**1. Data types:** A data type tells a kind of var and size of data it can store.

***1. Primitives datatypes:***

byte //java

void (noting) 0 bytes

char(0-255),wchar\_t 1 and 2- 4 byes

bool (true false) 1 bytes

int, short int, long int 4 2 8 bytes

float,

double, long double 8 and 16 bytes

*Datatype modifiers:* used with the built-in data types to modify the length of data that a particular data type can hold.

short: int

long: int, double

signd: int, char, long

unsignd: int, char, short

***2. Derived datatypes:***

array //vector, list, sequence

function

string //java

date //java

pointer

reference

***3. Non premitive datatypes:***

class

interface

struct

union

enum

typedef

Data types are actually classes and variables are instance (object) of these classes./**/python**

**Numeric type:**

int

float

complex.

**Sequence type:**

list[] //x = ["apple", "banana", "cherry"]

touple() //x = ("apple", "banana", "cherry") //struct or record

range //x = range(6)

**Binary types:**

bytes //x = b"Hello"

bytearray() //x = bytearray(5)

memoryview()// x = memoryview(bytes(5))

**Set types:**

set{} // = {"apple", "banana", "cherry"}

frozenset{} //x = frozenset({"apple", "banana", "cherry"})

Text type:

string '' "" //x = "Hello World"

**Boolean type:**

bool

**Dictionary:**

dict{} //x = {"name" : "John", "age" : 36}

\* **To determine the size of a data type or a variable.**

sizeof(datatype): .//c c++

datatype.SIZE/8: to determine size of datatype //java

sys.getsizeof():return type in bytes //python

\* **To check if an object belongs to a particular class or not.**

isinstance(): // python

instanceof(): // java

The condition is true if 'a/obj' is an instance of X/class, or if a is an instance of a subclass of X.

used to test whether the object is an instance of the specified type (class or subclass or interface).

It is also known as type comparison operator because it compares the instance with type.

\* **Type casting and conversation**:

converts a data type into another data type in both ways manually and automatically.

1.implict: can data loss double > float > long > int > short > byte

2.explict:

datatype(var/val) //c++

var(datatype) //java c c++

3. ValueOf(var/dt): //java

type conversion from int to String

int n=10;

string s=String.valueOf(n)

4.ParseInt(string data): //java

Type conversion from String to other

string s="10";

int n=Integer.parseInt(s);

5. Type conversion operators: //c++

i. **static\_cast()<>** :

It is a compile time cast.It does things like implicit conversions between types

ii. **dynamic\_cast <new\_type>** (expression)

iii. **const\_cast()<>:**

const\_cast can be used to change non-const class members inside a const member function.

iv. r**einterpret\_cast(ptr var) <>**:

It is used to convert a pointer of some data type into a pointer of another data type.

6. explict Type casting functions: //python

nt(): other type to integer type (execpt complex)

float(): other type to float type. (execpt complex)

complex(): other type to complex type.

bool(): other type to bool type.

str(): other type to string type.

7. 5 inbuild different type casting functions available. //c

atof(): string to float data type.

atoi(): string to int data type.

atbol(): string to long data type.

itoba(): int to string data type.

ltoa(): long to string data type.

**\*To Know type of data of a var/object:**

Python:

obj reference: **type(obj)** to determine data type of the var //to check data type

obj identity: i**d(obj)** Using the id() function, you can verify that two variables indeed point to the same object(address)// to check address

Java:

getclass().1.2

1. **getName()** method of java.lang.Class class is used to get the name of the entity,// returns class name of obj of package and that entity can be class, interface, array, enum, method, etc. of the class object.

2. **getSimpleName()**: returns the simple name of this class in the form of String. // give you only the name of the class:

3. **equals(class obj)**

C++:

**typeid().name()** //returns ref of var/obj

**2. Oparators:**

1. Unary: ++, --

2. Bianry:

arithmetic: (+ - \* / % (//,\*\*))

operate or work with two operands,

% operator can only be used with integers.

relational: (<,<=,>,>=,==,!=)

used for the comparison of the values of two operands,

Relational operators are used in decision-making and loops.

assignment: (= += -= \*=/= %=(//= \*\*= &= |= ^= >>= <<=))

used to assigning value to a variable.

bitwise: (| & >> << ~ ^)

used to perform bit-level operations on the operands,

operators are first converted to bit-level and then the calculation is performed on the operands.

They can only be used alongside char and int data types.

Bitwise optrs act on operands as if they were strings of binary digits.//python

logical: (&&, ||, !(and or not ))

are used to check whether an expression is true or false.

commonly used in decision making.

Misccellaneous // c c++

ternary ? :

sizeof(): returns the size of data type. // sizeof(int) 4

\* Pointer to a variable.

& Returns the adress of a var. represents memory address of theoperand // &a

. accesses members of struct variables or class objects //obj.dm=12

-> used with pointers to access the class or struct variables //ptr->dm=12

<< prints the output value

>> gets the input value //cin>>a

spectial: //python

identity: used to check if two values (or variables) are located on the same part of the memory.

is

is not

membership: are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

in

not in

**3. Conditionals(if/else/switch)**

if

if-else

if-else if ladder

if-eliff ladder// python

Nested if

switch case

jump statements

break

continue

goto //c c++

return

**4. Loops:** which are capable of repeating some specific code several numbers of times

while(condition){}"a statement to execute continuously till the condition specified is true."

for(initilization; Conditional, increement){}

do{} while(condition){}"the condition is checked at the bottom of the loop."

Ranged Based for(variable:collection){}

range()//python

nested loops

Infinite loop

loop control statements

break

continue

goto //c c++

return

**5. Identifiers & literals:** only allowed characters for identifiers are ([A-Z],[a-z],[0-9]), ‘$‘, \_

50 Key words: are predefined, reserved words //java

3 literals: are data used for representing fixed values.(true, false, null,) /java //'none' python

**6. Variables:**

**Types of variables:**

Java:

Local,

Instance, (Non-Static or obj dm)

Static, (class dm): It cannot be local.

Parameter.

C++:

local,

global,

static,

automati,

external. An extern variable is available to other files too. extern vars are only declared; they are not defined

Parameter.

\* deaclaration & assignment:

int a; //declaration

a=10 // defination

Class obj1\*;

obj1=&obj2

\* initilization

int a=10 //static

int b=a; // dynamic

**7. Funtions:**

**Python:**

User-define,

Built-in,

Anonymous/lambda. //python

default (default constructor / static initilization)

int a;

Class obj1;

call by value(parameterized cons/ dynamic initilization)

call by address( copy constructor/ copy initialization )

overloading,

overriting.

**8. pointers:**

**9. structure/union:**

**10. data structues**

physical ds:

array:

one dime

two dime

multi dime

linked list: collection of nodes node= data+poointer

single,

double,

circular,

logical ds:

stack

queue