C# C++ Python JS

# 1\ Class

## C#

public class Student: Person, ICompare<Student>

{

public int grade;

public ClassA(string name, int age, int grade):base(name, age)

{

this.grade = grade;

}

public void Info()

{

Console.WriteLine();

}

}

var a = new Student("kelly", 14, 10);

## C++

class Student : public Person

{

public:

int grade;

Student(string name, int age, int grade);

void Info();

}

Student::Student(string name, int age, int grade): Person(name, age)//or default constructor of base class will be called

{

this.grade = grade;

}

void Student::Info()

{

cout<<

}

Student a; // default constructor

Student a("kelly", 14, 10);

Student\* pt = new Student();

## Python

class Student(Person):

grade = 0

\_\_id = 0 #private property

def \_\_init\_\_(self, name, age, grade):

Person.\_\_init\_\_(self, name, age)

self.grade = grade

def \_\_add\_\_(self, other):

return Student(self.name+other.name)

def info(self):

print()

def \_\_privateFunc(self):

a = Student("keylly", 14, 10);

## JS

function Student(name, age, grade)

{

Person.call(this, name, age);

this.grade = grade;

this.info = function () {};

}

var a = new Student("kelly", 14, 10);

# 2\ Data Structures

## C#

var a = new int[size];

var a = new List<int>();

1. a.Add() a.AddRange() a.Insert() a.InsertRange() a.Remove() a.RemoveAt() a.RemoveRange()
2. a.First() a.Last()
3. a.Count a.IndexOf() a.Find() a.CopyTo()
4. a.Sort() a.Reverse()

var a = new Dictionary<string, int>();

1. a.Add() a.Remove() a.ContainsKey() a.ContainsValue()
2. a.Count a.Keys a.Values
3. a[]

var t = new Tuple<int,int,int>();

1. t.Item1 t.Item2
2. Tuple cannot be modified once instaniate.

var s = new StringBuilder();

1. s.Append()

## C++

vector<int> a;

1. a.push\_back() void a.pop\_back() a.insert(where) a.erase(where)
2. a[] a.front() a.back()
3. a.size()
4. reverse(a.begin(), a.end()) sort(a.begin(), a.end())

map<string, int> a

1. a.insert(pair<string,int>(k, v)) a.erase(k key)
2. a.find(key)!=a.end() a.count(key)==1
3. a[] a.size()
4. a.begin(), a.end() as iterator

for (std::map<char,int>::iterator it=mymap.begin(); it!=mymap.end(); ++it)

std::cout << it->first << " => " << it->second << '\n'; //iterator is like pointer

set<int> a

1. a.insert() a.erase() a.clear()
2. a.begin() a.end() a.find() a.count()
3. a.size()
4. you can not use index[] to search element.

unordered\_map<k, v> a

unordered\_set<k> a

stack<int> a

1. a.push() void a.pop() a.top
2. a.size() a.empty()

queue<int> a

1. a.push() a.pop() a.front() a.back()
2. a.size() a.empty()

priority\_queue<int> a

1. a.push() a.pop() a.top()
2. a.size() a.empty()

array<int, size> a

1. fixed size

deque<int> a

1. double-ended queue, can insert and delete on both sides.

list<int> a

1. linked list, quick in insertation, but cannot access each element through index.

int a[2,3] is not ok

int a[2][3] is ok

but in c#

var a = new int[2,3]

var a = new int[2][];

Except containers, string is another import data tye.

string a(string is mutable, so do not consider efficiency when modifying the string)

1. a.size() a.length()
2. a[] a.back() a.front()
3. a.find() a.begin() a.end()
4. operator +=
5. a.insert() a.erase() a.push\_back() a.pop\_back()
6. a.copy() s.substr(startIdx, len) () a.replace() a.append()
7. std::to\_string(number)

stringstream a

1. Characters can be inserted and/or extracted from the stream using any operation allowed on both [*input*](http://www.cplusplus.com/istream) and [*output*](http://www.cplusplus.com/ostream)streams.

Other important things (algorithm headfile and others)

reverse()

sort(a.begin(), a.end() , [](MyClass a, MyClass b) { return a.p – b.p; })

copy(\*first, \*last, where)

min() max() abs()

INT32\_MAX

INT32\_MIN

rand()%N: generate random number [0, N-1]

pass vector<int> instead of int[] as parameter in c++

pass & for vector, and \* for user-defined class.

sizeof(arr\_name):#elements\*sizeof(element)

sizeof(pointer): decided by system, usually 4 bytes

strlen() always exclude terminator

sizeof() include terminator. When fitting a string to char[], terminator will be included in the array.

string s1 = “ab”; strlen(s1) = 2(exclude terminator); sizeof(s1) = 3

char s2[] = “ab”; sizeof(s2) = 3(include terminator)

char s3[10] = “ab”; sizeof(s3) = 10 strlen(s3) = 2

char\* s4 = “ab”; sizeof(s4) = 4(pointer)

1\ convert int to string

string s = to\_string(n);

2\ substr

str.substr(int pos, int len)

in C#, we have delegate and Func to pass function parameter: void Sort(IList<T> arr, Func<T, T, int> comparator)

in C++, we have function pointer and std::function<return\_type(para\_type1, para\_type2)>. I think the second one is more convenient. Argument past to could be [](T item1, T item2){return item1-item2;}

long is 64 bits in C#, but 32bits in C++

long long is 64 bits in C++

## Python

**Number(immutable)**

Including int, float and complex

a = int(x) a = float(x) a = complex(x, y)

5/2 5//2 5\*\*2

pi e float(‘inf’) float(‘-inf’)

abs() min() max()

math.ceil() math.floor() math.sqrt()

random.choice(lst) random.random() random.shuffle(lst)

**String(immutable)**

no char in python,

cannot assign to element like s[0] = “a”

ord(‘a’) returns 97

chr(97) returns ‘a’

a = input(“Please input a number: ”) // a is always a string

a+b a\*2 a[] a[:] in not in str() s.split()

print(“hello % s %d” % (“world!”, 2000))

**List**

list和tuple是Python内置的有序集合，一个可变，一个不可变。根据需要来选择使用它们。

+ lst\*n lst[] lst[:] in not in

for x in lst:

len() list(seq) del lst[]

lst.append() lst.pop(w = -1) lst.insert() lst.remove()

lst.index() lst.count()

lst.reverse() lst.sort() lst.copy() sorted(lst)

**Tuple(immutable)**

tuple is immutable, is indexed

cannot assign to element like tpl[1] = 1

tpl = () tpl = (1,) tpl = (1,2,3)

len() + \* in for x in tpl [] [:]

**Set**

set has no repeatition, not indexed

\_set = {1,2,3,4} \_set = set()

len() & ^ - | in add() remove()

**Dict**

No repeat key, key is immutable so number, string and tuple can be the key, not list.

dict = {}

dict[“name”] = 1 should not used as right value before you assign a value to the key.

del dict[“name”] dict.pop(key) dict.get(key, -1)

len() key in dict key in dict.keys() k, v in dict.items()

## JS

a = []

1. a.length
2. a.push() a.pop() a.unshift() a.shift()
3. a.map() a.reduce() a.filter()
4. a.sort() a.reverse()

a = new Map()

1. a.set(key, value) a.has(key) a.get(key) a.delete(key)

# 3\ Data Type

## Python

number(int, float, complex), string, list, tuple, set, map

all reference types, but number, string and tuple are immutable once initialized.

int() float()

str()

list(seq)

## JS

Number, string, boolean, array, function, object(including set and map)

Number, string and boolean are 原始值

The others are reference types.

Number() String() .toString()

只有JS有undefined值， 所以访问数组元素a = [], a[1] = 1可以直接赋值，此时a[0] = undefined

Python则要使用append向数组增加元素

a = []

a.append([])

a[0].append(0)

a[0][0] = 0 is not right

## C#

Int double float boolean string char List<> Queue<> Stack<> Set<> Dictionary<>

.ToString() Convert.ToInt32() Int32.Parse()

C#可以只用matrix[i,j]访问元素

Python和c++中都只能用m[i][j]

## C++

Vector<> queue<> stack<> map<> set<>