November 2024

Introduction to C++: Workshop Four

Dr. Alexander Hill

a.d.hill@liverpool.ac.uk

LIV.INNO

Last Week

- Memory addresses and pointers
- Passing vectors into functions
- Basic saving of text-based data
- Plotting with Python

Aim of Workshop Four

- Homework recap
- Classes in C++
- Monte Carlo Background

Resources

- alex-hill94.github.io/#WS4
- https://www.w3schools.com/cpp/cpp_classes.asp

Challenge Six (Homework)

- Create a function called func() that takes in a vector, and computes:

- $f(x) = \begin{cases} e^{-1/x^2} & x \neq 0 \\ 0 & x = 0 \end{cases}$

- Create a vector with a range -10 to 10 inside main(), and pass it into func()

- **Save the input and output to a file 'data.py'. Bonus points if the file writing is done inside a function called write_out(string filename, vector<int>& vect)**

- Plot the input and output using a separate python file, 'plot.py'

- Compile, run, and plot this all in the command line

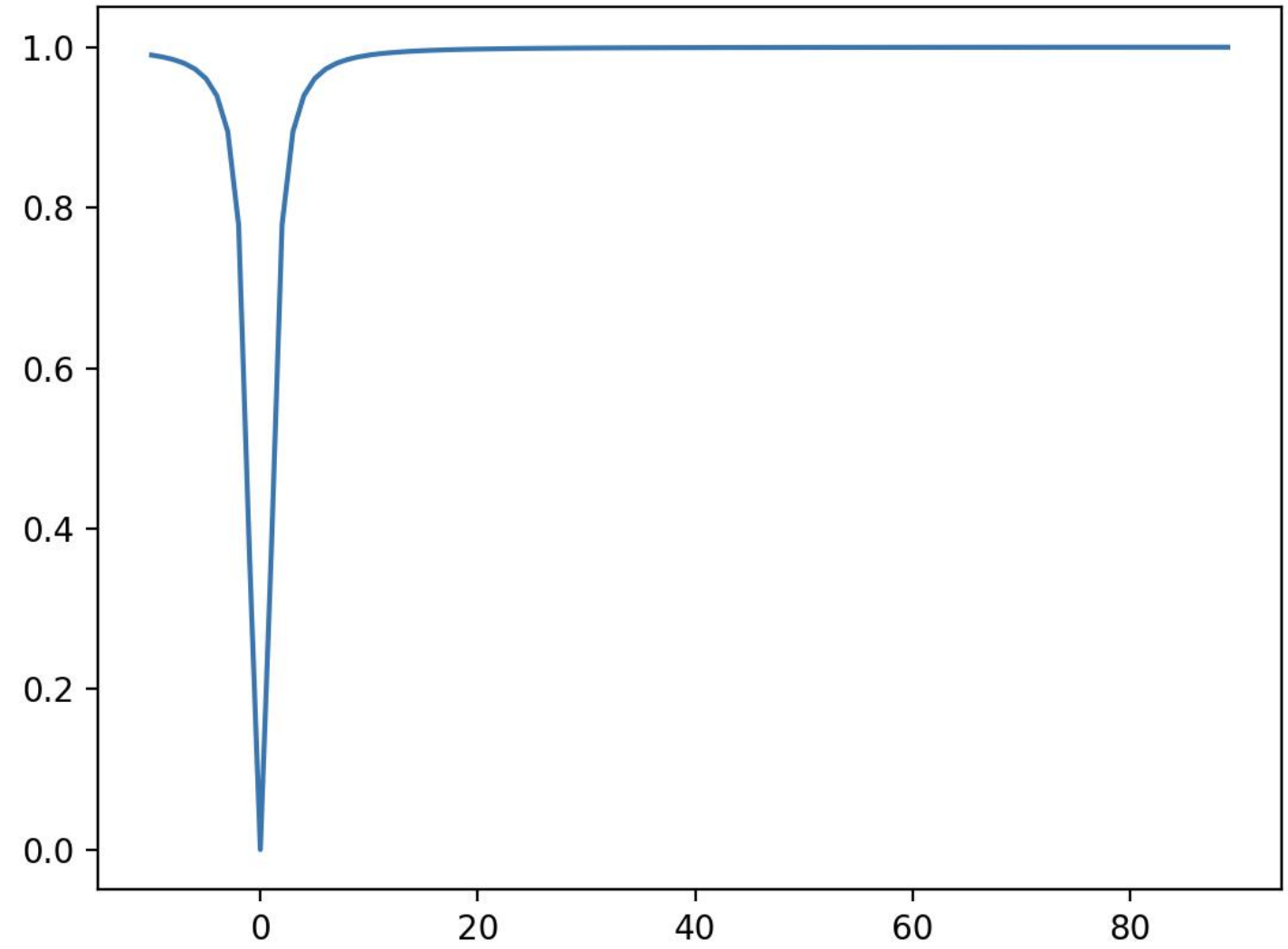
Liam

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <vector>
using namespace std;

void make(int n, int start, vector<int>& vect);
void func(const vector<int>& input, vector<double>& output);
void write_out(string filename, const vector<int>& input, const vector<double>& output);
void plot(string filename, string data);
```

```
void make(int n, int start, vector<int>& vect){
int end = start + n;

for (int i = start; i < end; ++i){
vect.push_back(i);
}
}

void func(const vector<int>& input, vector<double>& output){
int n = input.size();

for (int i = 0; i < n; ++i){
if (input.at(i) == 0)
{
output.push_back(0);
}
else{
output.push_back(exp(-1/pow(input.at(i), 2)));
}}}
```

```
int main(){

int n, start;

cout << "How many values would you like? \n";
cin >> n;
cout << "What value would you like to start with? \n";
cin >> start;

vector<int> x;
vector<double> out;

make(n, start, x);
func(x, out);

for (int j: x){
cout << "Values of x =" << j << "\n";
}

for (double j: out){
cout << "Values of f(x) = " << j << "\n";
}

string dataname = "data.py";
string plotname = "plot.py";

write_out(dataname, x, out);
plot(plotname, dataname);

return 0;
}
```

Liam

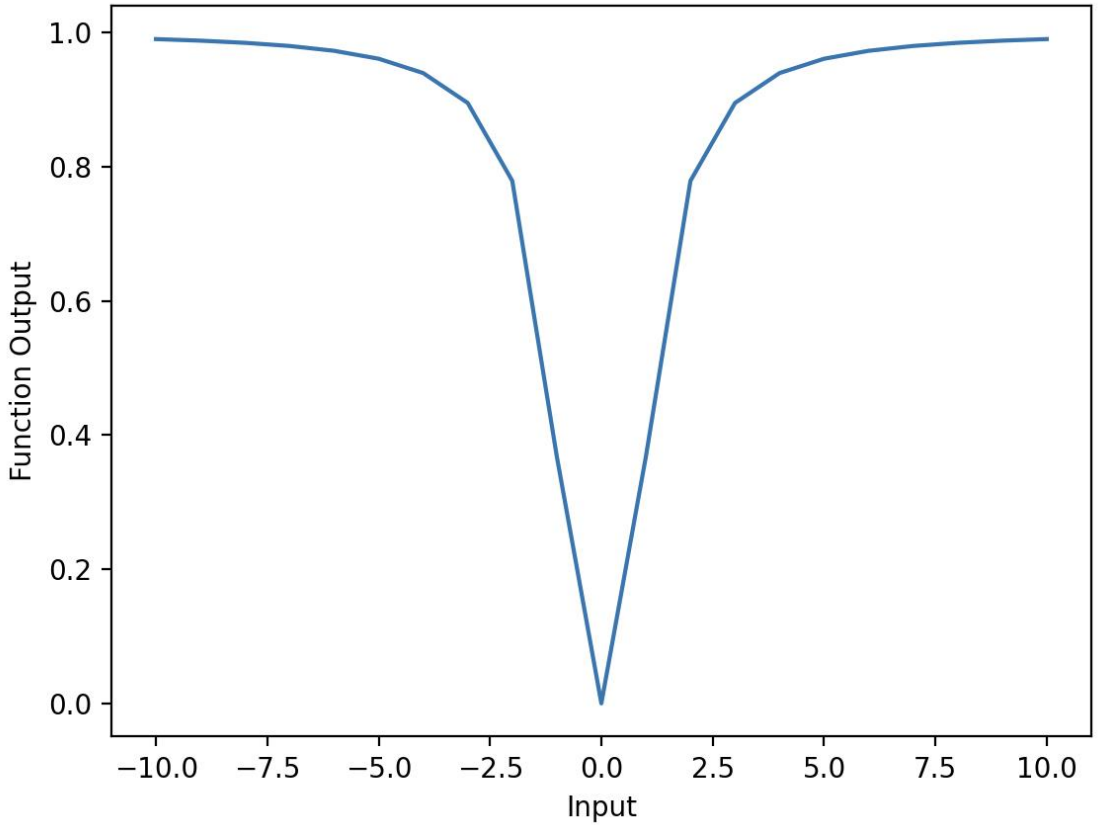



```
$ ./liam
How many values would you like?
100
What value would you like to start with?
-10
Values of x =-10
Values of x =-9
Values of x =-8
Values of x =-7
Values of x =-6
Values of x =-5
Values of x =-4
Values of x =-3
Values of x =-2
Values of x =-1
Values of x =0
Values of x =1
Values of x =2
Values of x =3
Values of x =4
Values of x =5
Values of x =6
Values of x =7
Values of x =8
Values of x =9
Values of x =10
Values of x =11
Values of x =12
Values of x =13
Values of x =14
```

Jak

```cpp
int main() {
// Generate vector with range -10 to 10
vector<int> inputVec;
for (int i = -10; i <= 10; i++) {
inputVec.push_back(i);
}

// Calculate output
vector<double> outputVec = func(inputVec);

// Write to file 'data.py'
write_out("data.py", inputVec, outputVec);

cout << "Data written to 'data.py'." << endl;
return 0;
}
```

```cpp
// Function to get outputs
vector<double> func(const vector<int>& inputVec) {
vector<double> outputVec;
for (int x : inputVec) {
if (x != 0) {
outputVec.push_back(exp(-1/(x*x)));
} else {
outputVec.push_back(0);
}
}
return outputVec;
}
```



Jak

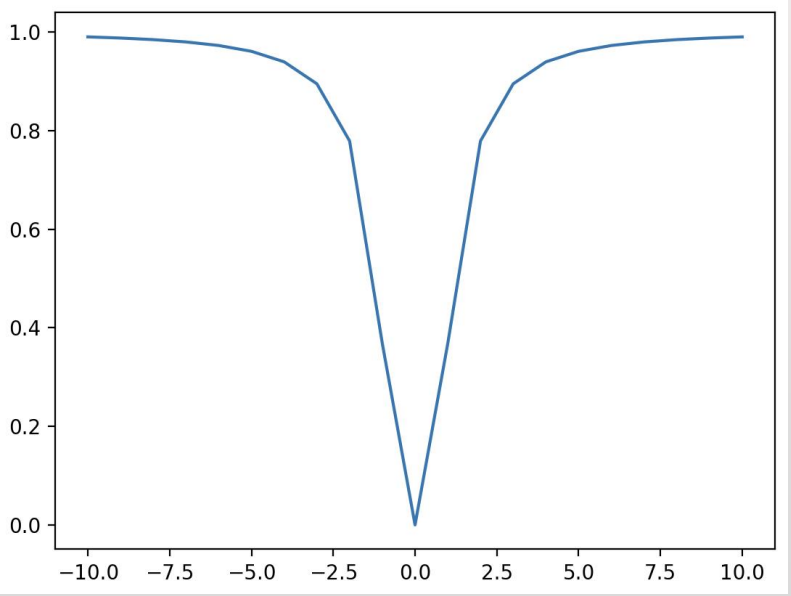
```
// Function to write input and output to python file
void write_out(const string& filename, const vector<double>& inputVec, const vector<double>& outputVec) {
ofstream file(filename);
if (file.is_open()) {
file << "# data.py\n";
file << "input_values = " << "[";
for (int i = 0; i < inputVec.size(); i++) {
file << inputVec[i] << (i < inputVec.size() - 1 ? ", " : "");
}
file << "]\n";
file << "output_values = " << "[";
for (int i = 0; i < outputVec.size(); i++) {
file << outputVec[i] << (i < outputVec.size() - 1 ? ", " : "");
}
file << "]\n";
file.close();
} else {
cerr << "Unable to open file";
}
}
```

condition ? expression_if_true : expression_if_false;

Naomi

```
(base) alexhill at Alexs-MacBook-Air-2 in
~/Documents/UOL/LIVINNO/Teaching/C++_Workshops/2024/WORKSHOPS/WS4/Naomi
$ ls
runplot.sh              week3hw.cpp
(base) alexhill at Alexs-MacBook-Air-2 in
~/Documents/UOL/LIVINNO/Teaching/C++_Workshops/2024/WORKSHOPS/WS4/Naomi
$ bash runplot.sh
```

```
#!/bin/sh
g++ week3hw.cpp -o run
./run
python plot.py
```



Naomi

```cpp
#include <iostream>
#include <cmath>
#include <vector>
#include <string>
#include <fstream>
using namespace std;
void write_out(string filename, vector<int>& input, vector<double>& output) {
ofstream myfile;
myfile.open (filename);
myfile << "import numpy as np" << "\n";
myfile << "import matplotlib.pyplot as plt" << "\n";
myfile << "x = np.array([" << "\n";
for (int i = 0; i < input.size(); ++i){
myfile << input[i] << ", ";
}
myfile << "])" << "\n";
myfile << "y = np.array([" << "\n";
for (int i = 0; i < output.size(); ++i){
myfile << output[i] << ", ";
}
myfile << "])" << "\n";
myfile << "plt.figure" << "\n";
myfile << "plt.plot(x, y)" << "\n";
myfile << "plt.show()" << "\n";
myfile.close();
}
```

```cpp
void func(vector<int>& input, vector<double>& output) {
for (int i = 0; i < input.size(); ++i){
output[i] = exp(-1 / pow(input[i], 2));
}}

int main() {
// Create a vector of integers from -5 to 5
vector<int> input;
int k = 10;
for (int i = -k; i <= k; ++i) {
input.push_back(i);
}

vector<double> output(input.size());
func(input, output);
write_out("plot.py", input, output);

return 0;
}
```

Rosie

```cpp
int main() {

//define input vector between -10 and 10
vector<double> input = {-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

//pass input vector through func function to get output vector
vector<double> output = func(input);

//saving input and output vectors to the same data file
writeout("data.py", output, "output", true);
writeout("data.py", input, "input", false);


return 0;
}
```

```cpp
void writeout(string filename, vector<double> writevector, string arrayname, bool overwrite){
//create file object
ofstream(datafile);

//open file, either overwriting or appending
if (overwrite == true) {
//open data file
datafile.open(filename);
datafile << "import numpy as np" << "\n"; //assumes that if you're appending to a file, it is already using numpy
}
if (overwrite == false) {
datafile.open(filename, fstream::app);
}

.....
.....
```

Challenge Six Solution: Alex Modular Approach

```
#include <cmath>
#include <iostream>
#include <vector>
#include <fstream>

using namespace std;

void f_x(const double& x, double& fx)
{some stuff}
void f_master(const vector<double>& input, vector<double>& output)
{some stuff}
void write_out_vec(const string& filename, string& array_name , const
vector<double>& output)
{some stuff}
int main()
{some stuff}
```

Alex

New data type – automatically assigns a data type based on the initialisation

```
void f_x(const double& x, double& fx)
{
if (x == 0)
{
fx = 0;
}
else
{
fx = exp(-1/pow(x,2));
}
}

void f_master(const vector<double>& input, vector<double>& output)
{
double temp_val;
for (auto i: input){
f_x(i, temp_val);
output.push_back(temp_val);
}

}
```

Fix the vector input

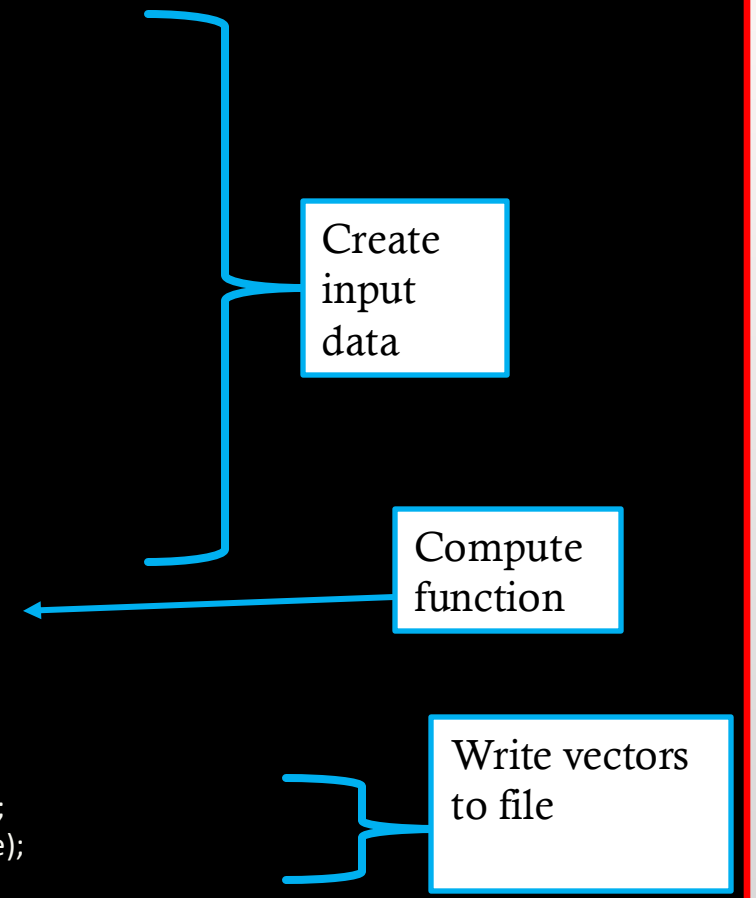
Create a temporary variable

temp_val is changed by f_x

Add new temp val to output vector

Alex

```
int main()

{
string filename = "data1.py";
double k;
cout << "n_vals = ?";
cin >> k;
double llim = -10;
double ulim = 10;
double seps = (ulim - llim)/(k - 1);
vector<double> x_range(k);
vector<double> fx_range;

for (int i =0; i < k; ++i)
{
x_range.at(i) = llim + (seps * i);
}

f_master(x_range, fx_range);

string x_lab = "x_range";
string fx_lab = "fx_range";

write_out_vec(filename, x_lab , x_range);
write_out_vec(filename, fx_lab , fx_range);

return 0;
}
```



Alex

Default value

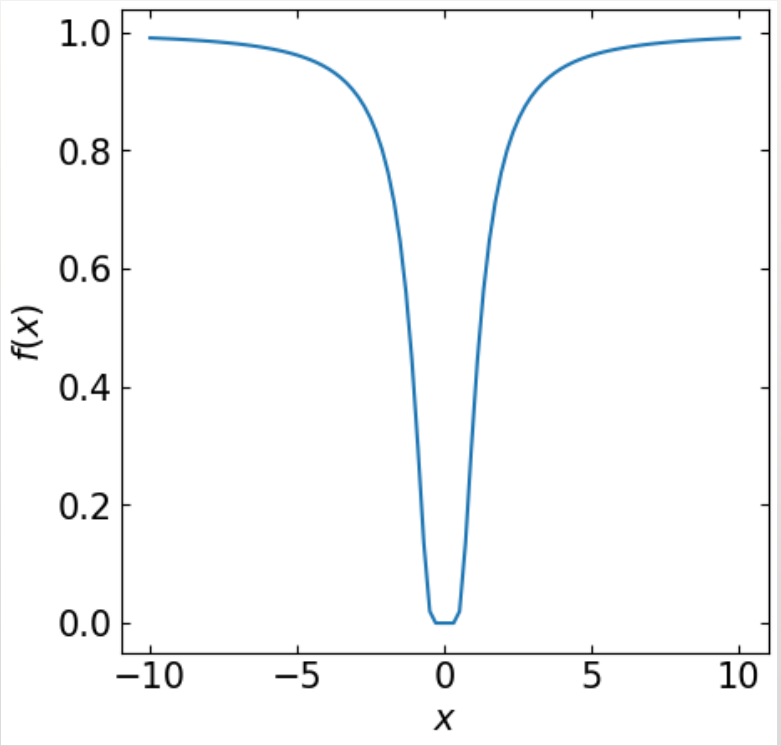
```
void write_out_vec(const string& filename, string& array_name , const vector<double>& output, bool initialise = true)
// Writes out data in numpy form. If initialise == true, creates file from scratch and includes top lines needed
{
ofstream myfile;
if (initialise){
myfile.open (filename);
myfile << "import numpy as np" << "\n" << "\n";}

else{myfile.open (filename, std::ios_base::app);}
myfile << array_name << " = np.array((" << "\n";


for (auto i = 0; i < output.size(); ++i)
{if (i != (output.size() - 1))
{myfile << output.at(i) << ", " << "\n";}
else
{myfile << output.at(i) << "\n";}}

myfile << "))" << "\n";
myfile.close();}
```

Opens data file without wiping what's already there

Loops over vector and writes out. If not at the end, then include comma

Alex

```
import matplotlib
import matplotlib.pyplot as plt
from data1 import *

FS = 15
matplotlib.rcParams.update({'font.size': FS})

fig, axs = plt.subplots(1,1, figsize = [5,5])

axs.plot(x_range, fx_range)

axs.set_xlabel('$x$')
axs.set_ylabel('$f(x)$')
axs.xaxis.set_tick_params(direction= 'in', which='both', right =True, top = True)
axs.yaxis.set_tick_params(direction= 'in', which='both', right =True, top = True)


plt.savefig('test.png',bbox_inches = 'tight', pad_inches = 0.05)
```



Takeaways

- Use const to fix vectors when they go into functions where you are sure they shouldn’t be changed
- Try to make your code easily adaptable
- Try writing python inside a C++ script, and executing with bash or other command line languages

Challenge Seven

- Aim: compute x^2 + 4x – 10 for an arbitrary input, save input/output data, plot the results
- In main():
  - Create an input vector x = (start, ….., end) with len(x) = k
  - Set start = -100, end = 100, k = 1e6 initially
  - Create a blank vector called output
  - Pass input and output to a function **func**
  - Pass input/output to a function called **write_out**
- In func():
  - Import memory addresses for input/output, i.e. don't create copies
  - Fix input, allow output to change
  - Modify output according to x^2 + 4x – 10
- In write_out():
  - Save data to a named python script data.py
- Python script:
  - Load in data, plot results (you can do this inside a C++ function if you like)

Classes

- Basics of Classes in C++
  - Attributes and Methods
  - Constructors
  - Access specifiers

Object oriented programming

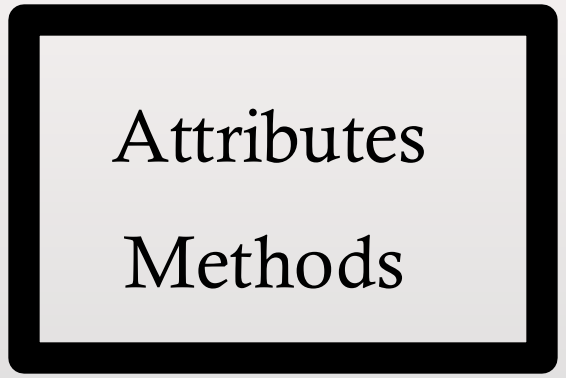
- Procedural programming specifies the specific steps a programme must take
- OOP creates objects that contain both data and functions
- Why? DRY! (Don’t repeat yourself)

Classes and Objects

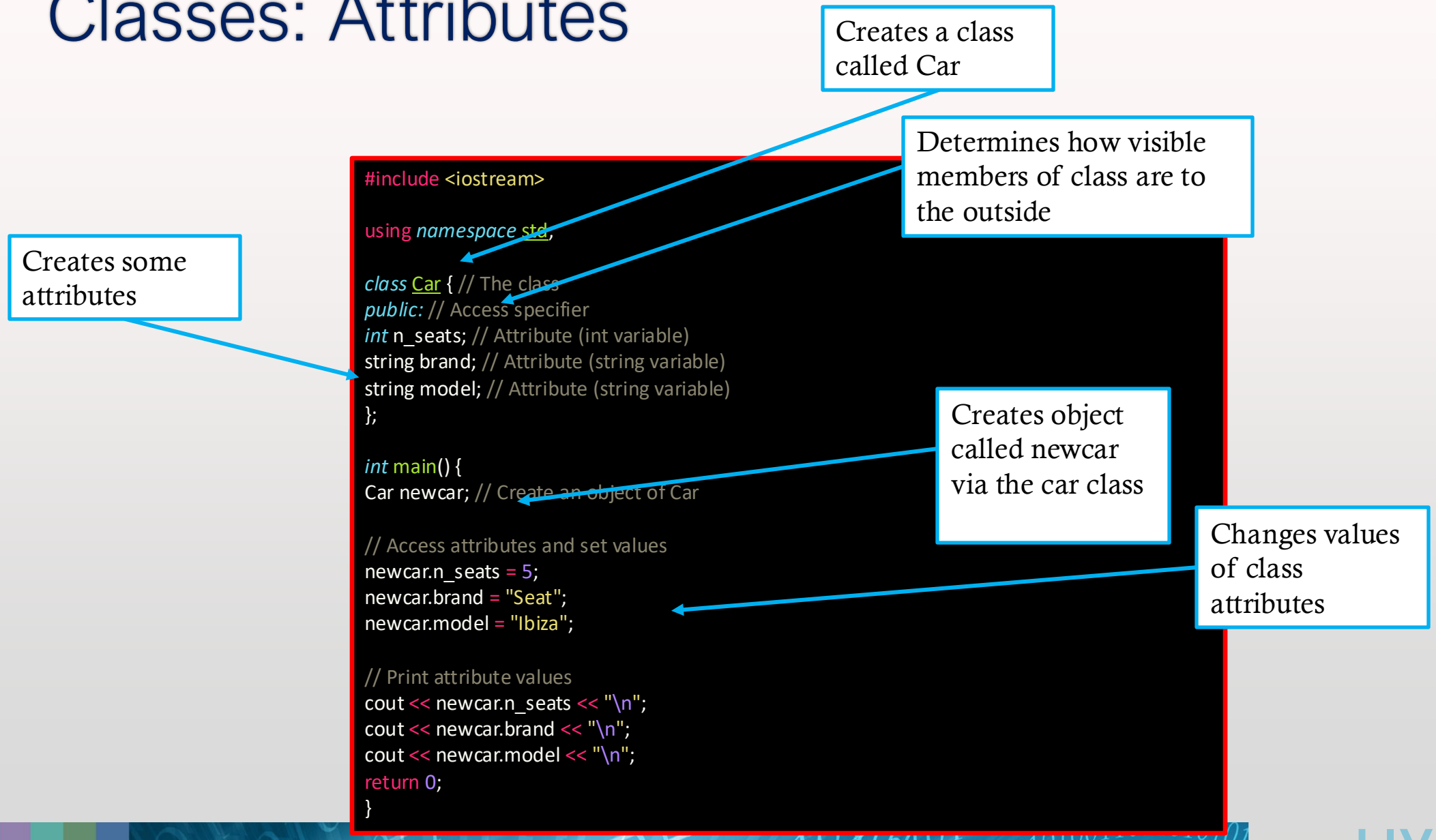
- A ‘Class’ is a template for an ‘Object’
- An ‘Object’ is an instance of a Class
- E.g. for the ’Car’ class, you may have ‘Volvo’, ‘Audi’ and ‘Ford’ objects

Classes and Objects

- Classes have **attributes** (variables, e.g. weight, colour)
- Classes have **methods** (functions, e.g. brake, open door)
- Attributes and functions are referred to as **class members**



Classes: Attributes



Classes: Attributes

```
int main() {
Car peoplecarrier; // Create an object of Car
Car sportscar; // Create an object of Car

// Access attributes and set values
peoplecarrier.n_seats = 7;
sportscar.n_seats = 2;
return 0;
}
```

You can create multiple objects of a given class

Classes: Methods

```
#include <iostream>
using namespace std;

class Car {
public:
void Beep() {
cout << "Beep Beep" << endl;
};

int speed(int maxSpeed);
};

int Car::speed(int maxSpeed) {
return maxSpeed;
}

int main() {
Car myObj; // Create an object of Car
myObj.Beep();
cout << myObj.speed(200); // Call the method with an argument
return 0;
}
```

Creates a function inside the class

Declare a function inside the class

Define the function outside the class

Scope resolution operator '::'

Constructors

Constructor is called with the class name

Pass arguments to constructors when objects are created

```
#include <iostream>
using namespace std;

class Car { // The class
public: // Access specifier
string brand; // Attribute
Car(string x) { // Constructor with parameters
brand = x;
}
};

int main() {
// Create Car objects and call the constructor with different values
Car carObj1("BMW");
Car carObj2("Ford");

// Print values
cout << carObj1.brand << "\n";
cout << carObj2.brand << "\n";
return 0;
}
```

- Constructors are special functions that are called whenever an object is created
- You can also define constructors outside of a class

Access Specifier

- Access specifiers can be:
- Private: members cannot be viewed outside the class
- Public: members are accessible outside the class
- Protected: members cannot be accessed outside the class, however they can be accessed in inherited classes
- By default, members are private

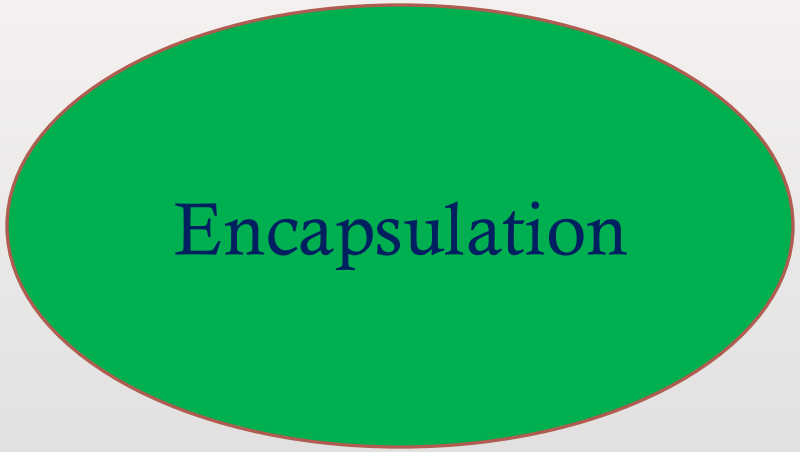
```
class Car {
public:
void Beep() {
cout << "Beep Beep" << endl;
}
```

Access Specifiers

```cpp
#include <iostream>
using namespace std;

class MyClass {
public: // Public access specifier
int x; // Public attribute
private: // Private access specifier
int y; // Private attribute
};

int main() {
MyClass myObj;
myObj.x = 25; // Allowed (public)
myObj.y = 50; // Not allowed (private)
return 0;
}
```

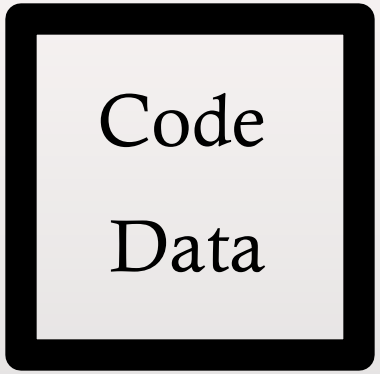
```
$ g++ -std=c++11 -o soln lesson_ex.cpp
lesson_ex.cpp:14:9: error: 'y' is a private member
of 'MyClass'
  myObj.y = 50;  // Not allowed (private)
        ^
```

Principles of OOP



Encapsulation

- Encapsulation ensures that code and data are in a black box (if desired)
- You can use the ‘private’ access specifier to ensure this
- It’s often the practice to use functions to retrieve (get) and define (set) attributes



Encapsulation

```
#include <iostream>
using namespace std;

class Employee {
private:
int salary;

public:
void setSalary(int s) {
salary = s;
}
int getSalary() {
return salary;
}};

int main() {
Employee myObj;
myObj.setSalary(50000);
cout << myObj.getSalary();
return 0;}
```

```
(base) alexhill at Alexs-Air in
~/Documents/UOL/Teaching/C++_Workshops/Workshops/WS4/Scripts
$ ./soln
50000
```

Inheritance

You can have multilevel inheritance (Grandparent -> Parent -> Child)

You can also have multiple inheritance, i.e.:

class MyChildClass: public MyClass, public MyOtherClass

```
#include <iostream>
using namespace std;

// Base class
class Vehicle {
public:
string brand = "Ford";
void honk() {
cout << "Toot, toot! \n" ;}};

// Derived class
class Car: public vehicle {
public:
string model = "Mustang";};

int main() {
Car myCar;
myCar.honk();
cout << myCar.brand + " " + myCar.model;
return 0;}
```

Derived class inherits from parent class using ':' symbol

```
$ ./soln
Toot, toot!
Ford Mustang
```

Inheritance: Access Specifiers

```
// Base class
class Employee {
protected: // Protected access specifier
int salary;
};

// Derived class
class Programmer: public Employee {
public:

void setSalary(int s) {
salary = s;}

int getSalary() {
return salary;
}}
```

'Protected' data can be accessed by the inherited class

Overriding base classes

```cpp
#include <iostream>
using namespace std;

// Base class
class Animal {
public:
void animalSound() {
cout << "The animal makes a sound \n";
}
};

// Derived class
class Pig : public Animal {
public:
void animalSound() {
cout << "The pig says: weeee\n";
}
};
```

```cpp
// Derived class
class Dog : public Animal {
public:
void animalSound() {
cout << "The dog says: woof \n";
}
};

int main() {
Animal myAnimal;
Pig myPig;
Dog myDog;

myAnimal.animalSound();
myPig.animalSound();
myDog.animalSound();
return 0;
}
```

```
$ ./soln
The animal makes a sound
The pig says: weee
The dog says: woof
```

Challenge Eight

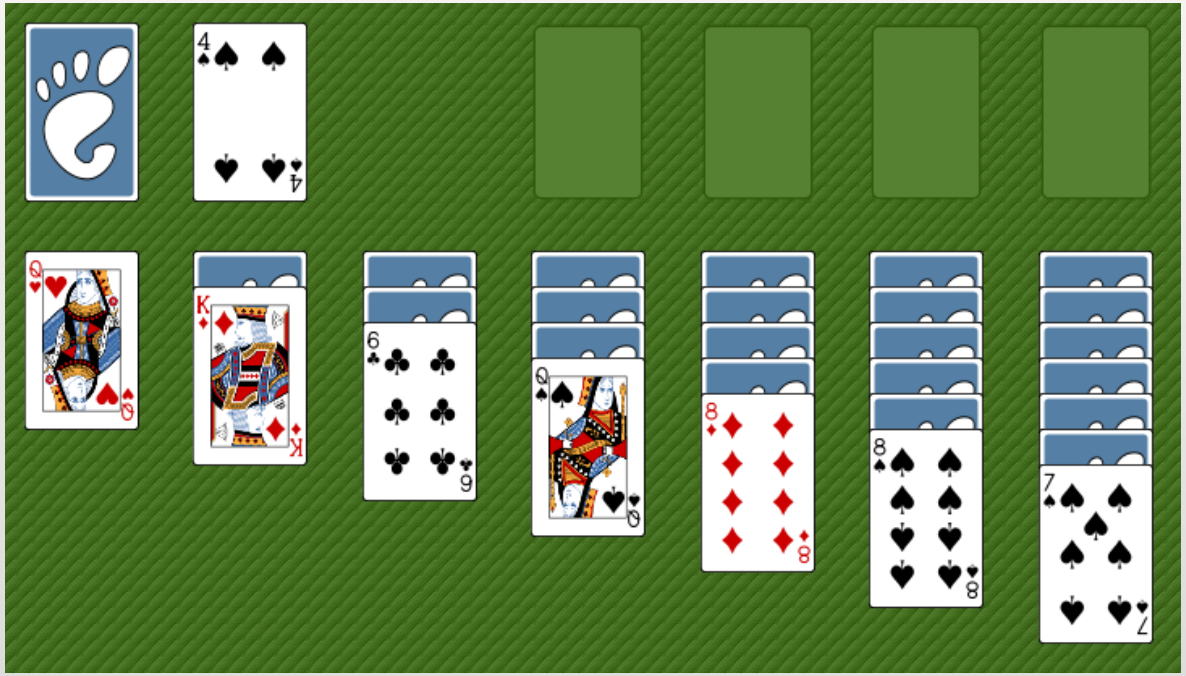
- Create a class called Country with attributes:
  - Population, size, national language
- Include a method called Greet, which outputs “Hello!”
- In main(), create objects from the country class called: UK, France, Spain
- Use constructors to initialise the attributes mentioned above
- Print all attributes and run Greet
- Bonus: inside Greet include an ‘if’ statement that changes language based on the national language

Monte Carlo Methods

- History, general concepts
- Quick example
- Other examples

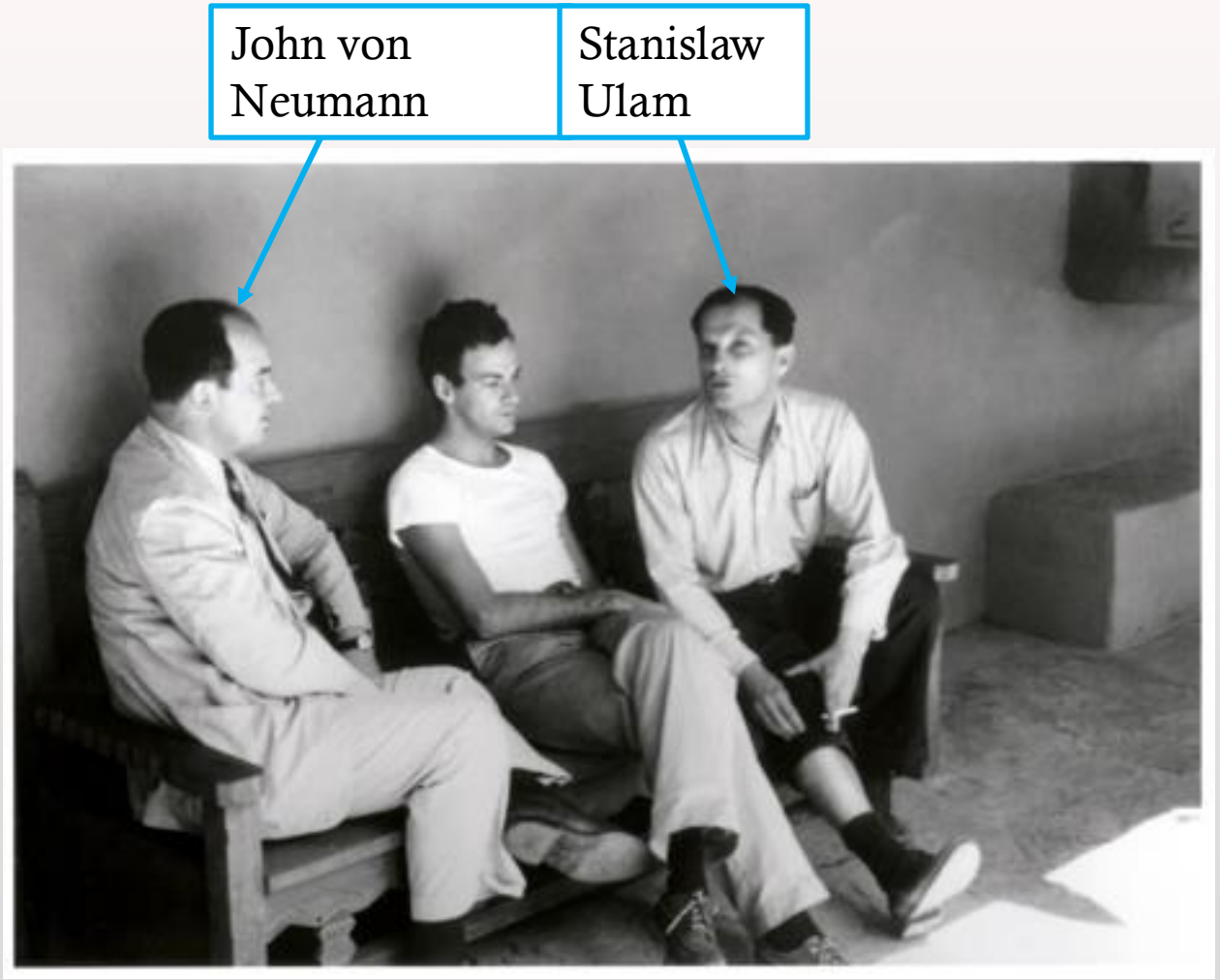
Monte Carlo History

- Initially created by Stanislaw Ulam, Polish-American scientist
- While ill, he wanted to know the probability of winning a game of solitaire
- Finding the pure calculations too hard, he adopted a statistical approach
- Playing the game would take too long, so he asked von Neumann to simulate it on an early (huge) computer



Monte Carlo History

- The Monte Carlo Method is a mathematical technique used to estimate the outcomes of an uncertain event
- Developed during WWII in relation to the Manhattan project
- Named after the Monte Carlo Casino, frequented by Ulam's uncle, as chance is important to the modelling approach
- Idea: use random sampling of inputs to explore outputs complex systems



Monte Carlo History

- The challenge of constructing the atom bomb involved **neutron diffusion**
- Too challenging to be addressed analytically, needed a numerical approach
- Some of the first computers tried an exhaustive numerical approach (plug in many numbers into the equation and calculate the result), but this was too slow due to high dimensionality of the problem
- Monte Carlo methods were found to be remarkably successful

MANIAC Computer

Monte Carlo Basics

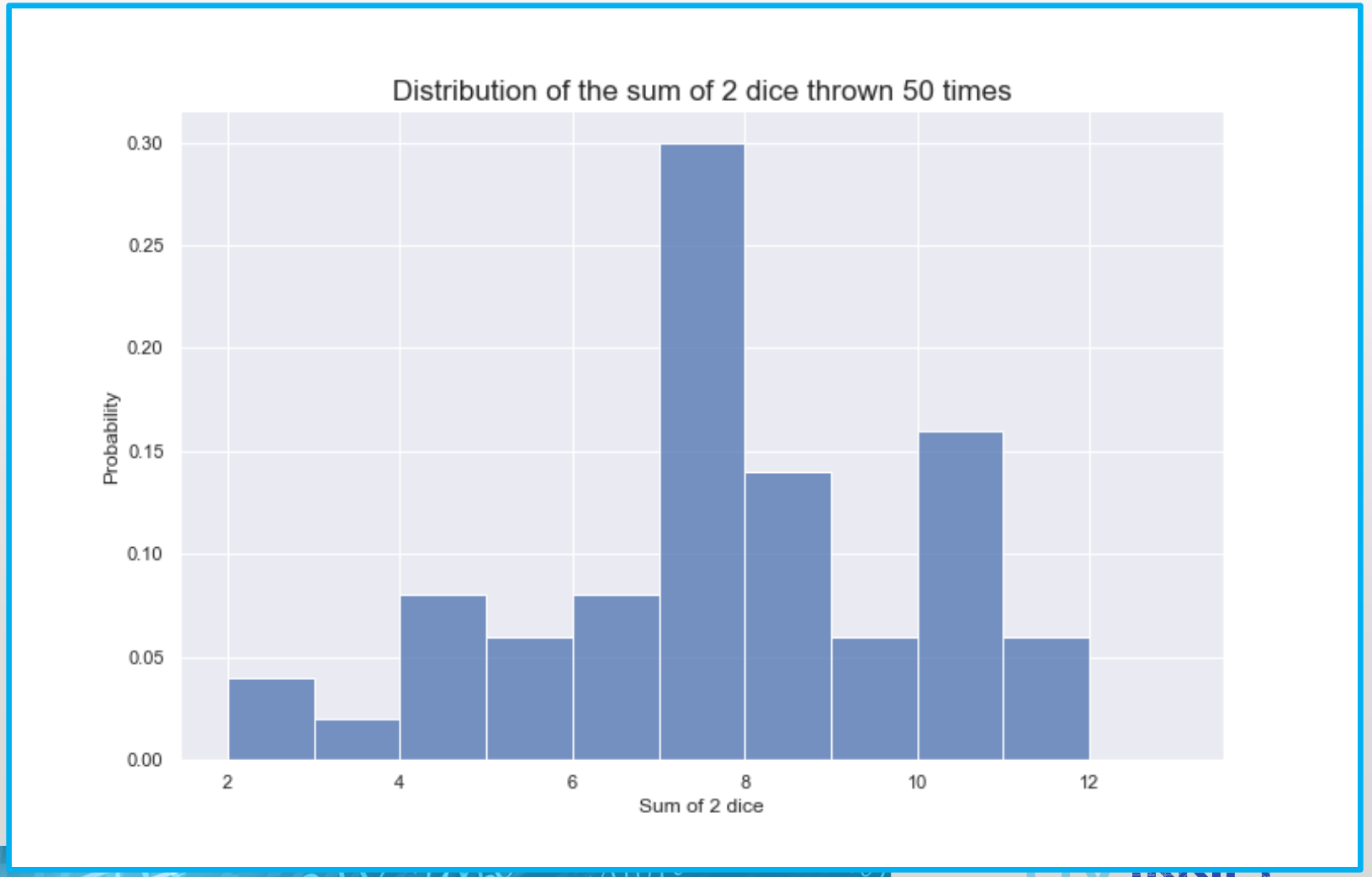
- A method of estimating the value of an unknown quantity using the principles of inferential statistics
- Key concepts:
  - Population: the universe of possible examples
  - Sample: a proper subset of a population
  - A random sample tends to exhibit the same properties as the population from which it is drawn
  - Use the sample to infer the statistics of the population
  - As the variance of the population increases, the more samples are needed to reach the same degree of confidence

Law of Large Numbers

- Law of large numbers:
  - In repeated independent tests with the same probability, *p*, of a particular outcome in each test, the chance that the fraction of times that the outcome occurs differs from *p* converges to zero as the number of trials goes to infinity

Law of large numbers

- Example: roll two dice to predict probability of getting a given number in total



Next week

- Monte Carlo with Classes