasets/housing/ho
```

In [3]:

```
1 import os
```

In [4]:

```
1 os.path.basename(url)
```

Out[4]:

'housing.tgz'

In [5]:

```
1 file_path=r"D:\Project\machine_learning_project\config"
```

In [8]:

```
1 os.listdir(file_path)[0]
```

Out[8]:

'config.yaml'

In [1]:

```
1 ## Data Ingestion
```

In [2]:

```
1 import pandas as pd
```

In [3]:

```
1 csv_file_path=r"D:\Project\machine_learning_project\housing\artifact\data_ingestion\"  
2 housing_data_frame=pd.read_csv(csv_file_path)
```

In [4]:

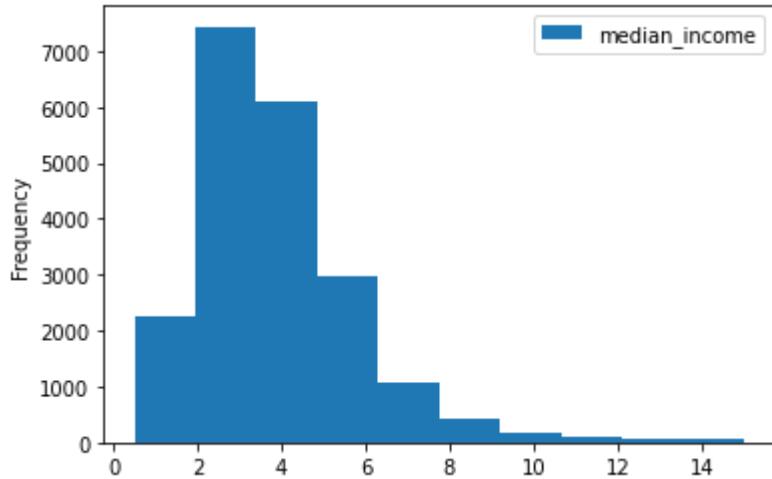
```
1 import matplotlib.pyplot as plt
```

In [5]:

```
1 housing_data_frame[["median_income"]].plot(kind="hist")
```

Out[5]:

```
<AxesSubplot:ylabel='Frequency'>
```



In [6]:

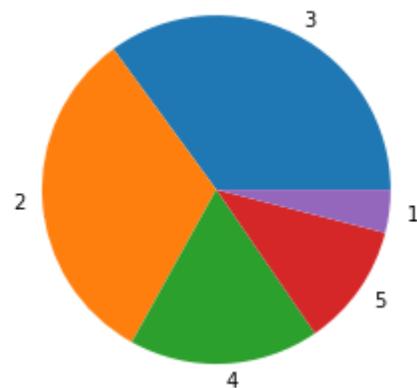
```
1 import numpy as np
```

In [7]:

```
1 housing_data_frame["income_cat"] = pd.cut(
2     housing_data_frame["median_income"],
3     bins=[0.0, 1.5, 3.0, 4.5, 6.0, np.inf],
4     labels=[1,2,3,4,5]
5 )
```

In [8]:

```
1 plt.pie(housing_data_frame.income_cat.value_counts(),labels=housing_data_frame.incom
2 plt.show()
```



In [9]:

```
1 from sklearn.model_selection import StratifiedShuffleSplit
```

In [10]:

```
1 split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
```

In [11]:

```
1 for train_ix,test_ix in split.split(housing_data_frame, housing_data_frame["income_c  
2     print(train_ix,test_ix)
```

```
[12655 15502 2908 ... 19263 19140 19773] [ 5241 17352 3505 ... 17223 107  
86 3965]
```

In [12]:

```
1 train_df=housing_data_frame.loc[train_ix]
```

In [13]:

```
1 housing_data_frame.shape
```

Out[13]:

```
(20640, 11)
```

In [14]:

```
1 train_df.shape
```

Out[14]:

```
(16512, 11)
```

In [15]:

```
1 test_df=housing_data_frame.loc[test_ix]
```

In [16]:

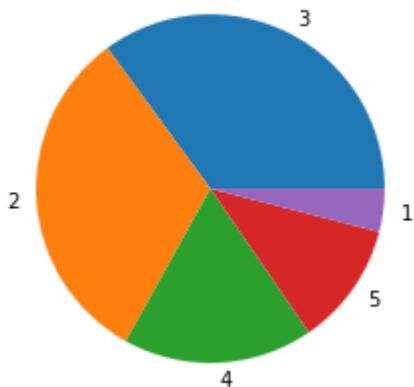
```
1 test_df.shape
```

Out[16]:

```
(4128, 11)
```

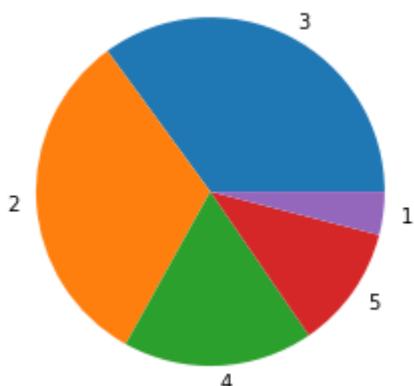
In [17]:

```
1 plt.pie(train_df.income_cat.value_counts(),labels=train_df.income_cat.value_counts())
2 plt.show()
```



In [18]:

```
1 plt.pie(test_df.income_cat.value_counts(),labels=test_df.income_cat.value_counts().i
2 plt.show()
```



In [27]:

```
1 housing_data_frame.dtypes.index,housing_data_frame.dtypes.values
```

Out[27]:

```
(Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity', 'income_cat'],
      dtype='object'),
array([dtype('float64'), dtype('float64'), dtype('float64'),
       dtype('float64'), dtype('float64'), dtype('float64'),
       dtype('float64'), dtype('float64'), dtype('float64'), dtype('O'),
       CategoricalDtype(categories=[1, 2, 3, 4, 5], ordered=True)],
      dtype=object))
```

In [30]:

```
1 data_type = list(map(lambda x:str(x).replace("dtype('')","").replace("'", "") ,housing
```

In [31]:

```
1 column=housing_data_frame.columns
```

In [32]:

```
1 dict(zip(column,data_type))
```

Out[32]:

```
{'longitude': 'float64',
'latitude': 'float64',
'housing_median_age': 'float64',
'total_rooms': 'float64',
'total_bedrooms': 'float64',
'population': 'float64',
'households': 'float64',
'median_income': 'float64',
'median_house_value': 'float64',
'ocean_proximity': 'object',
'income_cat': 'category'}
```

In []:

```
1
```

In [1]:

```
1 from scipy.stats import ks_2samp
```

In [2]:

```
1 import numpy as np
```

In [16]:

```
1 arr1 = np.arange(10)
2 arr2=np.arange(10)
```

In [17]:

```
1 arr1
```

Out[17]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [18]:

```
1 arr2
```

Out[18]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [19]:

```
1 res = ks_2samp(arr1,arr2)
```

Type *Markdown* and *LaTeX*: α^2

In [20]:

```
1 round(res.pvalue)
```

Out[20]:

```
1
```

In []:

```
1 Null: Two datset are from same distribution
2
3 Alterate: Two dataset are not from same distribution
4
5
6 if p>=0.05 :
7     We have sufficient proof that null hypothesis is True
8 else:
9     We don't have sufficient proof that null hypothesis is True
10
```

In [25]:

```
1 res= ks_2samp(arr1,arr2)
2 round(res.pvalue,3)
```

Out[25]:

1.0

In []:

```
1
```

In [1]:

```
1 import pandas as pd
```

In [2]:

```
1 f="/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningProject/machine_
```

In [10]:

```
1
```

In [21]:

```
1
```

Out[21]:

	Time stamp	Log Level	line number	file name	function name	
0	[2022-07-06 19:40:36,738]	INFO	226	configuration.py	get_training_pipeline_config()	Trai
1	[2022-07-06 19:40:36,743]	INFO	224	_internal.py	_log()	* Rur
2	[2022-07-06 19:40:42,739]	INFO	224	_internal.py	_log()	127.
3	[2022-07-06 19:40:45,899]	INFO	224	_internal.py	_log()	127
4	[2022-07-06 19:40:50,204]	INFO	226	configuration.py	get_training_pipeline_config()	Trai
...
134	[2022-07-06 19:40:55,171]	INFO	38	model_pusher.py	export_model()	Mod
135	[2022-07-06 19:40:55,171]	INFO	50	model_pusher.py	__del__()	>>>>>>>
136	[2022-07-06 19:40:55,171]	INFO	149	pipeline.py	run_pipeline()	Mod
137	[2022-07-06 19:40:55,171]	INFO	152	pipeline.py	run_pipeline()	
138	[2022-07-06 19:40:55,171]	INFO	166	pipeline.py	run_pipeline()	Pipeline

139 rows × 6 columns

In [20]:

```
1
```

In []:

```
1
```

In [12]:

```
1 df=pd.DataFrame(data)
```

In [17]:

```
1
```

In [18]:

```
1 df.columns=columns
```

In []:

```
1
```

In []:

```
1
```

```
machine_learning_project/.github/workflows  
/main.yaml
```

```
# Your workflow name.  
  
name: Deploy to heroku.  
  
# Run workflow on every push to main branch.  
  
on:  
  
push:  
  
branches: [main]
```

```
# Your workflows jobs.  
  
jobs:  
  
build:  
  
runs-on: ubuntu-latest  
  
steps:  
  
# Check-out your repository.  
  
- name: Checkout  
  
uses: actions/checkout@v2
```

↓ IMPORTANT PART ↓

```
- name: Build, Push and Release a Docker container to Heroku. # Your custom step name  
  
uses: gonuit/heroku-docker-deploy@v1.3.3 # GitHub action name (leave it as it is).  
  
with:
```

```
# Below you must provide variables for your Heroku app.
```

```
# The email address associated with your Heroku account.
```

```
# If you don't want to use repository secrets (which is recommended) you can do:
```

```
# email: my.email@example.com
```

```
email: ${{ secrets.HEROKU_EMAIL }}

# Heroku API key associated with provided user's email.
# Api Key is available under your Heroku account settings.

heroku_api_key: ${{ secrets.HEROKU_API_KEY }}

# Name of the heroku application to which the build is to be sent.

heroku_app_name: ${{ secrets.HEROKU_APP_NAME }}

# (Optional, default: "./")

# Dockerfile directory.

# For example, if you have a Dockerfile in the root of your project, leave it as follows:

dockerfile_directory: ./

# (Optional, default: "Dockerfile")

# Dockerfile name.

dockerfile_name: Dockerfile

# (Optional, default: "")

# Additional options of docker build command.

docker_options: "--no-cache"

# (Optional, default: "web")

# Select the process type for which you want the docker container to be uploaded.

# By default, this argument is set to "web".

# For more information look at https://devcenter.heroku.com/articles/process-model

process_type: web
```

↑ IMPORTANT PART ↑

[machine_learning_project/config/config.yaml](#)

```
training_pipeline_config:  
  pipeline_name: housing  
  artifact_dir: artifact  
  
data_ingestion_config:  
  dataset_download_url: https://raw.githubusercontent.com/ageron/handson-  
    ml/master/datasets/housing/housing.tgz  
  raw_data_dir: raw_data  
  tgz_download_dir: tgz_data  
  ingested_dir: ingested_data  
  ingested_train_dir: train  
  ingested_test_dir: test  
  
data_validation_config:  
  schema_dir: config  
  schema_file_name: schema.yaml  
  report_file_name: report.json  
  report_page_file_name: report.html  
  
data_transformation_config:  
  add_bedroom_per_room: true  
  transformed_dir: transformed_data  
  transformed_train_dir: train  
  transformed_test_dir: test  
  preprocessing_dir: preprocessed  
  preprocessed_object_file_name: preprocessed.pkl  
  
model_trainer_config:  
  trained_model_dir: trained_model  
  model_file_name: model.pkl
```

```
base_accuracy: 0.6
model_config_dir: config
model_config_file_name: model.yaml

model_evaluation_config:
  model_evaluation_file_name: model_evaluation.yaml

model_pusher_config:
  model_export_dir: saved_models
```

[machine_learning_project](#) /config//model.yaml

```
grid_search:  
  class: GridSearchCV  
  module: sklearn.model_selection  
  params:  
    cv: 5  
    verbose: 2  
  
model_selection:  
  module_0:  
    class: LinearRegression  
    module: sklearn.linear_model  
    params:  
      fit_intercept: true  
      search_param_grid:  
        fit_intercept:  
          - true  
          - false  
  
    module_1:  
      class: RandomForestRegressor  
      module: sklearn.ensemble  
      params:  
        min_samples_leaf: 3  
        search_param_grid:  
          min_samples_leaf:  
            - 6
```

1. [machine_learning_project/config/schema.yaml](#)

columns:

```
longitude: float  
latitude: float  
housing_median_age: float  
total_rooms: float  
total_bedrooms: float  
population: float  
households: float  
median_income: float  
median_house_value: float  
ocean_proximity: category
```

numerical_columns:

- longitude
- latitude
- housing_median_age
- total_rooms
- total_bedrooms
- population
- households
- median_income

categorical_columns:

- ocean_proximity

target_column: median_house_value

domain_value:

ocean_proximity:

- <1H OCEAN

- INLAND

- ISLAND

- NEAR BAY

- NEAR OCEAN

[machine_learning_project/housing/component/data_ ingestion.py](#)

```
from housing.entity.config_entity import DataIngestionConfig
import sys,os
from housing.exception import HousingException
from housing.logger import logging
from housing.entity.artifact_entity import DataIngestionArtifact
import tarfile
import numpy as np
from six.moves import urllib
import pandas as pd
from sklearn.model_selection import StratifiedShuffleSplit

class DataIngestion:

    def __init__(self,data_ingestion_config:DataIngestionConfig ):
        try:
            logging.info(f"{'>*>'*20}Data Ingestion log started.{ '<*<'*20} ")
            self.data_ingestion_config = data_ingestion_config
        except Exception as e:
            raise HousingException(e,sys)

    def download_housing_data(self,) -> str:
        try:
```

```
#extraction remote url to download dataset

download_url = self.data_ingestion_config.dataset_download_url


#folder location to download file

tgz_download_dir = self.data_ingestion_config.tgz_download_dir


os.makedirs(tgz_download_dir,exist_ok=True)


housing_file_name = os.path.basename(download_url)


tgz_file_path = os.path.join(tgz_download_dir, housing_file_name)

logging.info(f"Downloading file from :[{download_url}] into :[{tgz_file_path}]")

urllib.request.urlretrieve(download_url, tgz_file_path)

logging.info(f"File :[{tgz_file_path}] has been downloaded successfully.")

return tgz_file_path


except Exception as e:

    raise HousingException(e,sys) from e
```

```
def extract_tgz_file(self,tgz_file_path:str):

    try:

        raw_data_dir = self.data_ingestion_config.raw_data_dir

        if os.path.exists(raw_data_dir):
```

```
os.remove(raw_data_dir)

os.makedirs(raw_data_dir,exist_ok=True)

logging.info(f"Extracting tgz file: [{tgz_file_path}] into dir: [{raw_data_dir}]")
with tarfile.open(tgz_file_path) as housing_tgz_file_obj:
    housing_tgz_file_obj.extractall(path=raw_data_dir)
logging.info(f"Extraction completed")

except Exception as e:
    raise HousingException(e,sys) from e

def split_data_as_train_test(self) -> DataIngestionArtifact:
    try:
        raw_data_dir = self.data_ingestion_config.raw_data_dir

        file_name = os.listdir(raw_data_dir)[0]

        housing_file_path = os.path.join(raw_data_dir,file_name)

        logging.info(f"Reading csv file: [{housing_file_path}]")
        housing_data_frame = pd.read_csv(housing_file_path)

        housing_data_frame["income_cat"] = pd.cut(
```

```
    housing_data_frame["median_income"],  
    bins=[0.0, 1.5, 3.0, 4.5, 6.0, np.inf],  
    labels=[1,2,3,4,5]  
)  
  
logging.info(f"Splitting data into train and test")  
strat_train_set = None  
strat_test_set = None  
  
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)  
  
for train_index,test_index in split.split(housing_data_frame,  
housing_data_frame["income_cat"]):  
    strat_train_set = housing_data_frame.loc[train_index].drop(["income_cat"],axis=1)  
    strat_test_set = housing_data_frame.loc[test_index].drop(["income_cat"],axis=1)  
  
train_file_path = os.path.join(self.data_ingestion_config.ingested_train_dir,  
                               file_name)  
  
test_file_path = os.path.join(self.data_ingestion_config.ingested_test_dir,  
                               file_name)  
  
if strat_train_set is not None:  
    os.makedirs(self.data_ingestion_config.ingested_train_dir,exist_ok=True)  
    logging.info(f"Exporting training dataset to file: [{train_file_path}]")
```

```
strat_train_set.to_csv(train_file_path,index=False)

if strat_test_set is not None:

    os.makedirs(self.data_ingestion_config.ingested_test_dir, exist_ok= True)

    logging.info(f"Exporting test dataset to file: [{test_file_path}]")

    strat_test_set.to_csv(test_file_path,index=False)

data_ingestion_artifact = DataIngestionArtifact(train_file_path=train_file_path,
                                                test_file_path=test_file_path,
                                                is_ingested=True,
                                                message=f"Data ingestion completed successfully."
                                               )

logging.info(f"Data Ingestion artifact:[{data_ingestion_artifact}]")

return data_ingestion_artifact

except Exception as e:

    raise HousingException(e,sys) from e

def initiate_data_ingestion(self)-> DataIngestionArtifact:

    try:

        tgz_file_path = self.download_housing_data()

        self.extract_tgz_file(tgz_file_path=tgz_file_path)

        return self.split_data_as_train_test()

    except Exception as e:

        raise HousingException(e,sys)
```

```
raise HousingException(e,sys) from e
```

```
def __del__(self):
```

```
    logging.info(f"{'>>'*20}Data Ingestion log completed.{''<<'*20} \n\n")
```

machine_learning_project/housing/component/data_transformation.py

```
from cgi import test

from sklearn import preprocessing

from housing.exception import HousingException

from housing.logger import logging

from housing.entity.config_entity import DataTransformationConfig

from housing.entity.artifact_entity import DataIngestionArtifact,\n    DataValidationArtifact, DataTransformationArtifact

import sys,os

import numpy as np

from sklearn.base import BaseEstimator, TransformerMixin

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.pipeline import Pipeline

from sklearn.compose import ColumnTransformer

from sklearn.impute import SimpleImputer

import pandas as pd

from housing.constant import *

from housing.util.util import read_yaml_file, save_object, save_numpy_array_data, load_data

# longitude: float

# latitude: float

# housing_median_age: float

# total_rooms: float

# total_bedrooms: float
```

```
# population: float  
  
# households: float  
  
# median_income: float  
  
# median_house_value: float  
  
# ocean_proximity: category  
  
# income_cat: float
```

```
class FeatureGenerator(BaseEstimator, TransformerMixin):
```

```
    def __init__(self, add_bedrooms_per_room=True,  
                 total_rooms_ix=3,  
                 population_ix=5,  
                 households_ix=6,  
                 total_bedrooms_ix=4, columns=None):
```

```
        """
```

FeatureGenerator Initialization

add_bedrooms_per_room: bool

total_rooms_ix: int index number of total rooms columns

population_ix: int index number of total population columns

households_ix: int index number of households columns

total_bedrooms_ix: int index number of bedrooms columns

```
        """
```

try:

```
    self.columns = columns
```

```
if self.columns is not None:
```

```
    total_rooms_ix = self.columns.index(COLUMN_TOTAL_ROOMS)
```

```
    population_ix = self.columns.index(COLUMN_POPULATION)
```

```
    households_ix = self.columns.index(COLUMN_HOUSEHOLDS)
```

```
    total_bedrooms_ix = self.columns.index(COLUMN_TOTAL_BEDROOM)
```

```
    self.add_bedrooms_per_room = add_bedrooms_per_room
```

```
    self.total_rooms_ix = total_rooms_ix
```

```
    self.population_ix = population_ix
```

```
    self.households_ix = households_ix
```

```
    self.total_bedrooms_ix = total_bedrooms_ix
```

```
except Exception as e:
```

```
    raise HousingException(e, sys) from e
```

```
def fit(self, X, y=None):
```

```
    return self
```

```
def transform(self, X, y=None):
```

```
    try:
```

```
        room_per_household = X[:, self.total_rooms_ix] / \
```

```
        X[:, self.households_ix]
```

```
        population_per_household = X[:, self.population_ix] / \
```

```
        X[:, self.households_ix]
```

```
        if self.add_bedrooms_per_room:
```

```
            bedrooms_per_room = X[:, self.total_bedrooms_ix] / \
```

```
        X[:, self.total_rooms_ix]

    generated_feature = np.c_[
        X, room_per_household, population_per_household, bedrooms_per_room]

else:
    generated_feature = np.c_[
        X, room_per_household, population_per_household]

return generated_feature

except Exception as e:
    raise HousingException(e, sys) from e
```

class DataTransformation:

```
def __init__(self, data_transformation_config: DataTransformationConfig,
            data_ingestion_artifact: DataIngestionArtifact,
            data_validation_artifact: DataValidationArtifact
            ):
    try:
        logging.info(f">>' * 30}Data Transformation log started.'<<' * 30} ")
        self.data_transformation_config= data_transformation_config
        self.data_ingestion_artifact = data_ingestion_artifact
```

```
    self.data_validation_artifact = data_validation_artifact

except Exception as e:
    raise HousingException(e,sys) from e

def get_data_transformer_object(self)->ColumnTransformer:
    try:
        schema_file_path = self.data_validation_artifact.schema_file_path

        dataset_schema = read_yaml_file(file_path=schema_file_path)

        numerical_columns = dataset_schema[NUMBERICAL_COLUMN_KEY]
        categorical_columns = dataset_schema[CATEGORICAL_COLUMN_KEY]

        num_pipeline = Pipeline(steps=[

            ('imputer', SimpleImputer(strategy="median")),

            ('feature_generator', FeatureGenerator(
                add_bedrooms_per_room=self.data_transformation_config.add_bedroom_per_room,
                columns=numerical_columns
            )),

            ('scaler', StandardScaler())
        ])
    
```

```
)  
  
cat_pipeline = Pipeline(steps=[  
    ('impute', SimpleImputer(strategy="most_frequent")),  
    ('one_hot_encoder', OneHotEncoder()),  
    ('scaler', StandardScaler(with_mean=False))  
]  
)
```

```
logging.info(f"Categorical columns: {categorical_columns}")  
logging.info(f"Numerical columns: {numerical_columns}")
```

```
preprocessing = ColumnTransformer([  
    ('num_pipeline', num_pipeline, numerical_columns),  
    ('cat_pipeline', cat_pipeline, categorical_columns),  
])  
  
return preprocessing
```

```
except Exception as e:  
    raise HousingException(e,sys) from e
```

```
def initiate_data_transformation(self)->DataTransformationArtifact:  
    try:
```

```
logging.info(f"Obtaining preprocessing object.")

preprocessing_obj = self.get_data_transformer_object()

logging.info(f"Obtaining training and test file path.")

train_file_path = self.data_ingestion_artifact.train_file_path

test_file_path = self.data_ingestion_artifact.test_file_path

schema_file_path = self.data_validation_artifact.schema_file_path

logging.info(f"Loading training and test data as pandas dataframe.")

train_df = load_data(file_path=train_file_path, schema_file_path=schema_file_path)

test_df = load_data(file_path=test_file_path, schema_file_path=schema_file_path)

schema = read_yaml_file(file_path=schema_file_path)

target_column_name = schema[TARGET_COLUMN_KEY]

logging.info(f"Splitting input and target feature from training and testing dataframe.")

input_feature_train_df = train_df.drop(columns=[target_column_name], axis=1)

target_feature_train_df = train_df[target_column_name]
```

```
input_feature_test_df = test_df.drop(columns=[target_column_name],axis=1)

target_feature_test_df = test_df[target_column_name]

logging.info(f"Applying preprocessing object on training dataframe and testing
dataframe")

input_feature_train_arr=preprocessing_obj.fit_transform(input_feature_train_df)

input_feature_test_arr = preprocessing_obj.transform(input_feature_test_df)

train_arr = np.c_[ input_feature_train_arr, np.array(target_feature_train_df)]


test_arr = np.c_[input_feature_test_arr, np.array(target_feature_test_df)]


transformed_train_dir = self.data_transformation_config.transformed_train_dir

transformed_test_dir = self.data_transformation_config.transformed_test_dir


train_file_name = os.path.basename(train_file_path).replace(".csv",".npz")

test_file_name = os.path.basename(test_file_path).replace(".csv",".npz")


transformed_train_file_path = os.path.join(transformed_train_dir, train_file_name)

transformed_test_file_path = os.path.join(transformed_test_dir, test_file_name)


logging.info(f"Saving transformed training and testing array.")

save_numpy_array_data(file_path=transformed_train_file_path,array=train_arr)
```

```
    save_numpy_array_data(file_path=transformed_test_file_path,array=test_arr)

    preprocessing_obj_file_path =
self.data_transformation_config.preprocessed_object_file_path

    logging.info(f"Saving preprocessing object.")

    save_object(file_path=preprocessing_obj_file_path,obj=preprocessing_obj)

data_transformation_artifact = DataTransformationArtifact(is_transformed=True,
message="Data transformation successfull.",
transformed_train_file_path=transformed_train_file_path,
transformed_test_file_path=transformed_test_file_path,
preprocessed_object_file_path=preprocessing_obj_file_path

)

logging.info(f"Data transformationa artifact: {data_transformation_artifact}")

return data_transformation_artifact

except Exception as e:

    raise HousingException(e,sys) from e

def __del__(self):

    logging.info(f"{'>'*30}Data Transformation log completed.{ '<'*30} \n\n")
```

machine_learning_project/housing/component/data_validation.py

```
from housing.logger import logging
from housing.exception import HousingException
from housing.entity.config_entity import DataValidationConfig
from housing.entity.artifact_entity import DataIngestionArtifact, DataValidationArtifact
import os,sys
import pandas as pd
from evidently.model_profile import Profile
from evidently.model_profile.sections import DataDriftProfileSection
from evidently.dashboard import Dashboard
from evidently.dashboard.tabs import DataDriftTab
import json

class DataValidation:

    def __init__(self, data_validation_config:DataValidationConfig,
                 data_ingestion_artifact:DataIngestionArtifact):
        try:
            logging.info(f">{'*'30}Data Valdaition log started.{<'*30} \n\n")
            self.data_validation_config = data_validation_config
            self.data_ingestion_artifact = data_ingestion_artifact
        except Exception as e:
            raise HousingException(e,sys) from e

    def get_train_and_test_df(self):
        try:
            train_df = pd.read_csv(self.data_ingestion_artifact.train_file_path)
```

```
test_df = pd.read_csv(self.data_ingestion_artifact.test_file_path)
return train_df,test_df

except Exception as e:
    raise HousingException(e,sys) from e

def is_train_test_file_exists(self)->bool:
    try:
        logging.info("Checking if training and test file is available")
        is_train_file_exist = False
        is_test_file_exist = False

        train_file_path = self.data_ingestion_artifact.train_file_path
        test_file_path = self.data_ingestion_artifact.test_file_path

        is_train_file_exist = os.path.exists(train_file_path)
        is_test_file_exist = os.path.exists(test_file_path)

        is_available = is_train_file_exist and is_test_file_exist

        logging.info(f"Is train and test file exists?-> {is_available}")

        if not is_available:
            training_file = self.data_ingestion_artifact.train_file_path
            testing_file = self.data_ingestion_artifact.test_file_path
            message=f"Training file: {training_file} or Testing file: {testing_file}" \
                    "is not present"
            raise Exception(message)

        return is_available

    except Exception as e:
```

```
raise HousingException(e,sys) from e

def validate_dataset_schema(self)->bool:
    try:
        validation_status = False

        #Assignment validate training and testing dataset using schema file

        #1. Number of Column

        #2. Check the value of ocean proximity

        # acceptable values <1H OCEAN

        # INLAND

        # ISLAND

        # NEAR BAY

        # NEAR OCEAN

        #3. Check column names

        validation_status = True

    return validation_status

except Exception as e:
    raise HousingException(e,sys) from e

def get_and_save_data_drift_report(self):
    try:
        profile = Profile(sections=[DataDriftProfileSection()])

        train_df,test_df = self.get_train_and_test_df()

        profile.calculate(train_df,test_df)

    
```

```
report = json.loads(profile.json())

report_file_path = self.data_validation_config.report_file_path
report_dir = os.path.dirname(report_file_path)
os.makedirs(report_dir, exist_ok=True)

with open(report_file_path, "w") as report_file:
    json.dump(report, report_file, indent=6)
return report

except Exception as e:
    raise HousingException(e, sys) from e
```

```
def save_data_drift_report_page(self):
    try:
        dashboard = Dashboard(tabs=[DataDriftTab()])
        train_df, test_df = self.get_train_and_test_df()
        dashboard.calculate(train_df, test_df)

        report_page_file_path = self.data_validation_config.report_page_file_path
        report_page_dir = os.path.dirname(report_page_file_path)
        os.makedirs(report_page_dir, exist_ok=True)

        dashboard.save(report_page_file_path)

    except Exception as e:
        raise HousingException(e, sys) from e
```

```
def is_data_drift_found(self) -> bool:
    try:
        report = self.get_and_save_data_drift_report()
        self.save_data_drift_report_page()
        return True
    except Exception as e:
        raise HousingException(e, sys) from e
```

```
except Exception as e:  
    raise HousingException(e,sys) from e  
  
def initiate_data_validation(self)->DataValidationArtifact :  
    try:  
        self.is_train_test_file_exists()  
        self.validate_dataset_schema()  
        self.is_data_drift_found()  
  
        data_validation_artifact = DataValidationArtifact(  
            schema_file_path=self.data_validation_config.schema_file_path,  
            report_file_path=self.data_validation_config.report_file_path,  
            report_page_file_path=self.data_validation_config.report_page_file_path,  
            is_validated=True,  
            message="Data Validation performed successfully."  
        )  
        logging.info(f"Data validation artifact: {data_validation_artifact}")  
        return data_validation_artifact  
    except Exception as e:  
        raise HousingException(e,sys) from e  
  
def __del__(self):  
    logging.info(f">{'>'*30}Data Valdaiton log completed.{'<'*30} \n\n")
```

machine_learning_project/housing/component/model_evaluation.py

```
from housing.logger import logging
from housing.exception import HousingException
from housing.entity.config_entity import ModelEvaluationConfig
from housing.entity.artifact_entity import
DataIngestionArtifact, DataValidationArtifact, ModelTrainerArtifact, ModelEvaluationArtifact
from housing.constant import *
import numpy as np
import os
import sys
from housing.util.util import write_yaml_file, read_yaml_file, load_object, load_data
from housing.entity.model_factory import evaluate_regression_model
```

```
class ModelEvaluation:
```

```
    def __init__(self, model_evaluation_config: ModelEvaluationConfig,
                 data_ingestion_artifact: DataIngestionArtifact,
                 data_validation_artifact: DataValidationArtifact,
                 model_trainer_artifact: ModelTrainerArtifact):
        try:
            logging.info(f">>> * 30)Model Evaluation log started.<<< * 30")
            self.model_evaluation_config = model_evaluation_config
            self.model_trainer_artifact = model_trainer_artifact
            self.data_ingestion_artifact = data_ingestion_artifact
            self.data_validation_artifact = data_validation_artifact
        except Exception as e:
            raise HousingException(e, sys) from e
```

```
def get_best_model(self):  
    try:  
        model = None  
  
        model_evaluation_file_path = self.model_evaluation_config.model_evaluation_file_path  
  
        if not os.path.exists(model_evaluation_file_path):  
            write_yaml_file(file_path=model_evaluation_file_path,  
                           )  
        return model  
  
        model_eval_file_content = read_yaml_file(file_path=model_evaluation_file_path)  
  
        model_eval_file_content = dict() if model_eval_file_content is None else  
        model_eval_file_content  
  
        if BEST_MODEL_KEY not in model_eval_file_content:  
            return model  
  
        model =  
        load_object(file_path=model_eval_file_content[BEST_MODEL_KEY][MODEL_PATH_KEY])  
        return model  
  
    except Exception as e:  
        raise HousingException(e, sys) from e  
  
def update_evaluation_report(self, model_evaluation_artifact: ModelEvaluationArtifact):  
    try:  
        eval_file_path = self.model_evaluation_config.model_evaluation_file_path  
        model_eval_content = read_yaml_file(file_path=eval_file_path)  
        model_eval_content = dict() if model_eval_content is None else model_eval_content  
  
        previous_best_model = None
```

```
if BEST_MODEL_KEY in model_eval_content:
    previous_best_model = model_eval_content[BEST_MODEL_KEY]

logging.info(f"Previous eval result: {model_eval_content}")

eval_result = {
    BEST_MODEL_KEY: {
        MODEL_PATH_KEY: model_evaluation_artifact.evaluated_model_path,
    }
}

if previous_best_model is not None:
    model_history = {self.model_evaluation_config.time_stamp: previous_best_model}
    if HISTORY_KEY not in model_eval_content:
        history = {HISTORY_KEY: model_history}
        eval_result.update(history)
    else:
        model_eval_content[HISTORY_KEY].update(model_history)

model_eval_content.update(eval_result)
logging.info(f"Updated eval result:{model_eval_content}")
write_yaml_file(file_path=eval_file_path, data=model_eval_content)

except Exception as e:
    raise HousingException(e, sys) from e

def initiate_model_evaluation(self) -> ModelEvaluationArtifact:
    try:
        trained_model_file_path = self.model_trainer_artifact.trained_model_file_path
        trained_model_object = load_object(file_path=trained_model_file_path)

        train_file_path = self.data_ingestion_artifact.train_file_path
```

```
test_file_path = self.data_ingestion_artifact.test_file_path

schema_file_path = self.data_validation_artifact.schema_file_path

train_dataframe = load_data(file_path=train_file_path,
                           schema_file_path=schema_file_path,
                           )

test_dataframe = load_data(file_path=test_file_path,
                           schema_file_path=schema_file_path,
                           )

schema_content = read_yaml_file(file_path=schema_file_path)
target_column_name = schema_content[TARGET_COLUMN_KEY]

# target_column
logging.info(f"Converting target column into numpy array.")

train_target_arr = np.array(train_dataframe[target_column_name])
test_target_arr = np.array(test_dataframe[target_column_name])
logging.info(f"Conversion completed target column into numpy array.")

# dropping target column from the dataframe
logging.info(f"Dropping target column from the dataframe.")

train_dataframe.drop(target_column_name, axis=1, inplace=True)
test_dataframe.drop(target_column_name, axis=1, inplace=True)
logging.info(f"Dropping target column from the dataframe completed.")

model = self.get_best_model()

if model is None:
    logging.info("Not found any existing model. Hence accepting trained model")
    model_evaluation_artifact =
ModelEvaluationArtifact(evaluated_model_path=trained_model_file_path,
```

```
        is_model_accepted=True)

    self.update_evaluation_report(model_evaluation_artifact)

    logging.info(f"Model accepted. Model eval artifact {model_evaluation_artifact} created")

    return model_evaluation_artifact

model_list = [model, trained_model_object]

metric_info_artifact = evaluate_regression_model(model_list=model_list,
                                                 X_train=train_dataframe,
                                                 y_train=train_target_arr,
                                                 X_test=test_dataframe,
                                                 y_test=test_target_arr,
                                                 base_accuracy=self.model_trainer_artifact.model_accuracy,
                                                 )

logging.info(f"Model evaluation completed. model metric artifact: {metric_info_artifact}")

if metric_info_artifact is None:

    response = ModelEvaluationArtifact(is_model_accepted=False,
                                       evaluated_model_path=trained_model_file_path
                                       )

    logging.info(response)

    return response

if metric_info_artifact.index_number == 1:

    model_evaluation_artifact =
    ModelEvaluationArtifact(evaluated_model_path=trained_model_file_path,
                           is_model_accepted=True)

    self.update_evaluation_report(model_evaluation_artifact)

    logging.info(f"Model accepted. Model eval artifact {model_evaluation_artifact} created")

else:
```

```
    logging.info("Trained model is no better than existing model hence not accepting trained
model")

    model_evaluation_artifact =
ModelEvaluationArtifact(evaluated_model_path=trained_model_file_path,
                        is_model_accepted=False)

    return model_evaluation_artifact

except Exception as e:
    raise HousingException(e, sys) from e

def __del__(self):
    logging.info(f"{'=' * 20}Model Evaluation log completed.{=' * 20}")
```

machine_learning_project/housing/component/model_pusher.py

```
from housing.logger import logging
from housing.exception import HousingException
from housing.entity.artifact_entity import ModelPusherArtifact, ModelEvaluationArtifact
from housing.entity.config_entity import ModelPusherConfig
import os, sys
import shutil

class ModelPusher:

    def __init__(self, model_pusher_config: ModelPusherConfig,
                 model_evaluation_artifact: ModelEvaluationArtifact
                 ):
        try:
            logging.info(f'{">>' * 30}Model Pusher log started.{ '<<' * 30} ')
            self.model_pusher_config = model_pusher_config
            self.model_evaluation_artifact = model_evaluation_artifact

        except Exception as e:
            raise HousingException(e, sys) from e

    def export_model(self) -> ModelPusherArtifact:
        try:
            evaluated_model_file_path = self.model_evaluation_artifact.evaluated_model_path
            export_dir = self.model_pusher_config.export_dir_path
            model_file_name = os.path.basename(evaluated_model_file_path)
            export_model_file_path = os.path.join(export_dir, model_file_name)
            logging.info(f"Exporting model file: [{export_model_file_path}]")
            os.makedirs(export_dir, exist_ok=True)
```

```
shutil.copy(src=evaluated_model_file_path, dst=export_model_file_path)

#we can call a function to save model to Azure blob storage/ google cloud strorage / s3 bucket
logging.info(
    f"Trained model: {evaluated_model_file_path} is copied in export
dir:[{export_model_file_path}]")

model_pusher_artifact = ModelPusherArtifact(is_model_pusher=True,
                                             export_model_file_path=export_model_file_path
                                         )
logging.info(f"Model pusher artifact: [{model_pusher_artifact}]")
return model_pusher_artifact

except Exception as e:
    raise HousingException(e, sys) from e

def initiate_model_pusher(self) -> ModelPusherArtifact:
    try:
        return self.export_model()
    except Exception as e:
        raise HousingException(e, sys) from e

def __del__(self):
    logging.info(f">{'*' * 20}Model Pusher log completed.{'*' * 20}")
```

machine_learning_project/housing/component/model_trainer.py

```
from housing.exception import HousingException
import sys
from housing.logger import logging
from typing import List
from housing.entity.artifact_entity import DataTransformationArtifact, ModelTrainerArtifact
from housing.entity.config_entity import ModelTrainerConfig
from housing.util.util import load_numpy_array_data,save_object,load_object
from housing.entity.model_factory import MetricInfoArtifact, ModelFactory,GridSearchedBestModel
from housing.entity.model_factory import evaluate_regression_model

class HousingEstimatorModel:
    def __init__(self, preprocessing_object, trained_model_object):
        """
        TrainedModel constructor
        preprocessing_object: preprocessing_object
        trained_model_object: trained_model_object
        """
        self.preprocessing_object = preprocessing_object
        self.trained_model_object = trained_model_object

    def predict(self, X):
        """
        function accepts raw inputs and then transformed raw input using preprocessing_object
        which guarantees that the inputs are in the same format as the training data
        At last it perform prediction on transformed features
        """

```

```
transformed_feature = self.preprocessing_object.transform(X)
return self.trained_model_object.predict(transformed_feature)

def __repr__(self):
    return f"{type(self.trained_model_object).__name__}()"

def __str__(self):
    return f"{type(self.trained_model_object).__name__}()"

class ModelTrainer:

    def __init__(self, model_trainer_config:ModelTrainerConfig, data_transformation_artifact:
DataTransformationArtifact):
        try:
            logging.info(f">* 30}Model trainer log started.{*< 30} ")
            self.model_trainer_config = model_trainer_config
            self.data_transformation_artifact = data_transformation_artifact
        except Exception as e:
            raise HousingException(e, sys) from e

    def initiate_model_trainer(self)->ModelTrainerArtifact:
        try:
            logging.info(f"Loading transformed training dataset")
            transformed_train_file_path = self.data_transformation_artifact.transformed_train_file_path
            train_array = load_numpy_array_data(file_path=transformed_train_file_path)

            logging.info(f"Loading transformed testing dataset")
            transformed_test_file_path = self.data_transformation_artifact.transformed_test_file_path
```

```
test_array = load_numpy_array_data(file_path=transformed_test_file_path)

logging.info(f"Splitting training and testing input and target feature")
x_train,y_train,x_test,y_test = train_array[:, :-1],train_array[:, -1],test_array[:, :-1],test_array[:, -1]

logging.info(f"Extracting model config file path")
model_config_file_path = self.model_trainer_config.model_config_file_path

logging.info(f"Initializing model factory class using above model config file: {model_config_file_path}")
model_factory = ModelFactory(model_config_path=model_config_file_path)

base_accuracy = self.model_trainer_config.base_accuracy
logging.info(f"Expected accuracy: {base_accuracy}")

logging.info(f"Initiating operation model selecttion")
best_model =
model_factory.get_best_model(X=x_train,y=y_train,base_accuracy=base_accuracy)

logging.info(f"Best model found on training dataset: {best_model}")

logging.info(f"Extracting trained model list.")

grid_searched_best_model_list:List[GridSearchedBestModel]=model_factory.grid_searched_best_model_list

model_list = [model.best_model for model in grid_searched_best_model_list ]
logging.info(f"Evaluation all trained model on training and testing dataset both")
```

```
    metric_info:MetricInfoArtifact =
evaluate_regression_model(model_list=model_list,X_train=x_train,y_train=y_train,X_test=x_test,y_t
est=y_test,base_accuracy=base_accuracy)

    logging.info(f"Best found model on both training and testing dataset.")

    preprocessing_obj=
load_object(file_path=self.data_transformation_artifact.preprocessed_object_file_path)

    model_object = metric_info.model_object

trained_model_file_path=self.model_trainer_config.trained_model_file_path

    housing_model =
HousingEstimatorModel(preprocessing_object=preprocessing_obj,trained_model_object=model_obj
ect)

    logging.info(f"Saving model at path: {trained_model_file_path}")

    save_object(file_path=trained_model_file_path,obj=housing_model)

model_trainer_artifact= ModelTrainerArtifact(is_trained=True,message="Model Trained
successfully",

    trained_model_file_path=trained_model_file_path,
    train_rmse=metric_info.train_rmse,
    test_rmse=metric_info.test_rmse,
    train_accuracy=metric_info.train_accuracy,
    test_accuracy=metric_info.test_accuracy,
    model_accuracy=metric_info.model_accuracy

)

logging.info(f"Model Trainer Artifact: {model_trainer_artifact}")

return model_trainer_artifact

except Exception as e:
```

```
raise HousingException(e, sys) from e

def __del__(self):
    logging.info(f'{">>' * 30}Model trainer log completed.{ '<<' * 30} ')

#loading transformed training and testing datset
#reading model config file
#getting best model on training datset
#evaludation models on both training & testing datset -->model object
#loading preprocessing pbjct
#custom model object by combining both preprocessing obj and model obj
#saving custom model object
#return model_trainer_artifact
```

machine_learning_project/housing/config/configuration.py

```
from housing.entity.config_entity import DataIngestionConfig,
DataTransformationConfig, DataValidationConfig, \
ModelTrainerConfig, ModelEvaluationConfig, ModelPusherConfig, TrainingPipelineConfig
from housing.util.util import read_yaml_file
from housing.logger import logging
import sys,os
from housing.constant import *
from housing.exception import HousingException

class Configuartion:

    def __init__(self,
                 config_file_path: str = CONFIG_FILE_PATH,
                 current_time_stamp: str = CURRENT_TIME_STAMP
                ) -> None:
        try:
            self.config_info = read_yaml_file(file_path=config_file_path)
            self.training_pipeline_config = self.get_training_pipeline_config()
            self.time_stamp = current_time_stamp
        except Exception as e:
            raise HousingException(e,sys) from e

    def get_data_ingestion_config(self) -> DataIngestionConfig:
        try:
            artifact_dir = self.training_pipeline_config.artifact_dir
            data_ingestion_artifact_dir = os.path.join(
```

```
        artifact_dir,  
        DATA_INGESTION_ARTIFACT_DIR,  
        self.time_stamp  
    )  
  
    data_ingestion_info = self.config_info[DATA_INGESTION_CONFIG_KEY]  
  
    dataset_download_url = data_ingestion_info[DATA_INGESTION_DOWNLOAD_URL_KEY]  
    tgz_download_dir = os.path.join(  
        data_ingestion_artifact_dir,  
        data_ingestion_info[DATA_INGESTION_TGZ_DOWNLOAD_DIR_KEY]  
    )  
    raw_data_dir = os.path.join(data_ingestion_artifact_dir,  
        data_ingestion_info[DATA_INGESTION_RAW_DATA_DIR_KEY]  
    )  
  
    ingested_data_dir = os.path.join(  
        data_ingestion_artifact_dir,  
        data_ingestion_info[DATA_INGESTION_INGESTED_DIR_NAME_KEY]  
    )  
    ingested_train_dir = os.path.join(  
        ingested_data_dir,  
        data_ingestion_info[DATA_INGESTION_TRAIN_DIR_KEY]  
    )  
    ingested_test_dir = os.path.join(  
        ingested_data_dir,  
        data_ingestion_info[DATA_INGESTION_TEST_DIR_KEY]  
    )  
  
data_ingestion_config=DataIngestionConfig(  
    dataset_download_url=dataset_download_url,
```

```
    tgz_download_dir=tgz_download_dir,
    raw_data_dir=raw_data_dir,
    ingested_train_dir=ingested_train_dir,
    ingested_test_dir=ingested_test_dir
)
logging.info(f"Data Ingestion config: {data_ingestion_config}")
return data_ingestion_config
except Exception as e:
    raise HousingException(e,sys) from e

def get_data_validation_config(self) -> DataValidationConfig:
    try:
        artifact_dir = self.training_pipeline_config.artifact_dir

        data_validation_artifact_dir=os.path.join(
            artifact_dir,
            DATA_VALIDATION_ARTIFACT_DIR_NAME,
            self.time_stamp
        )
        data_validation_config = self.config_info[DATA_VALIDATION_CONFIG_KEY]

        schema_file_path = os.path.join(ROOT_DIR,
            data_validation_config[DATA_VALIDATION_SCHEMA_DIR_KEY],
            data_validation_config[DATA_VALIDATION_SCHEMA_FILE_NAME_KEY]
        )

        report_file_path = os.path.join(data_validation_artifact_dir,
            data_validation_config[DATA_VALIDATION_REPORT_FILE_NAME_KEY]
        )
    
```

```
    report_page_file_path = os.path.join(data_validation_artifact_dir,
                                         data_validation_config[DATA_VALIDATION_REPORT_PAGE_FILE_NAME_KEY])

)

data_validation_config = DataValidationConfig(
    schema_file_path=schema_file_path,
    report_file_path=report_file_path,
    report_page_file_path=report_page_file_path,
)
return data_validation_config

except Exception as e:
    raise HousingException(e,sys) from e

def get_data_transformation_config(self) -> DataTransformationConfig:
    try:
        artifact_dir = self.training_pipeline_config.artifact_dir

        data_transformation_artifact_dir=os.path.join(
            artifact_dir,
            DATA_TRANSFORMATION_ARTIFACT_DIR,
            self.time_stamp
        )

        data_transformation_config_info=self.config_info[DATA_TRANSFORMATION_CONFIG_KEY]

        add_bedroom_per_room=data_transformation_config_info[DATA_TRANSFORMATION_ADD_BEDROOM_PER_ROOM_KEY]

        preprocessed_object_file_path = os.path.join(
```

```
        data_transformation_artifact_dir,
        data_transformation_config_info[DATA_TRANSFORMATION_PREPROCESSING_DIR_KEY],
        data_transformation_config_info[DATA_TRANSFORMATION_PREPROCESSED_FILE_NAME_KEY]
    )

transformed_train_dir=os.path.join(
    data_transformation_artifact_dir,
    data_transformation_config_info[DATA_TRANSFORMATION_DIR_NAME_KEY],
    data_transformation_config_info[DATA_TRANSFORMATION_TRAIN_DIR_NAME_KEY]
)

transformed_test_dir = os.path.join(
    data_transformation_artifact_dir,
    data_transformation_config_info[DATA_TRANSFORMATION_DIR_NAME_KEY],
    data_transformation_config_info[DATA_TRANSFORMATION_TEST_DIR_NAME_KEY]
)

data_transformation_config=DataTransformationConfig(
    add_bedroom_per_room=add_bedroom_per_room,
    preprocessed_object_file_path=preprocessed_object_file_path,
    transformed_train_dir=transformed_train_dir,
    transformed_test_dir=transformed_test_dir
)

logging.info(f"Data transformation config: {data_transformation_config}")

return data_transformation_config
```

```
except Exception as e:  
    raise HousingException(e,sys) from e  
  
def get_model_trainer_config(self) -> ModelTrainerConfig:  
    try:  
        artifact_dir = self.training_pipeline_config.artifact_dir  
  
        model_trainer_artifact_dir=os.path.join(  
            artifact_dir,  
            MODEL_TRAINER_ARTIFACT_DIR,  
            self.time_stamp  
        )  
        model_trainer_config_info = self.config_info[MODEL_TRAINER_CONFIG_KEY]  
        trained_model_file_path = os.path.join(model_trainer_artifact_dir,  
            model_trainer_config_info[MODEL_TRAINER_TRAINED_MODEL_DIR_KEY],  
            model_trainer_config_info[MODEL_TRAINER_TRAINED_MODEL_FILE_NAME_KEY]  
        )  
  
        model_config_file_path =  
os.path.join(model_trainer_config_info[MODEL_TRAINER_MODEL_CONFIG_DIR_KEY],  
            model_trainer_config_info[MODEL_TRAINER_MODEL_CONFIG_FILE_NAME_KEY]  
        )  
  
        base_accuracy = model_trainer_config_info[MODEL_TRAINER_BASE_ACCURACY_KEY]  
  
        model_trainer_config = ModelTrainerConfig(  
            trained_model_file_path=trained_model_file_path,  
            base_accuracy=base_accuracy,  
            model_config_file_path=model_config_file_path  
        )  
        logging.info(f"Model trainer config: {model_trainer_config}")
```

```

        return model_trainer_config

    except Exception as e:
        raise HousingException(e,sys) from e

def get_model_evaluation_config(self) ->ModelEvaluationConfig:
    try:
        model_evaluation_config = self.config_info[MODEL_EVALUATION_CONFIG_KEY]
        artifact_dir = os.path.join(self.training_pipeline_config.artifact_dir,
                                    MODEL_EVALUATION_ARTIFACT_DIR, )

        model_evaluation_file_path = os.path.join(artifact_dir,
                                                model_evaluation_config[MODEL_EVALUATION_FILE_NAME_KEY])
        response = ModelEvaluationConfig(model_evaluation_file_path=model_evaluation_file_path,
                                         time_stamp=self.time_stamp)

        logging.info(f"Model Evaluation Config: {response}.")
        return response

    except Exception as e:
        raise HousingException(e,sys) from e

def get_model_pusher_config(self) -> ModelPusherConfig:
    try:
        time_stamp = f"{datetime.now().strftime('%Y%m%d%H%M%S')}"
        model_pusher_config_info = self.config_info[MODEL_PUSHER_CONFIG_KEY]
        export_dir_path = os.path.join(ROOT_DIR,
                                      model_pusher_config_info[MODEL_PUSHER_MODEL_EXPORT_DIR_KEY],
                                      time_stamp)

        model_pusher_config = ModelPusherConfig(export_dir_path=export_dir_path)

```

```
logging.info(f"Model pusher config {model_pusher_config}")

return model_pusher_config

except Exception as e:
    raise HousingException(e,sys) from e

def get_training_pipeline_config(self) ->TrainingPipelineConfig:
    try:
        training_pipeline_config = self.config_info[TRAINING_PIPELINE_CONFIG_KEY]
        artifact_dir = os.path.join(ROOT_DIR,
        training_pipeline_config[TRAINING_PIPELINE_NAME_KEY],
        training_pipeline_config[TRAINING_PIPELINE_ARTIFACT_DIR_KEY]
    )

        training_pipeline_config = TrainingPipelineConfig(artifact_dir=artifact_dir)
        logging.info(f"Training pipeline config: {training_pipeline_config}")
        return training_pipeline_config
    except Exception as e:
        raise HousingException(e,sys) from e
```

```
machine_learning_project/housing/constant/__init__.py
```

```
import os
from datetime import datetime

def get_current_time_stamp():
    return f"{'datetime.now().strftime('%Y-%m-%d-%H-%M-%S')}'"

ROOT_DIR = os.getcwd() #to get current working directory
CONFIG_DIR = "config"
CONFIG_FILE_NAME = "config.yaml"
CONFIG_FILE_PATH = os.path.join(ROOT_DIR,CONFIG_DIR,CONFIG_FILE_NAME)

CURRENT_TIME_STAMP = get_current_time_stamp()

# Training pipeline related variable
TRAINING_PIPELINE_CONFIG_KEY = "training_pipeline_config"
TRAINING_PIPELINE_ARTIFACT_DIR_KEY = "artifact_dir"
TRAINING_PIPELINE_NAME_KEY = "pipeline_name"

# Data Ingestion related variable
```

```
DATA_INGESTION_CONFIG_KEY = "data_ingestion_config"
DATA_INGESTION_ARTIFACT_DIR = "data_ingestion"
DATA_INGESTION_DOWNLOAD_URL_KEY = "dataset_download_url"
DATA_INGESTION_RAW_DATA_DIR_KEY = "raw_data_dir"
DATA_INGESTION_TGZ_DOWNLOAD_DIR_KEY = "tgz_download_dir"
DATA_INGESTION_INGESTED_DIR_NAME_KEY = "ingested_dir"
DATA_INGESTION_TRAIN_DIR_KEY = "ingested_train_dir"
DATA_INGESTION_TEST_DIR_KEY = "ingested_test_dir"
```

```
# Data Validation related variable
```

```
# Data Validation related variables
DATA_VALIDATION_CONFIG_KEY = "data_validation_config"
DATA_VALIDATION_SCHEMA_FILE_NAME_KEY = "schema_file_name"
DATA_VALIDATION_SCHEMA_DIR_KEY = "schema_dir"
DATA_VALIDATION_ARTIFACT_DIR_NAME="data_validation"
DATA_VALIDATION_REPORT_FILE_NAME_KEY = "report_file_name"
DATA_VALIDATION_REPORT_PAGE_FILE_NAME_KEY = "report_page_file_name"
```

```
# Data Transformation related variables
```

```
DATA_TRANSFORMATION_ARTIFACT_DIR = "data_transformation"
DATA_TRANSFORMATION_CONFIG_KEY = "data_transformation_config"
DATA_TRANSFORMATION_ADD_BEDROOM_PER_ROOM_KEY = "add_bedroom_per_room"
DATA_TRANSFORMATION_DIR_NAME_KEY = "transformed_dir"
DATA_TRANSFORMATION_TRAIN_DIR_NAME_KEY = "transformed_train_dir"
DATA_TRANSFORMATION_TEST_DIR_NAME_KEY = "transformed_test_dir"
DATA_TRANSFORMATION_PREPROCESSING_DIR_KEY = "preprocessing_dir"
DATA_TRANSFORMATION_PREPROCESSED_FILE_NAME_KEY = "preprocessed_object_file_name"
```

```
COLUMN_TOTAL_ROOMS = "total_rooms"
COLUMN_POPULATION = "population"
COLUMN_HOUSEHOLDS = "households"
COLUMN_TOTAL_BEDROOM = "total_bedrooms"
DATASET_SCHEMA_COLUMNS_KEY= "columns"

NUMERICAL_COLUMN_KEY="numerical_columns"
CATEGORICAL_COLUMN_KEY = "categorical_columns"

TARGET_COLUMN_KEY="target_column"

# Model Training related variables

MODEL_TRAINER_ARTIFACT_DIR = "model_trainer"
MODEL_TRAINER_CONFIG_KEY = "model_trainer_config"
MODEL_TRAINER_TRAINED_MODEL_DIR_KEY = "trained_model_dir"
MODEL_TRAINER_TRAINED_MODEL_FILE_NAME_KEY = "model_file_name"
MODEL_TRAINER_BASE_ACCURACY_KEY = "base_accuracy"
MODEL_TRAINER_MODEL_CONFIG_DIR_KEY = "model_config_dir"
MODEL_TRAINER_MODEL_CONFIG_FILE_NAME_KEY = "model_config_file_name"

MODEL_EVALUATION_CONFIG_KEY = "model_evaluation_config"
MODEL_EVALUATION_FILE_NAME_KEY = "model_evaluation_file_name"
MODEL_EVALUATION_ARTIFACT_DIR = "model_evaluation"

# Model Pusher config key
```

```
MODEL_PUSHER_CONFIG_KEY = "model_pusher_config"
MODEL_PUSHER_MODEL_EXPORT_DIR_KEY = "model_export_dir"

BEST_MODEL_KEY = "best_model"
HISTORY_KEY = "history"
MODEL_PATH_KEY = "model_path"

EXPERIMENT_DIR_NAME="experiment"
EXPERIMENT_FILE_NAME="experiment.csv"
```

```
machine_learning_project/housing/entity/artifact_entity.py
```

```
from collections import namedtuple
```

```
DataIngestionArtifact = namedtuple("DataIngestionArtifact",  
[ "train_file_path", "test_file_path", "is_ingested", "message"])
```

```
DataValidationArtifact = namedtuple("DataValidationArtifact",  
["schema_file_path","report_file_path","report_page_file_path","is_validated","message"]))
```

```
DataTransformationArtifact = namedtuple("DataTransformationArtifact",  
["is_transformed", "message", "transformed_train_file_path","transformed_test_file_path",  
"preprocessed_object_file_path"])
```

```
ModelTrainerArtifact = namedtuple("ModelTrainerArtifact", ["is_trained", "message",  
"trained_model_file_path",  
"train_rmse", "test_rmse", "train_accuracy", "test_accuracy",  
"model_accuracy"])
```

```
ModelEvaluationArtifact = namedtuple("ModelEvaluationArtifact", ["is_model_accepted",  
"evaluated_model_path"])
```

```
ModelPusherArtifact = namedtuple("ModelPusherArtifact", ["is_model_pusher",  
"export_model_file_path"])
```

```
machine_learning_project/housing/entity/config_entity.py

from collections import namedtuple

DataIngestionConfig=namedtuple("DataIngestionConfig",
["dataset_download_url","tgz_download_dir","raw_data_dir","ingested_train_dir","ingested_test_di
r"])

DataValidationConfig = namedtuple("DataValidationConfig",
["schema_file_path","report_file_path","report_page_file_path"])

DataTransformationConfig = namedtuple("DataTransformationConfig", ["add_bedroom_per_room",
                    "transformed_train_dir",
                    "transformed_test_dir",
                    "preprocessed_object_file_path"])

ModelTrainerConfig = namedtuple("ModelTrainerConfig",
["trained_model_file_path","base_accuracy","model_config_file_path"])

ModelEvaluationConfig = namedtuple("ModelEvaluationConfig",
["model_evaluation_file_path","time_stamp"])

ModelPusherConfig = namedtuple("ModelPusherConfig", ["export_dir_path"])

TrainingPipelineConfig = namedtuple("TrainingPipelineConfig", ["artifact_dir"])
```

`machine_learning_project/housing/entity/experiment.py`

```
class Experiment:  
    running_status=False  
  
    def __new__(cls,*args,**kwargs):  
        if Experiment.running_status:  
            raise Exception("Experiment is already running hence new experiment can not be created")  
        return super(Experiment,cls).__new__(cls,*args,**kwargs)  
  
    def __init__(self,experiment_id):  
        self.experiment_id = experiment_id  
        self.running_status = Experiment.running_status
```

```
machine_learning_project/housing/entity/housing_predictor.py
```

```
import os
```

```
import sys
```

```
from housing.exception import HousingException
```

```
from housing.util.util import load_object
```

```
import pandas as pd
```

```
class HousingData:
```

```
    def __init__(self,
```

```
        longitude: float,
```

```
        latitude: float,
```

```
        housing_median_age: float,
```

```
        total_rooms: float,
```

```
        total_bedrooms: float,
```

```
        population: float,
```

```
        households: float,
```

```
        median_income: float,
```

```
        ocean_proximity: str,
```

```
        median_house_value: float = None
```

```
):
```

```
    try:
```

```
        self.longitude = longitude
```

```
        self.latitude = latitude
```

```
        self.housing_median_age = housing_median_age
```

```
        self.total_rooms = total_rooms
```

```
        self.total_bedrooms = total_bedrooms
```

```
    self.population = population
    self.households = households
    self.median_income = median_income
    self.ocean_proximity = ocean_proximity
    self.median_house_value = median_house_value
except Exception as e:
    raise HousingException(e, sys) from e

def get_housing_input_data_frame(self):
    try:
        housing_input_dict = self.get_housing_data_as_dict()
        return pd.DataFrame(housing_input_dict)
    except Exception as e:
        raise HousingException(e, sys) from e

def get_housing_data_as_dict(self):
    try:
        input_data = {
            "longitude": [self.longitude],
            "latitude": [self.latitude],
            "housing_median_age": [self.housing_median_age],
            "total_rooms": [self.total_rooms],
            "total_bedrooms": [self.total_bedrooms],
            "population": [self.population],
            "households": [self.households],
            "median_income": [self.median_income],
            "ocean_proximity": [self.ocean_proximity]}
        return input_data
    except Exception as e:
        raise HousingException(e, sys)
```

```
class HousingPredictor:

    def __init__(self, model_dir: str):
        try:
            self.model_dir = model_dir
        except Exception as e:
            raise HousingException(e, sys) from e

    def get_latest_model_path(self):
        try:
            folder_name = list(map(int, os.listdir(self.model_dir)))
            latest_model_dir = os.path.join(self.model_dir, f"{max(folder_name)}")
            file_name = os.listdir(latest_model_dir)[0]
            latest_model_path = os.path.join(latest_model_dir, file_name)
        return latest_model_path

        except Exception as e:
            raise HousingException(e, sys) from e

    def predict(self, X):
        try:
            model_path = self.get_latest_model_path()
            model = load_object(file_path=model_path)
            median_house_value = model.predict(X)
        return median_house_value

        except Exception as e:
            raise HousingException(e, sys) from e
```



```
BestModel = namedtuple("BestModel", ["model_serial_number",
                                     "model",
                                     "best_model",
                                     "best_parameters",
                                     "best_score", ]) 

MetricInfoArtifact = namedtuple("MetricInfoArtifact",
                                ["model_name", "model_object", "train_rmse", "test_rmse", "train_accuracy",
                                 "test_accuracy", "model_accuracy", "index_number"])
```

```
def evaluate_classification_model(model_list: list, X_train:np.ndarray, y_train:np.ndarray,
X_test:np.ndarray, y_test:np.ndarray, base_accuracy:float=0.6)->MetricInfoArtifact:
    pass
```

```
def evaluate_regression_model(model_list: list, X_train:np.ndarray, y_train:np.ndarray,
X_test:np.ndarray, y_test:np.ndarray, base_accuracy:float=0.6) -> MetricInfoArtifact:
```

.....

Description:

This function compare multiple regression model return best model

Params:

model_list: List of model

X_train: Training dataset input feature

y_train: Training dataset target feature

X_test: Testing dataset input feature

y_test: Testing dataset input feature

return

It retured a named tuple

```
MetricInfoArtifact = namedtuple("MetricInfo",
    ["model_name", "model_object", "train_rmse", "test_rmse", "train_accuracy",
     "test_accuracy", "model_accuracy", "index_number"])

.....
try:
    index_number = 0
    metric_info_artifact = None
    for model in model_list:
        model_name = str(model) #getting model name based on model object
        logging.info(f'{">>'*30}Started evaluating model: [{type(model).__name__}] {'<<'*30}')

        #Getting prediction for training and testing dataset
        y_train_pred = model.predict(X_train)
        y_test_pred = model.predict(X_test)

        #Calculating r squared score on training and testing dataset
        train_acc = r2_score(y_train, y_train_pred)
        test_acc = r2_score(y_test, y_test_pred)

        #Calculating mean squared error on training and testing dataset
        train_rmse = np.sqrt(mean_squared_error(y_train, y_train_pred))
        test_rmse = np.sqrt(mean_squared_error(y_test, y_test_pred))

        # Calculating harmonic mean of train_accuracy and test_accuracy
        model_accuracy = (2 * (train_acc * test_acc)) / (train_acc + test_acc)
        diff_test_train_acc = abs(test_acc - train_acc)
```

```
#logging all important metric

logging.info(f"{'>'*30} Score {'<'*30}")

logging.info(f"Train Score\t\t Test Score\t\t Average Score")

logging.info(f"{train_acc}\t\t {test_acc}\t\t{model_accuracy}")

logging.info(f"{'>'*30} Loss {'<'*30}")

logging.info(f"Diff test train accuracy: [{diff_test_train_acc}].")

logging.info(f"Train root mean squared error: [{train_rmse}].")

logging.info(f"Test root mean squared error: [{test_rmse}].")
```

```
#if model accuracy is greater than base accuracy and train and test score is within certain
threshold
```

```
#we will accept that model as accepted model

if model_accuracy >= base_accuracy and diff_test_train_acc < 0.05:

    base_accuracy = model_accuracy

    metric_info_artifact = MetricInfoArtifact(model_name=model_name,
                                                model_object=model,
                                                train_rmse=train_rmse,
                                                test_rmse=test_rmse,
                                                train_accuracy=train_acc,
                                                test_accuracy=test_acc,
                                                model_accuracy=model_accuracy,
                                                index_number=index_number)
```

```
logging.info(f"Acceptable model found {metric_info_artifact}. ")

index_number += 1

if metric_info_artifact is None:

    logging.info(f"No model found with higher accuracy than base accuracy")

    return metric_info_artifact

except Exception as e:
```

```
raise HousingException(e, sys) from e

def get_sample_model_config_yaml_file(export_dir: str):
    try:
        model_config = {
            GRID_SEARCH_KEY: {
                MODULE_KEY: "sklearn.model_selection",
                CLASS_KEY: "GridSearchCV",
                PARAM_KEY: {
                    "cv": 3,
                    "verbose": 1
                }
            },
            MODEL_SELECTION_KEY: {
                "module_0": {
                    MODULE_KEY: "module_of_model",
                    CLASS_KEY: "ModelClassName",
                    PARAM_KEY:
                        {"param_name1": "value1",
                         "param_name2": "value2",
                         },
                    SEARCH_PARAM_GRID_KEY: {
                        "param_name": ['param_value_1', 'param_value_2']
                    }
                },
                }
            }
        os.makedirs(export_dir, exist_ok=True)
```

```

export_file_path = os.path.join(export_dir, "model.yaml")
with open(export_file_path, 'w') as file:
    yaml.dump(model_config, file)
return export_file_path

except Exception as e:
    raise HousingException(e, sys)

class ModelFactory:

    def __init__(self, model_config_path: str = None,):
        try:
            self.config: dict = ModelFactory.read_params(model_config_path)

            self.grid_search_cv_module: str = self.config[GRID_SEARCH_KEY][MODULE_KEY]
            self.grid_search_class_name: str = self.config[GRID_SEARCH_KEY][CLASS_KEY]
            self.grid_search_property_data: dict = dict(self.config[GRID_SEARCH_KEY][PARAM_KEY])

            self.models_initialization_config: dict = dict(self.config[MODEL_SELECTION_KEY])

            self.initialized_model_list = None
            self.grid_searched_best_model_list = None

        except Exception as e:
            raise HousingException(e, sys) from e

    @staticmethod
    def update_property_of_class(instance_ref:object, property_data: dict):
        try:
            if not isinstance(property_data, dict):
                raise Exception("property_data parameter required to dictionary")
            print(property_data)

```

```

        for key, value in property_data.items():
            logging.info(f"Executing:$ {str(instance_ref)}.{key}={value}")
            setattr(instance_ref, key, value)
        return instance_ref

    except Exception as e:
        raise HousingException(e, sys) from e

    @staticmethod
    def read_params(config_path: str) -> dict:
        try:
            with open(config_path) as yaml_file:
                config:dict = yaml.safe_load(yaml_file)
        return config

    except Exception as e:
        raise HousingException(e, sys) from e

    @staticmethod
    def class_for_name(module_name:str, class_name:str):
        try:
            # load the module, will raise ImportError if module cannot be loaded
            module = importlib.import_module(module_name)

            # get the class, will raise AttributeError if class cannot be found
            logging.info(f"Executing command: from {module} import {class_name}")
            class_ref = getattr(module, class_name)

        return class_ref

    except Exception as e:
        raise HousingException(e, sys) from e

    def execute_grid_search_operation(self, initialized_model: InitializedModelDetail, input_feature,
                                     output_feature) -> GridSearchedBestModel:
        .....

```

excute_grid_search_operation(): function will perform paramter search operation and it will return you the best optimistic model with best paramter:
estimator: Model object
param_grid: dictionary of paramter to perform search operation
input_feature: your all input features
output_feature: Target/Dependent features

=====

return: Function will return GridSearchOperation object

====

try:

instantiating GridSearchCV class

```
grid_search_cv_ref =  
ModelFactory.class_for_name(module_name=self.grid_search_cv_module,  
                            class_name=self.grid_search_class_name  
                            )
```

```
grid_search_cv = grid_search_cv_ref(estimator=initialized_model.model,  
                                    param_grid=initialized_model.param_grid_search)  
grid_search_cv = ModelFactory.update_property_of_class(grid_search_cv,  
                                                       self.grid_search_property_data)
```

```
message = f'{">>* 30} f"Training {type(initialized_model.model).__name__} Started."  
{"<**30}'  
logging.info(message)  
grid_search_cv.fit(input_feature, output_feature)  
message = f'{">>* 30} f"Training {type(initialized_model.model).__name__}" completed  
{"<**30}'  
grid_searched_best_model =  
GridSearchedBestModel(model_serial_number=initialized_model.model_serial_number,
```

```

        model=initialized_model.model,
        best_model=grid_search_cv.best_estimator_,
        best_parameters=grid_search_cv.best_params_
        best_score=grid_search_cv.best_score_
    )

return grid_searched_best_model
except Exception as e:
    raise HousingException(e, sys) from e

```

```

def get_initialized_model_list(self) -> List[InitializedModelDetail]:
    """
    This function will return a list of model details.

    return List[ModelDetail]
    """

try:
    initialized_model_list = []
    for model_serial_number in self.models_INITIALIZATION_CONFIG.keys():

        model_INITIALIZATION_CONFIG = self.models_INITIALIZATION_CONFIG[model_serial_number]
        model_obj_ref =
            ModelFactory.class_for_name(module_name=model_INITIALIZATION_CONFIG[MODULE_KEY],
                                         class_name=model_INITIALIZATION_CONFIG[CLASS_KEY])
    )
    model = model_obj_ref()

if PARAM_KEY in model_INITIALIZATION_CONFIG:
    model_obj_property_data = dict(model_INITIALIZATION_CONFIG[PARAM_KEY])
    model = ModelFactory.update_property_of_class(instance_ref=model,
                                                   property_data=model_obj_property_data)

```

```

param_grid_search = model_initialization_config[SEARCH_PARAM_GRID_KEY]

model_name =
f"{model_initialization_config[MODULE_KEY]}.{model_initialization_config[CLASS_KEY]}"

model_initialization_config =
InitializedModelDetail(model_serial_number=model_serial_number,
                      model=model,
                      param_grid_search=param_grid_search,
                      model_name=model_name
                     )

initialized_model_list.append(model_initialization_config)

self.initialized_model_list = initialized_model_list

return self.initialized_model_list

except Exception as e:
    raise HousingException(e, sys) from e

def initiate_best_parameter_search_for_initialized_model(self, initialized_model:
InitializedModelDetail,
                           input_feature,
                           output_feature) -> GridSearchedBestModel:
    """
    initiate_best_model_parameter_search(): function will perform paramter search operation and
    it will return you the best optimistic model with best paramter:
    estimator: Model object
    param_grid: dictionary of paramter to perform search operation
    input_feature: your all input features
    output_feature: Target/Dependent features
    =====
    return: Function will return a GridSearchOperation

```

```

"""
try:
    return self.execute_grid_search_operation(initialized_model=initialized_model,
                                              input_feature=input_feature,
                                              output_feature=output_feature)

except Exception as e:
    raise HousingException(e, sys) from e

def initiate_best_parameter_search_for_initialized_models(self,
                                                          initialized_model_list: List[InitializedModelDetail],
                                                          input_feature,
                                                          output_feature) -> List[GridSearchedBestModel]:
    try:
        self.grid_searched_best_model_list = []
        for initialized_model_list in initialized_model_list:
            grid_searched_best_model = self.initiate_best_parameter_search_for_initialized_model(
                initialized_model=initialized_model_list,
                input_feature=input_feature,
                output_feature=output_feature
            )
            self.grid_searched_best_model_list.append(grid_searched_best_model)
        return self.grid_searched_best_model_list
    except Exception as e:
        raise HousingException(e, sys) from e

    @staticmethod
    def get_model_detail(model_details: List[InitializedModelDetail],
                         model_serial_number: str) -> InitializedModelDetail:
        """
This function return ModelDetail

```

```
"""

try:
    for model_data in model_details:
        if model_data.model_serial_number == model_serial_number:
            return model_data
except Exception as e:
    raise HousingException(e, sys) from e

@staticmethod
def get_best_model_from_grid_searched_best_model_list(grid_searched_best_model_list: List[GridSearchedBestModel],
                                                       base_accuracy=0.6
                                                       ) -> BestModel:
    try:
        best_model = None
        for grid_searched_best_model in grid_searched_best_model_list:
            if base_accuracy < grid_searched_best_model.best_score:
                logging.info(f"Acceptable model found:{grid_searched_best_model}")
                base_accuracy = grid_searched_best_model.best_score

        best_model = grid_searched_best_model
        if not best_model:
            raise Exception(f"None of Model has base accuracy: {base_accuracy}")
        logging.info(f"Best model: {best_model}")
        return best_model
    except Exception as e:
        raise HousingException(e, sys) from e

def get_best_model(self, X, y, base_accuracy=0.6) -> BestModel:
    try:
        logging.info("Started Initializing model from config file")
```

```
initialized_model_list = self.get_initialized_model_list()
logging.info(f"Initialized model: {initialized_model_list}")
grid_searched_best_model_list = self.initiate_best_parameter_search_for_initialized_models(
    initialized_model_list=initialized_model_list,
    input_feature=X,
    output_feature=y
)
return
ModelFactory.get_best_model_from_grid_searched_best_model_list(grid_searched_best_model_list,
    base_accuracy=base_accuracy)
except Exception as e:
    raise HousingException(e, sys)
```

```
machine_learning_project/housing/exception/__init__.py

import os

import sys


class HousingException(Exception):

    def __init__(self, error_message:Exception,error_detail:sys):
        super().__init__(error_message)

        self.error_message=HousingException.get_detailed_error_message(error_message=error_message,
                                                                    error_detail=error_detail
                                                                    )



    @staticmethod
    def get_detailed_error_message(error_message:Exception,error_detail:sys)->str:
        """
        error_message: Exception object
        error_detail: object of sys module
        """

        _,exec_tb = error_detail.exc_info()

        exception_block_line_number = exec_tb.tb_frame.f_lineno
        try_block_line_number = exec_tb.tb_lineno
        file_name = exec_tb.tb_frame.f_code.co_filename
        error_message = f"""

Error occured in script:

[ {file_name} ] at

try block line number: [{try_block_line_number}] and exception block line number:
[{exception_block_line_number}]

error message: [{error_message}]

"""

        return error_message
```

```
def __str__(self):  
    return self.error_message  
  
def __repr__(self) -> str:  
    return HousingException.__name__.str()
```

```
machine_learning_project/housing/logger/__init__.py

import logging
from datetime import datetime
import os
import pandas as pd
from housing.constant import get_current_time_stamp
LOG_DIR="logs"

def get_log_file_name():
    return f"log_{get_current_time_stamp()}.log"

LOG_FILE_NAME=get_log_file_name()

os.makedirs(LOG_DIR,exist_ok=True)

LOG_FILE_PATH = os.path.join(LOG_DIR,LOG_FILE_NAME)

logging.basicConfig(filename=LOG_FILE_PATH,
filemode="w",
format='[%(asctime)s]^(%(levelname)s^(%(lineno)d^(%(filename)s^(%(funcName)s()^(%(message)s',
level=logging.INFO
)

def get_log_dataframe(file_path):
    data=[]
    with open(file_path) as log_file:
        for line in log_file.readlines():
            data.append(line.split("^(;"))


```

```
log_df = pd.DataFrame(data)

columns=["Time stamp","Log Level","line number","file name","function name","message"]

log_df.columns=columns

log_df["log_message"] = log_df['Time stamp'].astype(str) +":$"+ log_df["message"]

return log_df[["log_message"]]
```

machine_learning_project/housing/pipeline/pipeline.py

```
from collections import namedtuple
from datetime import datetime
import uuid

from housing.config.configuration import Configuartion
from housing.logger import logging, get_log_file_name
from housing.exception import HousingException
from threading import Thread
from typing import List

from multiprocessing import Process
from housing.entity.artifact_entity import ModelPusherArtifact, DataIngestionArtifact,
ModelEvaluationArtifact
from housing.entity.artifact_entity import DataValidationArtifact, DataTransformationArtifact,
ModelTrainerArtifact
from housing.entity.config_entity import DataIngestionConfig, ModelEvaluationConfig
from housing.component.data_ingestion import DataIngestion
from housing.component.data_validation import DataValidation
from housing.component.data_transformation import DataTransformation
from housing.component.model_trainer import ModelTrainer
from housing.component.model_evaluation import ModelEvaluation
from housing.component.model_pusher import ModelPusher
import os, sys
from collections import namedtuple
from datetime import datetime
import pandas as pd
from housing.constant import EXPERIMENT_DIR_NAME, EXPERIMENT_FILE_NAME

Experiment = namedtuple("Experiment", ["experiment_id", "initialization_timestamp",
"artifact_time_stamp",
"running_status", "start_time", "stop_time", "execution_time", "message",
"experiment_file_path", "accuracy", "is_model_accepted"])

```

```
class Pipeline(Thread):
    experiment: Experiment = Experiment(*([None] * 11))
    experiment_file_path = None

    def __init__(self, config: Configuartion ) -> None:
        try:
            os.makedirs(config.training_pipeline_config.artifact_dir, exist_ok=True)

            Pipeline.experiment_file_path=os.path.join(config.training_pipeline_config.artifact_dir,EXPERIMENT
            _DIR_NAME, EXPERIMENT_FILE_NAME)

            super().__init__(daemon=False, name="pipeline")
            self.config = config
        except Exception as e:
            raise HousingException(e, sys) from e

    def start_data_ingestion(self) -> DataIngestionArtifact:
        try:
            data_ingestion = DataIngestion(data_ingestion_config=self.config.get_data_ingestion_config())
            return data_ingestion.initiate_data_ingestion()
        except Exception as e:
            raise HousingException(e, sys) from e

    def start_data_validation(self, data_ingestion_artifact: DataIngestionArtifact) \
        -> DataValidationArtifact:
        try:
            data_validation =
DataValidation(data_validation_config=self.config.get_data_validation_config(),
```

```

        data_ingestion_artifact=data_ingestion_artifact
    )

    return data_validation.initiate_data_validation()

except Exception as e:
    raise HousingException(e, sys) from e


def start_data_transformation(self,
                               data_ingestion_artifact: DataIngestionArtifact,
                               data_validation_artifact: DataValidationArtifact
                               ) -> DataTransformationArtifact:

    try:
        data_transformation = DataTransformation(
            data_transformation_config=self.config.get_data_transformation_config(),
            data_ingestion_artifact=data_ingestion_artifact,
            data_validation_artifact=data_validation_artifact
        )
        return data_transformation.initiate_data_transformation()

    except Exception as e:
        raise HousingException(e, sys)

def start_model_trainer(self, data_transformation_artifact: DataTransformationArtifact) ->
ModelTrainerArtifact:
    try:
        model_trainer = ModelTrainer(model_trainer_config=self.config.get_model_trainer_config(),
                                     data_transformation_artifact=data_transformation_artifact
                                     )
        return model_trainer.initiate_model_trainer()

    except Exception as e:
        raise HousingException(e, sys)

def start_model_evaluation(self, data_ingestion_artifact: DataIngestionArtifact,

```

```

        data_validation_artifact: DataValidationArtifact,
        model_trainer_artifact: ModelTrainerArtifact) -> ModelEvaluationArtifact:

    try:
        model_eval = ModelEvaluation(
            model_evaluation_config=self.config.get_model_evaluation_config(),
            data_ingestion_artifact=data_ingestion_artifact,
            data_validation_artifact=data_validation_artifact,
            model_trainer_artifact=model_trainer_artifact)
        return model_eval.initiate_model_evaluation()
    except Exception as e:
        raise HousingException(e, sys) from e

    def start_model_pusher(self, model_eval_artifact: ModelEvaluationArtifact) ->
    ModelPusherArtifact:
        try:
            model_pusher = ModelPusher(
                model_pusher_config=self.config.get_model_pusher_config(),
                model_evaluation_artifact=model_eval_artifact
            )
            return model_pusher.initiate_model_pusher()
        except Exception as e:
            raise HousingException(e, sys) from e

    def run_pipeline(self):
        try:
            if Pipeline.experiment.running_status:
                logging.info("Pipeline is already running")
                return Pipeline.experiment
            # data ingestion
            logging.info("Pipeline starting.")

```

```
experiment_id = str(uuid.uuid4())

Pipeline.experiment = Experiment(experiment_id=experiment_id,
                                initialization_timestamp=self.config.time_stamp,
                                artifact_time_stamp=self.config.time_stamp,
                                running_status=True,
                                start_time=datetime.now(),
                                stop_time=None,
                                execution_time=None,
                                experiment_file_path=Pipeline.experiment_file_path,
                                is_model_accepted=None,
                                message="Pipeline has been started.",
                                accuracy=None,
                                )

logging.info(f"Pipeline experiment: {Pipeline.experiment}")

self.save_experiment()

data_ingestion_artifact = self.start_data_ingestion()
data_validation_artifact =
self.start_data_validation(data_ingestion_artifact=data_ingestion_artifact)

data_transformation_artifact = self.start_data_transformation(
    data_ingestion_artifact=data_ingestion_artifact,
    data_validation_artifact=data_validation_artifact
)

model_trainer_artifact =
self.start_model_trainer(data_transformation_artifact=data_transformation_artifact)

model_evaluation_artifact =
self.start_model_evaluation(data_ingestion_artifact=data_ingestion_artifact,
                           data_validation_artifact=data_validation_artifact,
                           model_trainer_artifact=model_trainer_artifact)
```

```

if model_evaluation_artifact.is_model_accepted:

    model_pusher_artifact =
self.start_model_pusher(model_eval_artifact=model_evaluation_artifact)

    logging.info(f'Model pusher artifact: {model_pusher_artifact}')

else:

    logging.info("Trained model rejected.")

logging.info("Pipeline completed.")

stop_time = datetime.now()

Pipeline.experiment = Experiment(experiment_id=Pipeline.experiment.experiment_id,
                                  initialization_timestamp=self.config.time_stamp,
                                  artifact_time_stamp=self.config.time_stamp,
                                  running_status=False,
                                  start_time=Pipeline.experiment.start_time,
                                  stop_time=stop_time,
                                  execution_time=stop_time - Pipeline.experiment.start_time,
                                  message="Pipeline has been completed.",
                                  experiment_file_path=Pipeline.experiment_file_path,
                                  is_model_accepted=model_evaluation_artifact.is_model_accepted,
                                  accuracy=model_trainer_artifact.model_accuracy
)

logging.info(f'Pipeline experiment: {Pipeline.experiment}')

self.save_experiment()

except Exception as e:

    raise HousingException(e, sys) from e

def run(self):

    try:

        self.run_pipeline()

    except Exception as e:

```

```

        raise e

def save_experiment(self):
    try:
        if Pipeline.experiment.experiment_id is not None:
            experiment = Pipeline.experiment
            experiment_dict = experiment._asdict()
            experiment_dict: dict = {key: [value] for key, value in experiment_dict.items()}

            experiment_dict.update({
                "created_time_stamp": [datetime.now()],
                "experiment_file_path": [
                    os.path.basename(Pipeline.experiment.experiment_file_path)]})

        experiment_report = pd.DataFrame(experiment_dict)

        os.makedirs(os.path.dirname(Pipeline.experiment_file_path), exist_ok=True)
        if os.path.exists(Pipeline.experiment_file_path):
            experiment_report.to_csv(Pipeline.experiment_file_path, index=False, header=False,
                                     mode="a")
        else:
            experiment_report.to_csv(Pipeline.experiment_file_path, mode="w", index=False,
                                     header=True)
        else:
            print("First start experiment")
    except Exception as e:
        raise HousingException(e, sys) from e

@classmethod
def get_experiments_status(cls, limit: int = 5) -> pd.DataFrame:
    try:
        if os.path.exists(Pipeline.experiment_file_path):

```

```
df = pd.read_csv(Pipeline.experiment_file_path)

limit = -1 * int(limit)

return df[limit:].drop(columns=["experiment_file_path", "initialization_timestamp"], axis=1)

else:

    return pd.DataFrame()

except Exception as e:

    raise HousingException(e, sys) from e
```

machine_learning_project/housing/util/util.py

```
import yaml
from housing.exception import HousingException
import os,sys
import numpy as np
import dill
import pandas as pd
from housing.constant import *

def write_yaml_file(file_path:str,data:dict=None):
    """
    Create yaml file
    file_path: str
    data: dict
    """
    try:
        os.makedirs(os.path.dirname(file_path), exist_ok=True)
        with open(file_path,"w") as yaml_file:
            if data is not None:
                yaml.dump(data,yaml_file)
    except Exception as e:
        raise HousingException(e,sys)
```

def read_yaml_file(file_path:str)->dict:

""""

Reads a YAML file and returns the contents as a dictionary.

```
file_path: str
"""

```

```
try:  
    with open(file_path, 'rb') as yaml_file:  
        return yaml.safe_load(yaml_file)  
    except Exception as e:  
        raise HousingException(e,sys) from e
```

```
def save_numpy_array_data(file_path: str, array: np.array):
```

```
    """
```

```
    Save numpy array data to file
```

```
    file_path: str location of file to save
```

```
    array: np.array data to save
```

```
    """
```

```
try:
```

```
    dir_path = os.path.dirname(file_path)
```

```
    os.makedirs(dir_path, exist_ok=True)
```

```
    with open(file_path, 'wb') as file_obj:
```

```
        np.save(file_obj, array)
```

```
except Exception as e:
```

```
    raise HousingException(e, sys) from e
```

```
def load_numpy_array_data(file_path: str) -> np.array:
```

```
    """
```

```
    load numpy array data from file
```

```
    file_path: str location of file to load
```

```
    return: np.array data loaded
```

```
    """
```

```
try:
```

```
    with open(file_path, 'rb') as file_obj:
```

```
        return np.load(file_obj)
```

```
except Exception as e:  
    raise HousingException(e, sys) from e
```

```
def save_object(file_path:str,obj):
    """
    file_path: str
    obj: Any sort of object
    """

    try:
        dir_path = os.path.dirname(file_path)
        os.makedirs(dir_path, exist_ok=True)
        with open(file_path, "wb") as file_obj:
            dill.dump(obj, file_obj)

    except Exception as e:
        raise HousingException(e,sys) from e
```

```
def load_object(file_path:str):
    """
    file_path: str
    """
    try:
        with open(file_path, "rb") as file_obj:
            return dill.load(file_obj)
    except Exception as e:
        raise HousingException(e,sys) from e
```

```
def load_data(file_path: str, schema_file_path: str) -> pd.DataFrame:  
  
    try:
```

```
dataset_schema = read_yaml_file(schema_file_path)

schema = dataset_schema[DATASET_SCHEMA_COLUMNS_KEY]

dataframe = pd.read_csv(file_path)

error_message = ""

for column in dataframe.columns:
    if column in list(schema.keys()):
        dataframe[column].astype(schema[column])
    else:
        error_message += f"\nColumn: [{column}] is not in the schema."
if len(error_message) > 0:
    raise Exception(error_message)

return dataframe

except Exception as e:
    raise HousingException(e,sys) from e
```

```
machine_learning_project/app.py

from flask import Flask, request

import sys

import pip

from housing.util.util import read_yaml_file, write_yaml_file

from matplotlib.style import context

from housing.logger import logging

from housing.exception import HousingException

import os, sys

import json

from housing.config.configuration import Configuration

from housing.constant import CONFIG_DIR, get_current_time_stamp

from housing.pipeline.pipeline import Pipeline

from housing.entity.housing_predictor import HousingPredictor, HousingData

from flask import send_file, abort, render_template

ROOT_DIR = os.getcwd()

LOG_FOLDER_NAME = "logs"

PIPELINE_FOLDER_NAME = "housing"

SAVED_MODELS_DIR_NAME = "saved_models"

MODEL_CONFIG_FILE_PATH = os.path.join(ROOT_DIR, CONFIG_DIR, "model.yaml")

LOG_DIR = os.path.join(ROOT_DIR, LOG_FOLDER_NAME)

PIPELINE_DIR = os.path.join(ROOT_DIR, PIPELINE_FOLDER_NAME)

MODEL_DIR = os.path.join(ROOT_DIR, SAVED_MODELS_DIR_NAME)

from housing.logger import get_log_dataframe

HOUSING_DATA_KEY = "housing_data"
```

```
MEDIAN_HOUSING_VALUE_KEY = "median_house_value"

app = Flask(__name__)

@app.route('/artifact', defaults={'req_path': 'housing'})
@app.route('/artifact/<path:req_path>')
def render_artifact_dir(req_path):
    os.makedirs("housing", exist_ok=True)
    # Joining the base and the requested path
    print(f"req_path: {req_path}")
    abs_path = os.path.join(req_path)
    print(abs_path)
    # Return 404 if path doesn't exist
    if not os.path.exists(abs_path):
        return abort(404)

    # Check if path is a file and serve
    if os.path.isfile(abs_path):
        if ".html" in abs_path:
            with open(abs_path, "r", encoding="utf-8") as file:
                content = ""
                for line in file.readlines():
                    content = f"{content}{line}"
            return content
        return send_file(abs_path)

    # Show directory contents
    files = {os.path.join(abs_path, file_name): file_name for file_name in os.listdir(abs_path) if
             "artifact" in os.path.join(abs_path, file_name)}
```

```

result = {
    "files": files,
    "parent_folder": os.path.dirname(abs_path),
    "parent_label": abs_path
}

return render_template('files.html', result=result)

@app.route('/', methods=['GET', 'POST'])

def index():
    try:
        return render_template('index.html')
    except Exception as e:
        return str(e)

@app.route('/view_experiment_hist', methods=['GET', 'POST'])

def view_experiment_history():
    experiment_df = Pipeline.get_experiments_status()
    context = {
        "experiment": experiment_df.to_html(classes='table table-striped col-12')
    }
    return render_template('experiment_history.html', context=context)

@app.route('/train', methods=['GET', 'POST'])

def train():
    message = """
    pipeline = Pipeline(config=Configuartion(current_time_stamp=get_current_time_stamp()))
    if not Pipeline.experiment.running_status:
        message = "Training started."
    """

```

```

pipeline.start()

else:
    message = "Training is already in progress."
    context = {
        "experiment": pipeline.get_experiments_status().to_html(classes='table table-striped col-12'),
        "message": message
    }
    return render_template('train.html', context=context)

@app.route('/predict', methods=['GET', 'POST'])

def predict():
    context = {
        HOUSING_DATA_KEY: None,
        MEDIAN_HOUSING_VALUE_KEY: None
    }

    if request.method == 'POST':
        longitude = float(request.form['longitude'])
        latitude = float(request.form['latitude'])
        housing_median_age = float(request.form['housing_median_age'])
        total_rooms = float(request.form['total_rooms'])
        total_bedrooms = float(request.form['total_bedrooms'])
        population = float(request.form['population'])
        households = float(request.form['households'])
        median_income = float(request.form['median_income'])
        ocean_proximity = request.form['ocean_proximity']

        housing_data = HousingData(longitude=longitude,
                                   latitude=latitude,
                                   housing_median_age=housing_median_age,

```

```

        total_rooms=total_rooms,
        total_bedrooms=total_bedrooms,
        population=population,
        households=households,
        median_income=median_income,
        ocean_proximity=ocean_proximity,
    )

housing_df = housing_data.get_housing_input_data_frame()
housing_predictor = HousingPredictor(model_dir=MODEL_DIR)
median_housing_value = housing_predictor.predict(X=housing_df)
context = {
    HOUSING_DATA_KEY: housing_data.get_housing_data_as_dict(),
    MEDIAN_HOUSING_VALUE_KEY: median_housing_value,
}
return render_template('predict.html', context=context)
return render_template("predict.html", context=context)

```

```

@app.route('/saved_models', defaults={'req_path': 'saved_models'})
@app.route('/saved_models/<path:req_path>')
def saved_models_dir(req_path):
    os.makedirs("saved_models", exist_ok=True)
    # Joining the base and the requested path
    print(f"req_path: {req_path}")
    abs_path = os.path.join(req_path)
    print(abs_path)
    # Return 404 if path doesn't exist
    if not os.path.exists(abs_path):
        return abort(404)

    # Check if path is a file and serve

```

```
if os.path.isfile(abs_path):
    return send_file(abs_path)

# Show directory contents
files = {os.path.join(abs_path, file): file for file in os.listdir(abs_path)}

result = {
    "files": files,
    "parent_folder": os.path.dirname(abs_path),
    "parent_label": abs_path
}
return render_template('saved_models_files.html', result=result)
```

```
@app.route("/update_model_config", methods=['GET', 'POST'])

def update_model_config():
    try:
        if request.method == 'POST':
            model_config = request.form['new_model_config']
            model_config = model_config.replace("''", "''")
            print(model_config)
            model_config = json.loads(model_config)

            write_yaml_file(file_path=MODEL_CONFIG_FILE_PATH, data=model_config)

        model_config = read_yaml_file(file_path=MODEL_CONFIG_FILE_PATH)
        return render_template('update_model.html', result={"model_config": model_config})
    except Exception as e:
        logging.exception(e)
        return str(e)
```

```
@app.route(f'/logs', defaults={'req_path': f'{LOG_FOLDER_NAME}'})  
@app.route(f'/{LOG_FOLDER_NAME}/<path:req_path>')  
  
def render_log_dir(req_path):  
    os.makedirs(LOG_FOLDER_NAME, exist_ok=True)  
  
    # Joining the base and the requested path  
    logging.info(f"req_path: {req_path}")  
    abs_path = os.path.join(req_path)  
    print(abs_path)  
  
    # Return 404 if path doesn't exist  
    if not os.path.exists(abs_path):  
        return abort(404)  
  
  
    # Check if path is a file and serve  
    if os.path.isfile(abs_path):  
        log_df = get_log_dataframe(abs_path)  
        context = {"log": log_df.to_html(classes="table-striped", index=False)}  
        return render_template('log.html', context=context)  
  
  
    # Show directory contents  
    files = {os.path.join(abs_path, file): file for file in os.listdir(abs_path)}  
  
  
    result = {  
        "files": files,  
        "parent_folder": os.path.dirname(abs_path),  
        "parent_label": abs_path  
    }  
  
    return render_template('log_files.html', result=result)
```

```
if __name__ == "__main__":
    app.run()
```

machine_learning_project/demo.py

```
from housing.pipeline.pipeline import Pipeline
from housing.exception import HousingException
from housing.logger import logging
from housing.config.configuration import Configuartion
from housing.component.data_transformation import DataTransformation
import os

def main():

    try:
        config_path = os.path.join("config", "config.yaml")
        pipeline = Pipeline(Configuartion(config_file_path=config_path))
        #pipeline.run_pipeline()
        pipeline.start()
        logging.info("main function execution completed.")
        # # data_validation_config = Configuartion().get_data_transformation_config()
        # # print(data_validation_config)
        # schema_file_path=r"D:\Project\machine_learning_project\config\schema.yaml"
        # file_path=r"D:\Project\machine_learning_project\housing\artifact\data_ingestion\2022-06-27-19-13-17\ingested_data\train\housing.csv"

        # df= DataTransformation.load_data(file_path=file_path,schema_file_path=schema_file_path)
        # print(df.columns)
        # print(df.dtypes)

    except Exception as e:
        logging.error(f"{e}")
        print(e)
```

```
if __name__=="__main__":
    main()
```

In [3]:

```
1 from housing.entity.model_factory import ModelFactory, get_sample_model_config_yaml_f
```

In [5]:

```
1 os.getcwd()
```

Out[5]:

```
'd:\\Project\\machine_learning_project\\notebook'
```

In [4]:

```
1 get_sample_model_config_yaml_file(export_dir="config")
```

Out[4]:

```
'config\\model.yaml'
```

In [6]:

```
1 from sklearn.linear_model import LinearRegression
```

In []:

```
1 LinearRegression()
```

In [1]:

```
1 model_config_file=r"D:\\Project\\machine_learning_project\\notebook\\config\\model.yaml"
```

In [2]:

```
1 from neuro_mf import ModelFactory
```

In [3]:

```
1 model_factory = ModelFactory(model_config_path=model_config_file)
```

In [9]:

```
1 model_list = model_factory.get_initialized_model_list()
```

```
{'fit_intercept': True}  
{'n_estimators': 40, 'min_samples_leaf': 2}
```

In [11]:

```
1 len(model_list)
```

Out[11]:

2

In [13]:

```
1 model_list[1]
```

Out[13]:

```
InitializedModelDetail(model_serial_number='module_1', model=RandomForestRegressor(min_samples_leaf=2, n_estimators=40), param_grid_search={'min_samples_leaf': [2, 4, 6], 'n_estimators': [50, 100, 80]}, model_name='sklearn.ensemble.RandomForestRegressor')
```

In [4]:

```
1 from housing.util.util import load_numpy_array_data
```

In [5]:

```
1 data_file_path=r"D:\Project\machine_learning_project\housing\artifact\data_transform
```



In [6]:

```
1 data = load_numpy_array_data(data_file_path)
```

In [7]:

```
1 x,y = data[:, :-1], data[:, -1]
```

In [9]:

```
1 model_factory.grid_searched_best_model_list
```

Out[9]:

```
[GridSearchedBestModel(model_serial_number='module_0', model=LinearRegression(), best_model=LinearRegression(fit_intercept=False), best_parameters={'fit_intercept': False}, best_score=0.6393153733826),
 GridSearchedBestModel(model_serial_number='module_1', model=RandomForestRegressor(min_samples_leaf=2, n_estimators=40), best_model=RandomForestRegressor(min_samples_leaf=2, n_estimators=80), best_parameters={'min_samples_leaf': 2, 'n_estimators': 80}, best_score=0.8050101845299591)]
```

In [10]:

```
1 best_model = model_factory.get_best_model(x,y,0.79)
```

```
Fitting 4 folds for each of 2 candidates, totalling 8 fits
[CV] END .....fit_intercept=True; total time=0.0s
[CV] END .....fit_intercept=False; total time=0.0s
Fitting 4 folds for each of 9 candidates, totalling 36 fits
[CV] END .....min_samples_leaf=2, n_estimators=50; total time=4.4s
[CV] END .....min_samples_leaf=2, n_estimators=50; total time=4.5s
[CV] END .....min_samples_leaf=2, n_estimators=50; total time=4.5s
[CV] END .....min_samples_leaf=2, n_estimators=50; total time=4.5s
[CV] END .....min_samples_leaf=2, n_estimators=100; total time=9.2s
[CV] END .....min_samples_leaf=2, n_estimators=100; total time=9.2s
[CV] END .....min_samples_leaf=2, n_estimators=100; total time=9.1s
[CV] END .....min_samples_leaf=2, n_estimators=100; total time=9.1s
[CV] END .....min_samples_leaf=2, n_estimators=80; total time=7.4s
[CV] END .....min_samples_leaf=2, n_estimators=80; total time=7.3s
[CV] END .....min_samples_leaf=2, n_estimators=80; total time=7.4s
[CV] END .....min_samples_leaf=2, n_estimators=80; total time=7.5s
[CV] END .....min_samples_leaf=4, n_estimators=50; total time=4.1s
[CV] END .....min_samples_leaf=4, n_estimators=50; total time=4.1s
[CV] END .....min_samples_leaf=4, n_estimators=50; total time=4.0s
[CV] END .....min_samples_leaf=4, n_estimators=50; total time=4.0s
[CV] END .....min_samples_leaf=4, n_estimators=100; total time=8.2s
[CV] END .....min_samples_leaf=4, n_estimators=100; total time=8.1s
[CV] END .....min_samples_leaf=4, n_estimators=100; total time=8.1s
[CV] END .....min_samples_leaf=4, n_estimators=100; total time=8.1s
[CV] END .....min_samples_leaf=4, n_estimators=80; total time=6.5s
[CV] END .....min_samples_leaf=4, n_estimators=80; total time=
```

```
6.4s
[CV] END .....min_samples_leaf=4, n_estimators=80; total time=
6.5s
[CV] END .....min_samples_leaf=4, n_estimators=80; total time=
6.5s
[CV] END .....min_samples_leaf=6, n_estimators=50; total time=
3.8s
[CV] END .....min_samples_leaf=6, n_estimators=50; total time=
3.7s
[CV] END .....min_samples_leaf=6, n_estimators=50; total time=
3.7s
[CV] END .....min_samples_leaf=6, n_estimators=50; total time=
3.8s
[CV] END .....min_samples_leaf=6, n_estimators=100; total time=
7.5s
[CV] END .....min_samples_leaf=6, n_estimators=100; total time=
7.7s
[CV] END .....min_samples_leaf=6, n_estimators=100; total time=
7.6s
[CV] END .....min_samples_leaf=6, n_estimators=100; total time=
7.5s
[CV] END .....min_samples_leaf=6, n_estimators=80; total time=
6.2s
[CV] END .....min_samples_leaf=6, n_estimators=80; total time=
6.0s
[CV] END .....min_samples_leaf=6, n_estimators=80; total time=
6.0s
[CV] END .....min_samples_leaf=6, n_estimators=80; total time=
6.0s
```

In [11]:

```
1 best_model.best_model
```

Out[11]:

```
RandomForestRegressor(min_samples_leaf=2)
```

In [12]:

```
1 model_factory.grid_searched_best_model_list[0]
```

Out[12]:

```
GridSearchedBestModel(model_serial_number='module_0', model=LinearRegression(),
best_model=LinearRegression(fit_intercept=False), best_parameters=
{'fit_intercept': False}, best_score=0.6393153733826)
```

In [26]:

```
1 model_factory.initialized_model_list
```

Out[26]:

```
[InitializedModelDetail(model_serial_number='module_0', model=LinearRegression(), param_grid_search={'fit_intercept': [True, False]}, model_name='sklearn.linear_model.LinearRegression'),  
 InitializedModelDetail(model_serial_number='module_1', model=RandomForestRegressor(min_samples_leaf=2, n_estimators=40), param_grid_search={'min_samples_leaf': [2, 4, 6], 'n_estimators': [50, 100, 80]}, model_name='sklearn.ensemble.RandomForestRegressor')]
```

In []:

```
1
```

In [1]:

```
1 from housing.entity import housing_predictor
```

In [5]:

```
1 housing_data=housing_predictor.HousingData(-118.39,34.12,29.0,6447.0,1012.0,2184.0,9
```

In [6]:

```
1 housing_data.get_housing_data_as_dict()
```

Out[6]:

```
{'longitude': [-118.39],  
'latitude': [34.12],  
'housing_median_age': [29.0],  
'total_rooms': [6447.0],  
'total_bedrooms': [1012.0],  
'population': [2184.0],  
'households': [960.0],  
'median_income': [8.2816],  
'ocean_proximity': ['<1H OCEAN']}
```

In [8]:

```
1 df=housing_data.get_housing_input_data_frame()
```

In [9]:

```
1 from housing.entity.housing_predictor import HousingPredictor
```

In [12]:

```
1 model_path="/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningProject/  
2 housing_predictor= HousingPredictor(model_dir=model_path)
```

In [13]:

```
1 housing_predictor.model_dir
```

Out[13]:

```
'/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningProject/  
machine_learning_project/saved_models'
```

In [14]:

```
1 housing_predictor.get_latest_model_path()
```

Out[14]:

```
'/home/avnish/iNeuron_Private_Intelligence_Limited/MachineLearningProject/  
machine_learning_project/saved_models/20220706202006/model.pkl'
```

In [15]:

```
1 housing_predictor.predict(df)
```

Out[15]:

424328.0048828125

In []:

```
1
```

In [1]:

```
1 import pandas as pd
```

In [2]:

```
1 da={"a":[1,2,3]}
```

In [4]:

```
1 df=pd.DataFrame(da)
```

In [7]:

```
1 df=df.astype('str')
```

In [11]:

```
1 df+" "+df
```

Out[11]:

a
0 1 1
1 2 2
2 3 3

In []:

```
1
```