October 2023

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

Resources

- alex-hill94.github.io/#WS4
- https://www.w3schools.com/cpp/cpp_classes.asp
- https://library.lanl.gov/cgi-bin/getfile?00326866.pdf

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

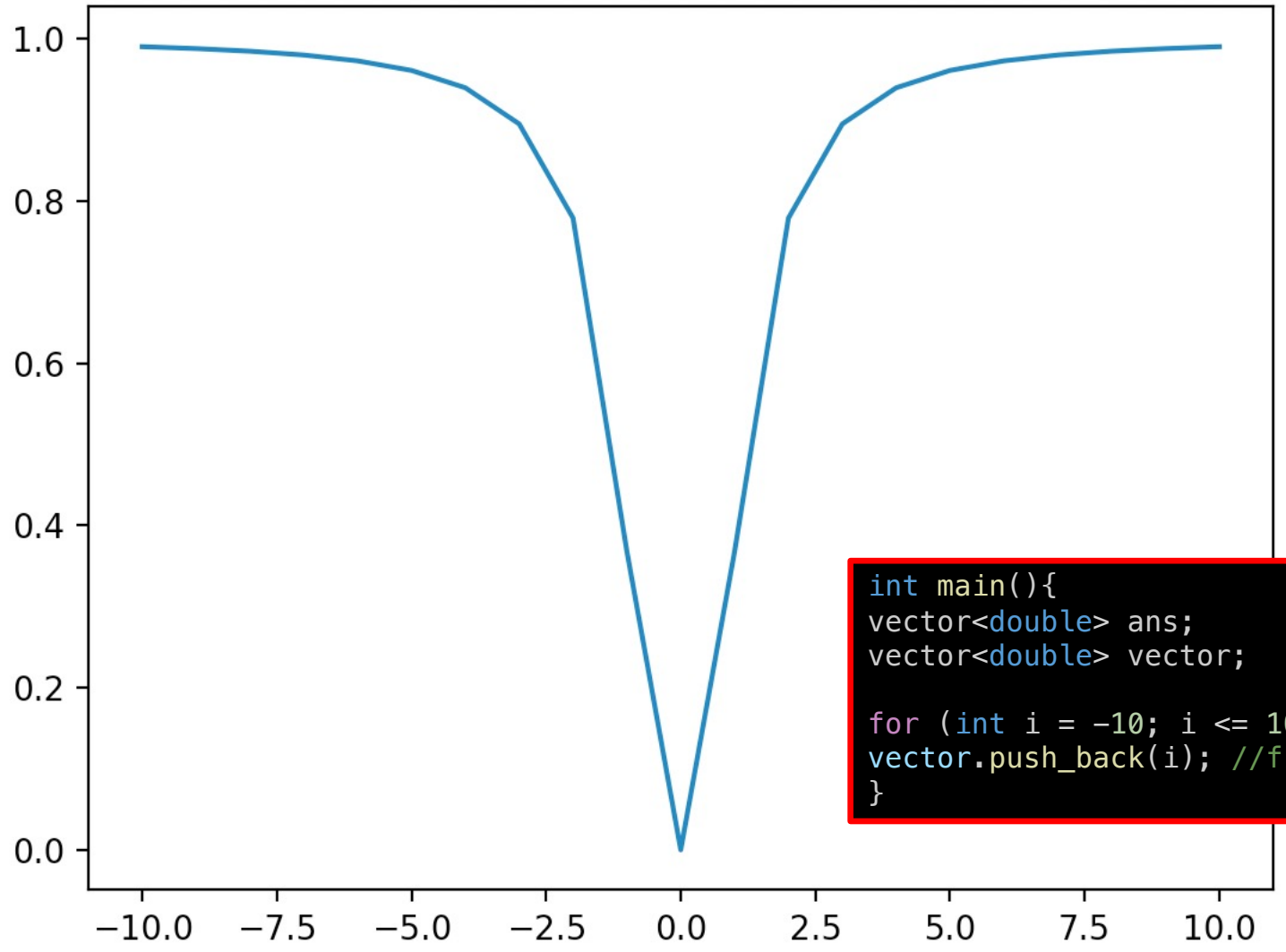
Emily

```cpp
//pointer for last element of ans
double* lastElement = &ans.back();
cout << lastElement << endl;
//writing input and output to python file
ofstream myfile;
myfile.open ("data.py");

myfile << "import numpy as np" << "\n";
myfile << "x = np.array((" << "\n";

for (double j: vector){
if (j != vector.back()){
myfile << j << ", " << "\n";
}
else{
myfile << j << "\n";
}
}

myfile << "))"<< "\n" << "\n";
myfile << "funcx = np.array((" << "\n";

for (double j: ans){
if (j != ans.at(*lastElement)){ //selects last value of ans
myfile << j << ", " << "\n";
}
else{
myfile << j << "\n";}
}
```

```
  File "/Users/alexhill/Documents/UOL/LIVINNO/Teaching/C++_Workshops/2023/WS4/scripts/emily/data.py", line 28
    0.98773,
    ^
SyntaxError: invalid syntax
```

```
10
))

funcx = np.array((
0.99005
0.98773,
0.984496,
0.979799,
0.972604,
0.960789,
0.939413,
0.894839,
0.778801,
0.367879,
0,
0.367879,
```

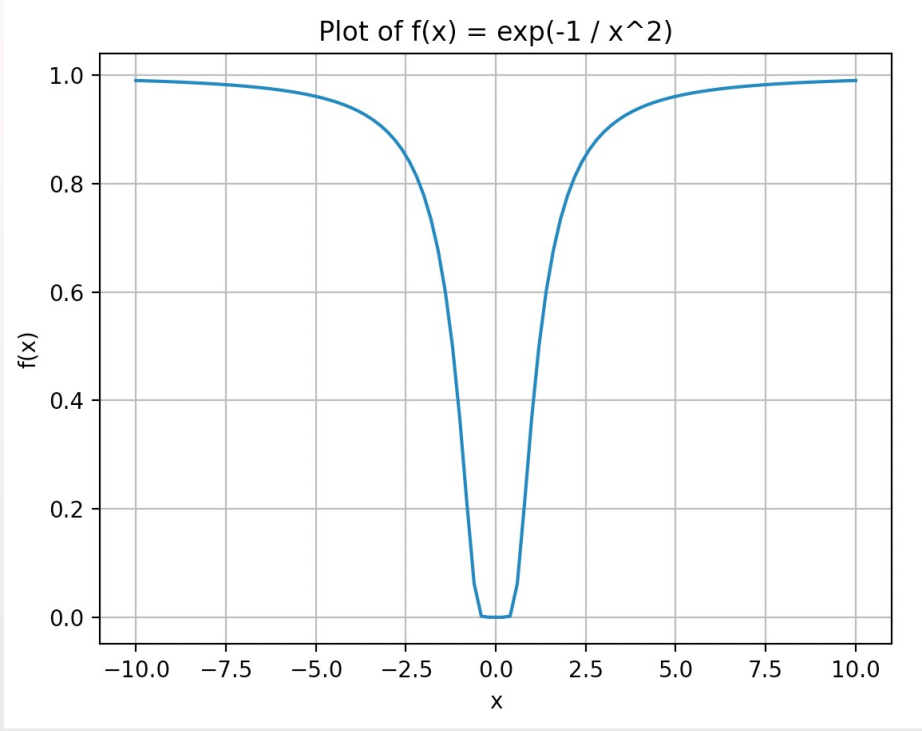
Emily




```cpp
int main(){
vector<double> ans;
vector<double> vector;

for (int i = -10; i <= 10; i++) {
vector.push_back(i); //fills empty vector with values between -10 and 10
}
```

Sam

```
double func(double x) {
if (x == 0) return 0;
return exp(-1.0 / (x * x));
}


int main() {
vector<double> thetas;
vector<double> ans;
cout << "What would you like the length of
the vector to be?\n";
int length;
cin >> length;
double stepSize = 20.0 / length;
for (double x = -10; x <= 10; x +=
stepSize) {
thetas.push_back(x);
ans.push_back(func(x));
}
```

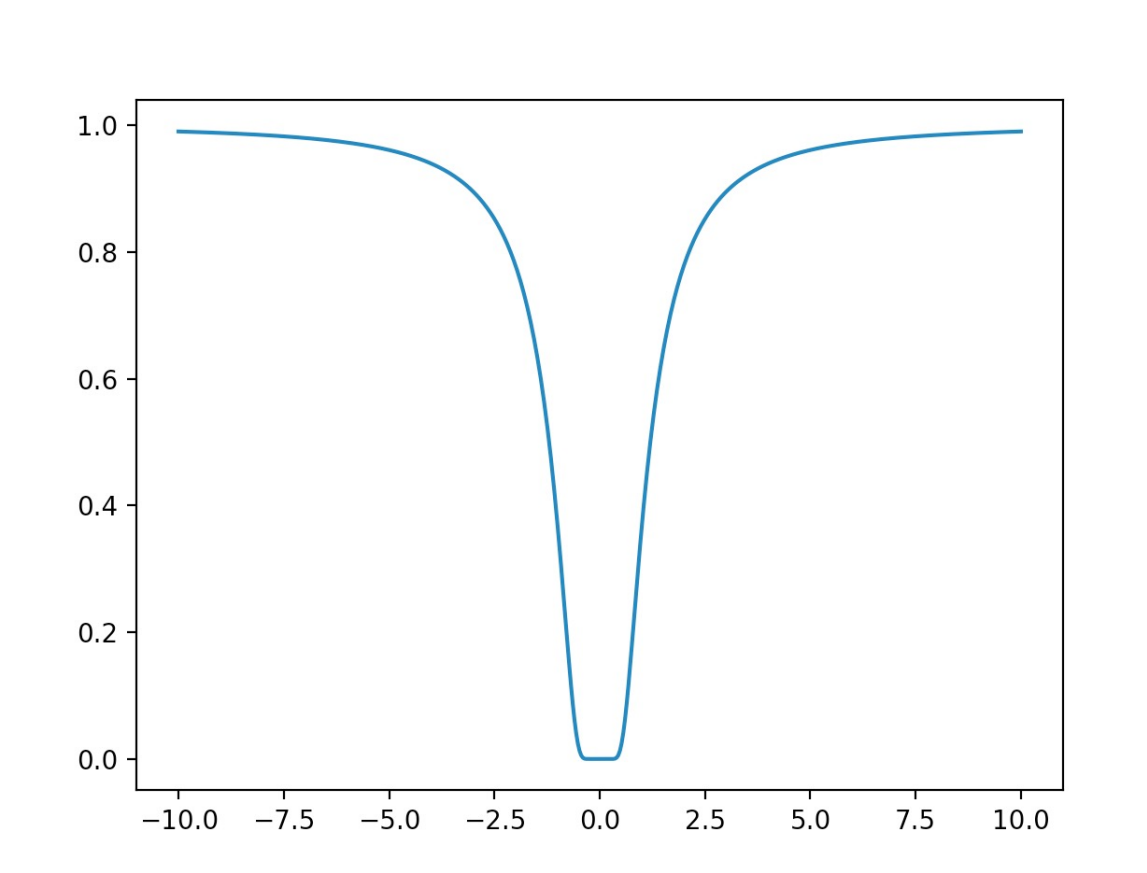


What would you like the length of the vector to be?
100

In [2]: len(theta)
Out[2]: 101

Ana

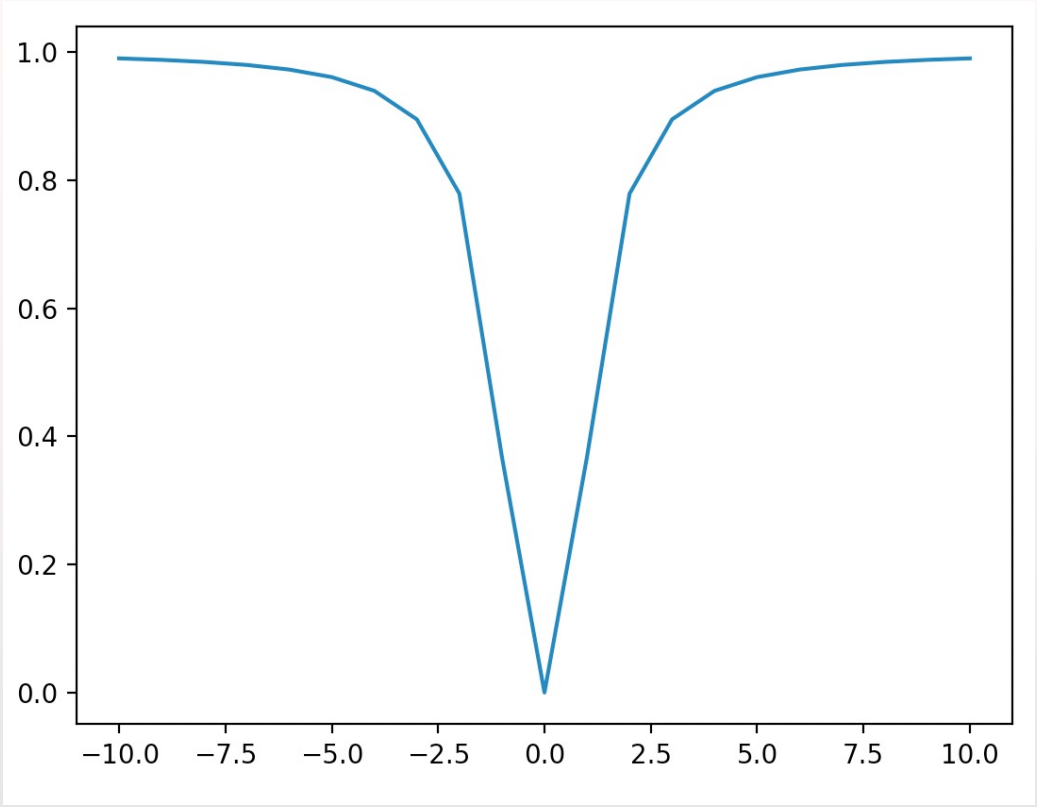
```
int main(){
double k=500;
//cout << "Input n_vals:";
//cin >> k;
vector<double> input;
double i = 20/k;
int b = 10;
for (double a=-10; a<=b ; a=a+i)
{
input.push_back(a);
}
vector<double> output;
func(input, output);
string myfile;
write_out("myfile", input, output);
```




```
void write_out(string outfile, vector
<double>& vect, vector <double>& vectf)
```

Marina

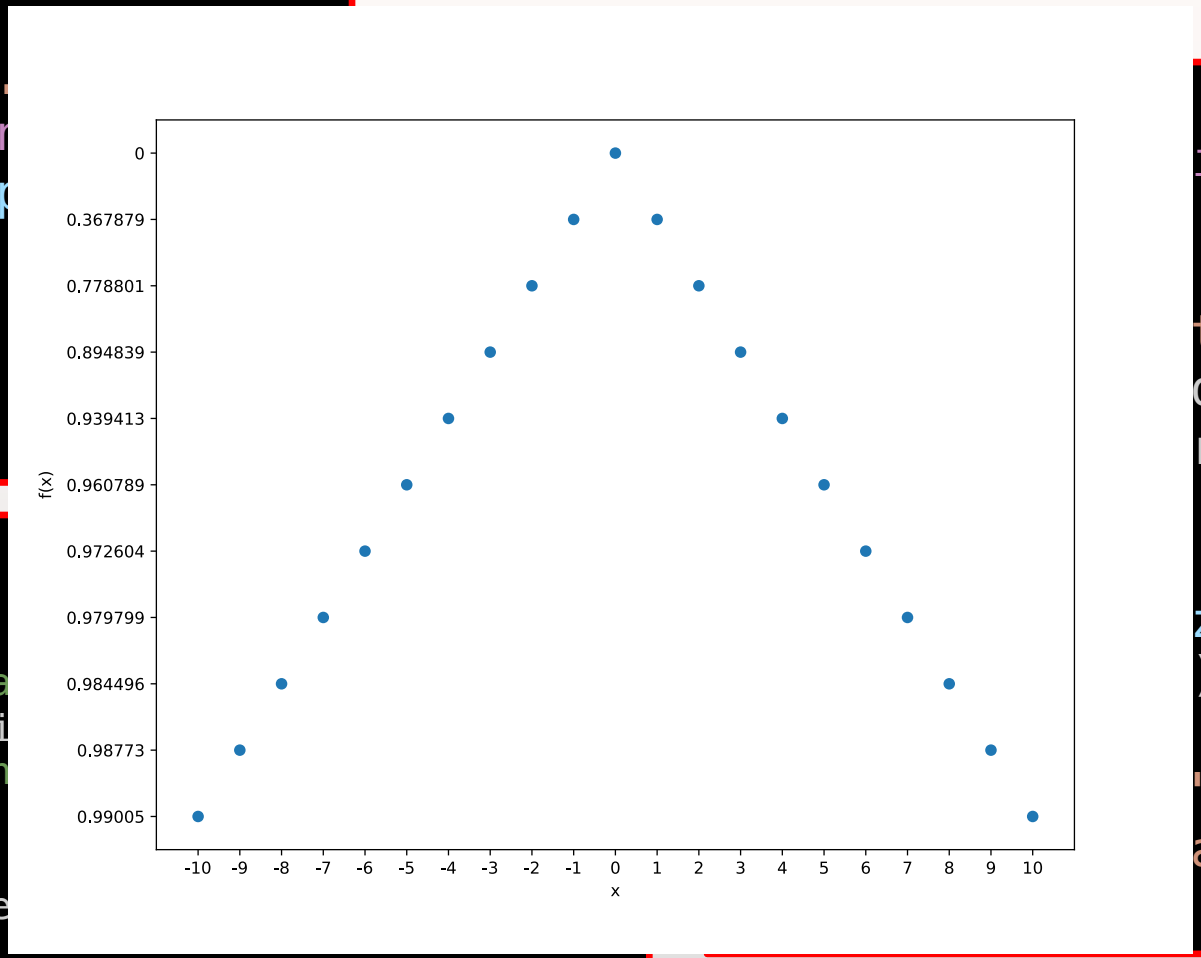
```
int write_out(string filename,
vector<int>& x, vector<double>& f){
std::ofstream myfile;
myfile.open ("data.py");
…
…
…
cout << "Wrote to file " << filename <<
endl;
```



Andrew

```
cd "$(dirname "$0")"
output=$(g++ ./andrew.
if [[ $? != 0 ]]; ther
echo -e "Error:\n$outp
else
./a.out
python plot.py
fi
```

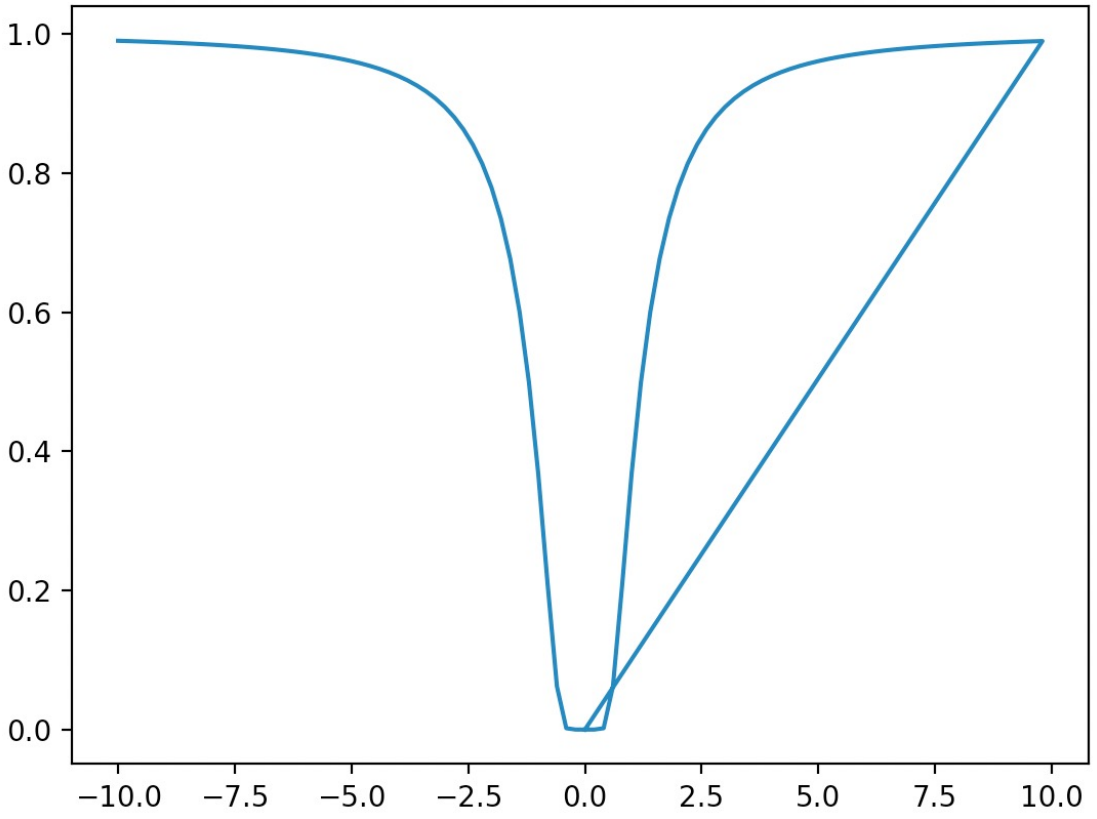
```
int main()
{
vector<float> vec, result;
// generate a vector with ra
for (int i = -10; i <= 10; i
// calculate results for fun
func(vec, result);
// write results into file
wrtie_out("my_data", vec, re
return 0;
}
```




```
import pyplot as plt


ta.csv') as f:
der(f, delimiter=',')
r:


ze=(10,8))
)

")
a_plot.pdf')
```

Luke

```
vector<double>x(n+1); //initialise x
vector
vector<double>y(n+1); //initialise y
vector
for (int i=0; i<n; ++i){
x.at(i)= -10 + 20*i/n;
}
…
…
…
{if (j != x.back())
{myfile << j << ", " << "\n";}
```




```
^
SyntaxError: invalid syntax
```

Sakrican

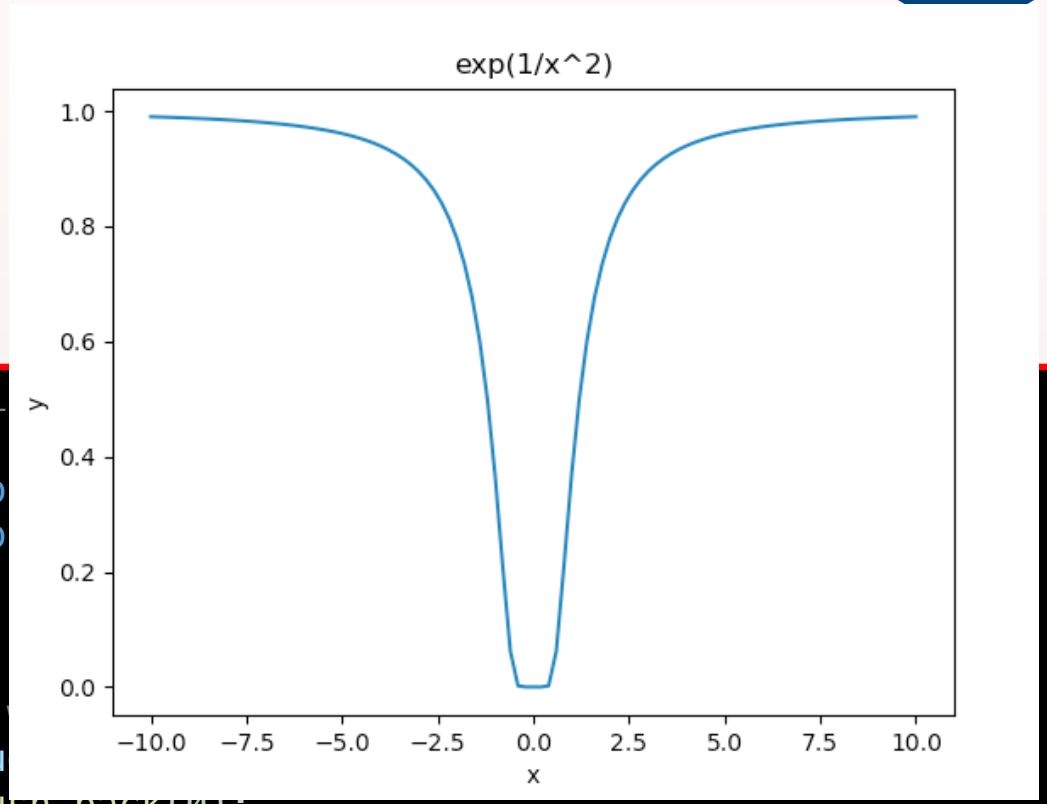
```
vector<double> vector1;


cout << "Evenly spaced vector between -10
and 10 " << endl;


for(int i=0; i < vector_size ; ++i) {
vector1.at(i) = (-10 + (i * step_size));
cout << i << " " << vector1.at(i) << endl;
}
```

```
Input Vector =
-10,
-8,
-6,
-4,
-2,
0,
2,
4,
6,
8,
10


Output =
0.99005
0.984496,
0.972604,
0.939413,
0.778801,
0,
0.778801,
0.939413,
0.972604,
0.984496,
0.99005
```

```
59 -10 0.99005
60 -10 0.99005
61 -10 0.99005
62 -10 0.99005
63 -10 0.99005
64 -10 0.99005
65 -10 0.99005
66 -10 0.99005
67 -10 0.99005
68 -10 0.99005
69 -10 0.99005
70 -10 0.99005
71 -10 0.99005
72 -10 0.99005
73 -10 0.99005
74 -10 0.99005
75 -10 0.99005
76 -10 0.99005
77 -10 0.99005
78 -10 0.99005
79 -10 0.99005
80 -10 0.99005
```

Joe

```cpp
int main() {

// Create input vector with arbitrary number of points
vector<double> xValues = linspace(-10, 10, 101);

// Use function to return y values
vector<double> yValues = func(xValues);

// Write out both using write_out function
write_out("input.dat", xValues);
write_out("output.dat", yValues);

return 0;
}
```

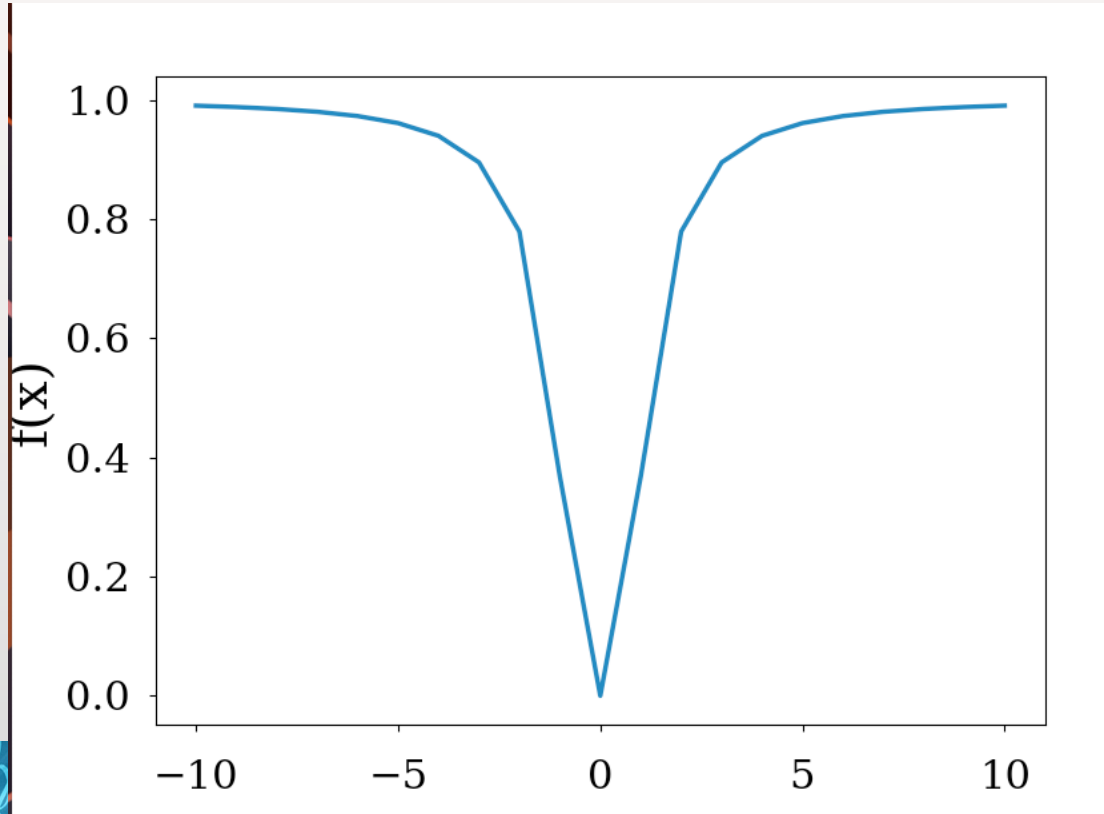



```cpp
// Functi
is 0
vector<do
vector<do

for (int

// Check
if (xValu
output.push_back(0);
} else{
output.push_back(exp(-1/pow(xValues[i], 2)));
}
}
return output;
}
```

LIV.INNO

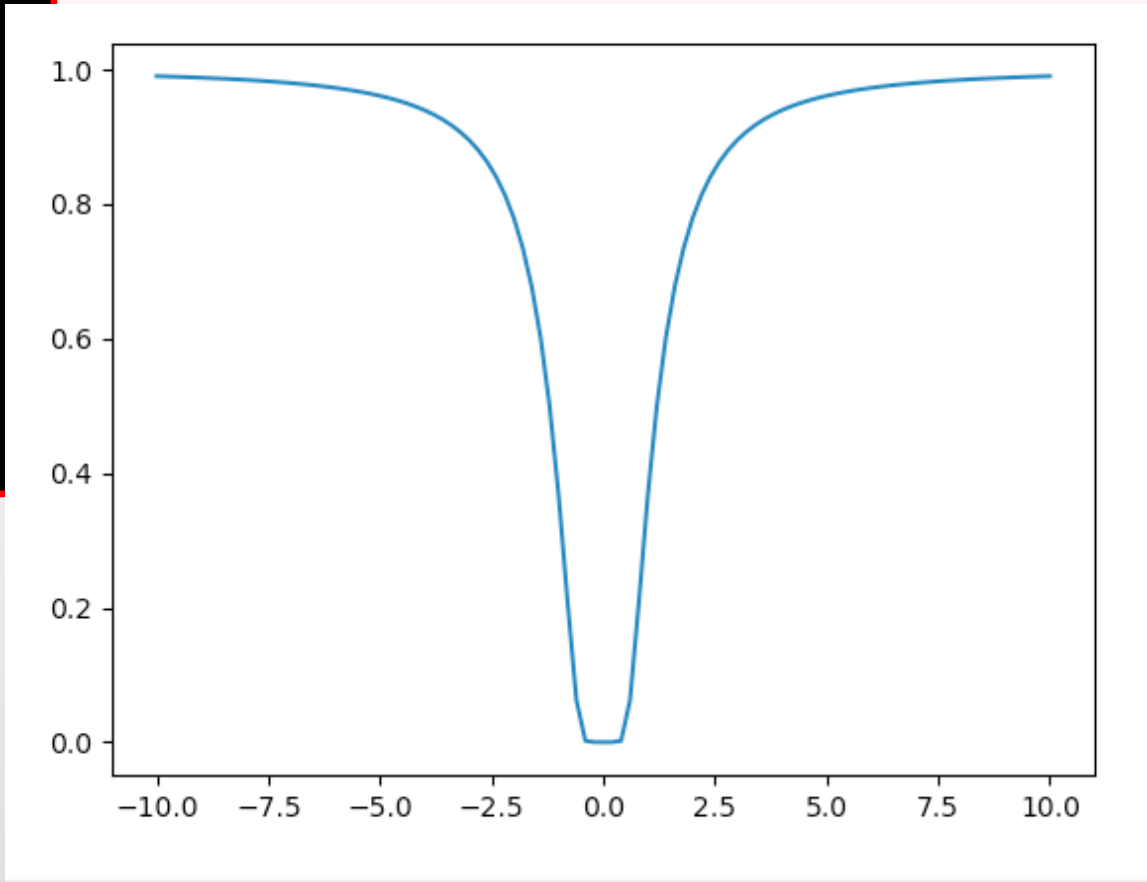
Khang

```
int main()
{
// vector of range -10 to 10
vector<double> vect_inp{-10,-9,-8,-7,-6,-5,-4,-3,-2,-
1,0,1,2,3,4,5,6,7,8,9,10};
vector<double> vect_out;

//passing vector into function
func(vect_inp, vect_out);

// write-to-file function
write_out(vect_inp, vect_out);

return 0;
}
```



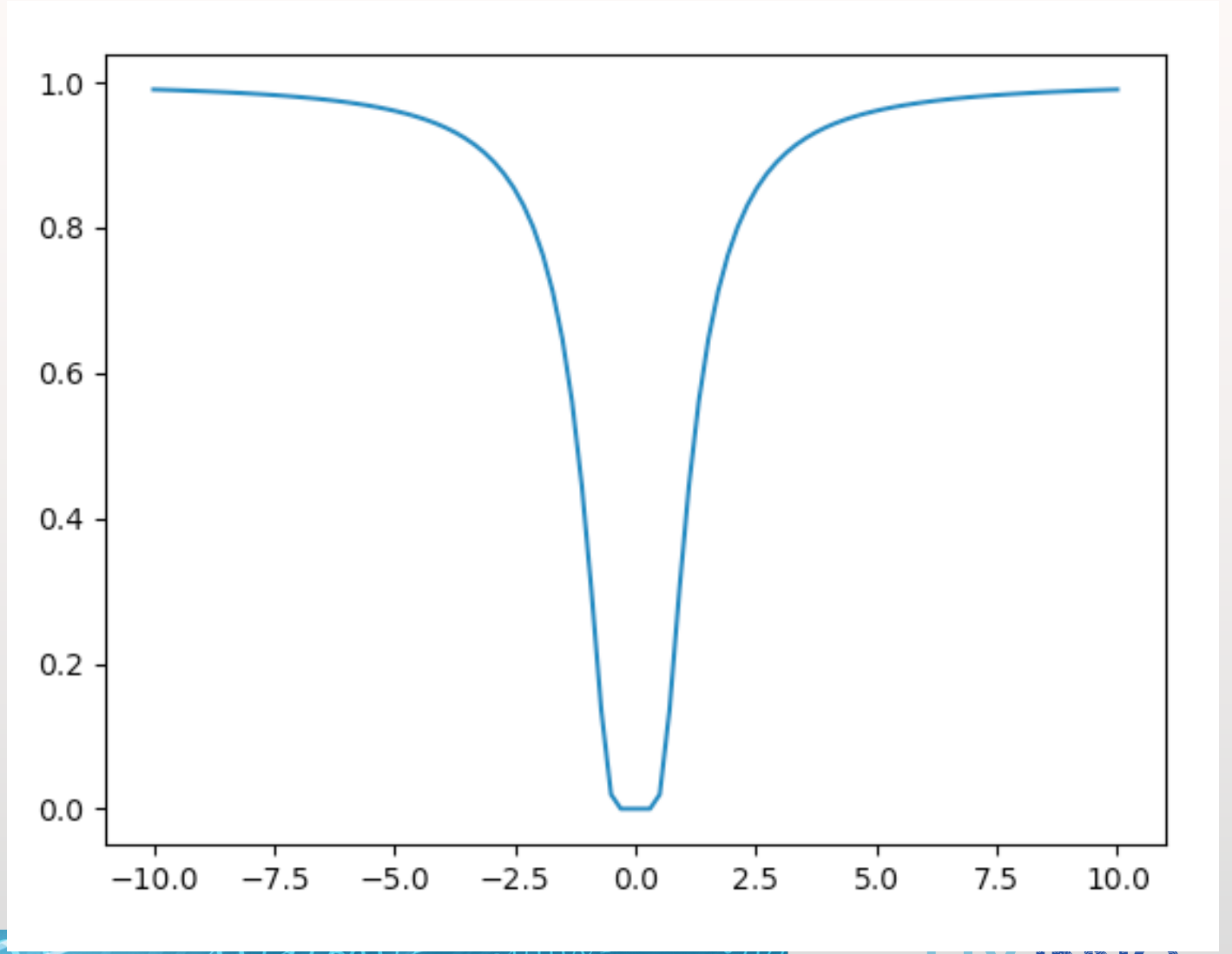
Mehul

```
double func(double x){
double f;
if (x == 0){ f ==0;}
else {
double poly =(1/(x*x));
f = exp(-poly);
}
return f;
}
```



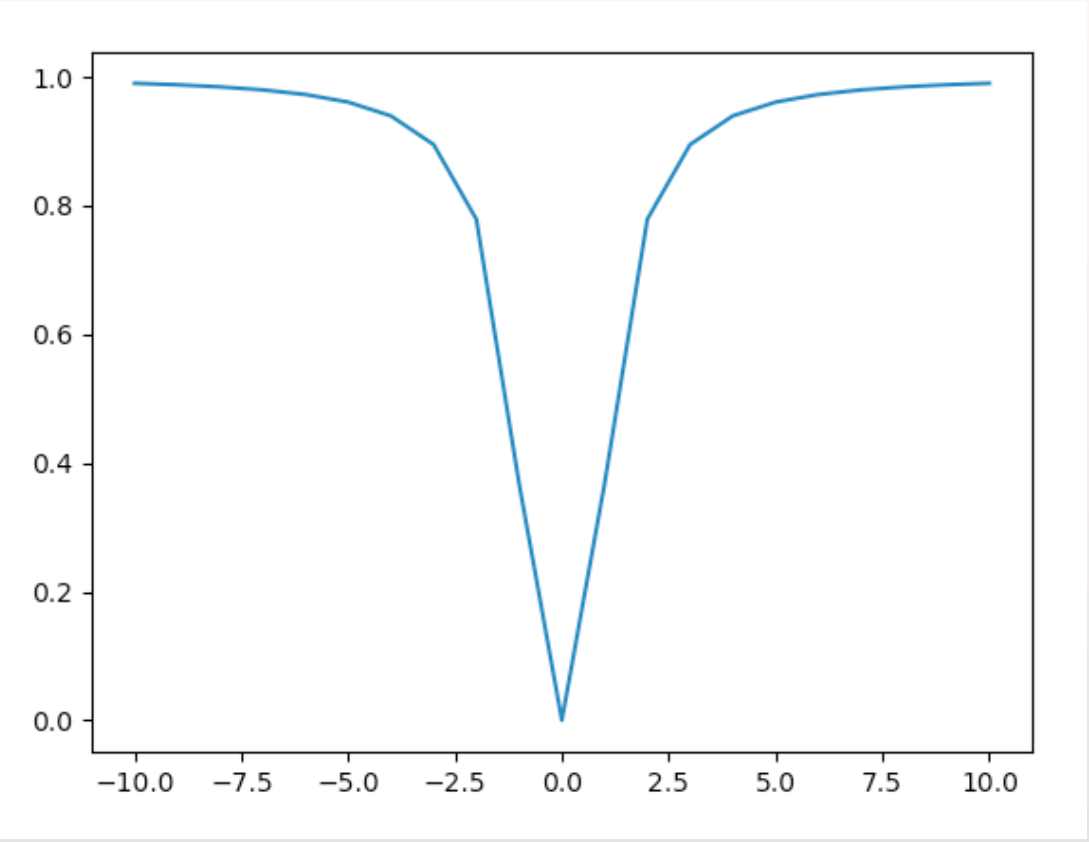
Rupesh

```
Ans = = np.array((
0.99005
0.989637,
0.989199,
0.988732,
0.988234,
0.987703,
```



Sinead

```
int main() {
vector<double> v1 = {-10, -9, -8, -7, -6, -
5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7,
8, 9, 10};
vector<double> v2(21, 0.0); // Initialize v2
with zeros and the same size as v1

for (int j = 0; j < 22; j++) {
v2[j] = func(v1[j]);
}

for (int j = 0; j < 22; j++) {
cout << v2[j] << " ";
}
```



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

```
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;
}
```

Create input data

Compute function

Write vectors to file

Alex

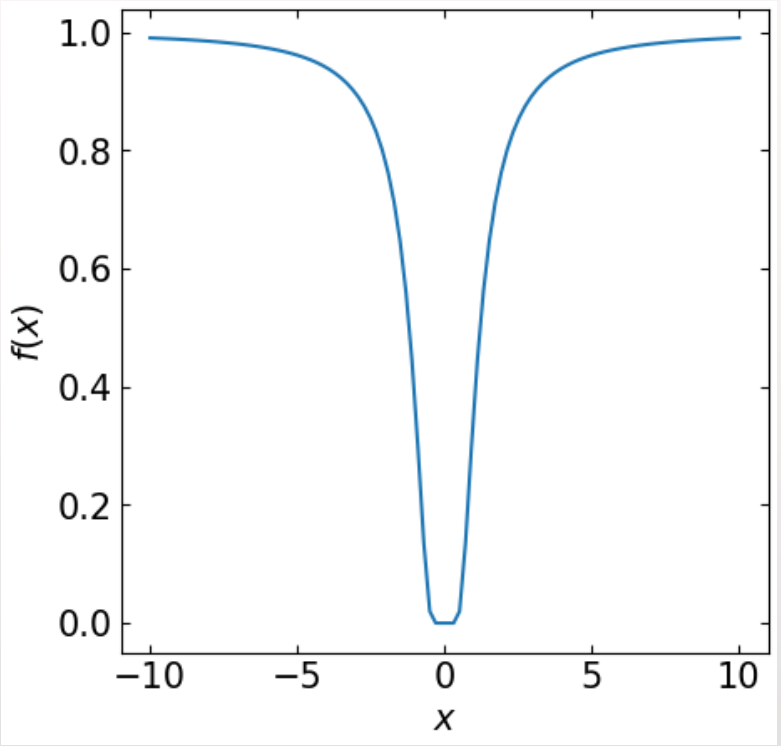
```
void write_out_vec(const string& filename, string& array_name , const vector<double>& output,
bool initialise = true)
// Writes out data in numpy form. If initialise == true, creates file from scratch and includes
top lines needed
{
ofstream myfile;
if (initialise){
myfile.open (filename);
myfile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

```python
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
- Careful with how you identify the last element in an array

Takeaways

- Try to make your code easily adaptable
- Try editing vectors inside functions, rather than returning individual values
- Fix the values of your vectors using const to ensure no mistakes
- Try writing python inside a C++ script!

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

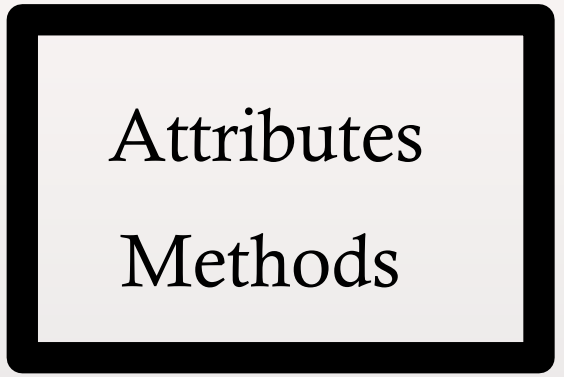
- Procedural programming specifies the specific steps a programme must take
- OOP creates objects that contain both data and functions
- Why? DRY! (Don't repeat yourself)

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

Creates a class called Car

Determines how visible members of class are to the outside

Creates some attributes

```cpp
#include <iostream>

using namespace std;

class Car { // The class
public: // Access specifier
int n_seats; // Attribute (int variable)
string brand; // Attribute (string variable)
string model; // Attribute (string variable)
};

int main() {
Car newcar; // Create an object of Car

// Access attributes and set values
newcar.n_seats = 5;
newcar.brand = "Seat";
newcar.model = "Ibiza";

// Print attribute values
cout << newcar.n_seats << "\n";
cout << newcar.brand << "\n";
cout << newcar.model << "\n";
return 0;
}
```

Creates object called newcar via the car class

Changes values of class 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

```cpp
#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

```cpp
#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

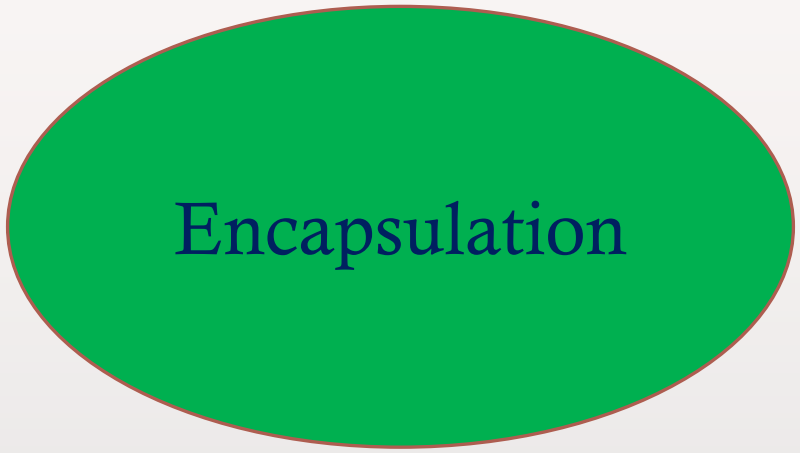
```
class Car {
public:
void Beep() {
cout << "Beep Beep" << endl;
}
```

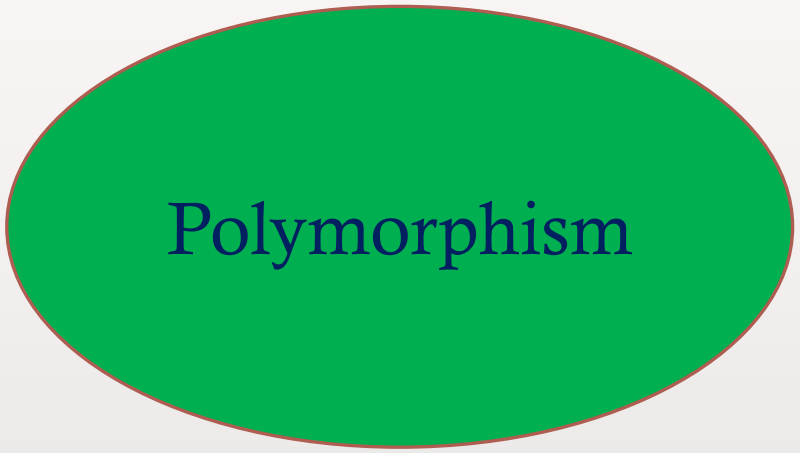
Access Specifiers

```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)
        ^
```

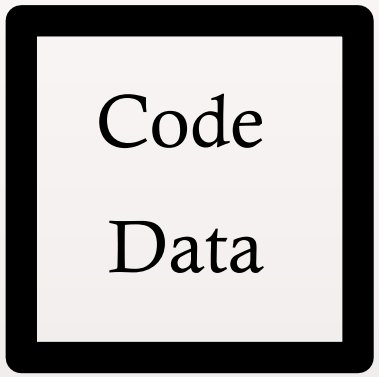
Principles of OOP





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

```cpp
#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:
int bonus;
void setSalary(int s) {
salary = s;
}
int getSalary() {
return salary;
}
}
```

'Protected' data can be accessed by the inherited class

Polymorphism

- Polymorphism allows us to use inherited methods to perform different tasks

```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

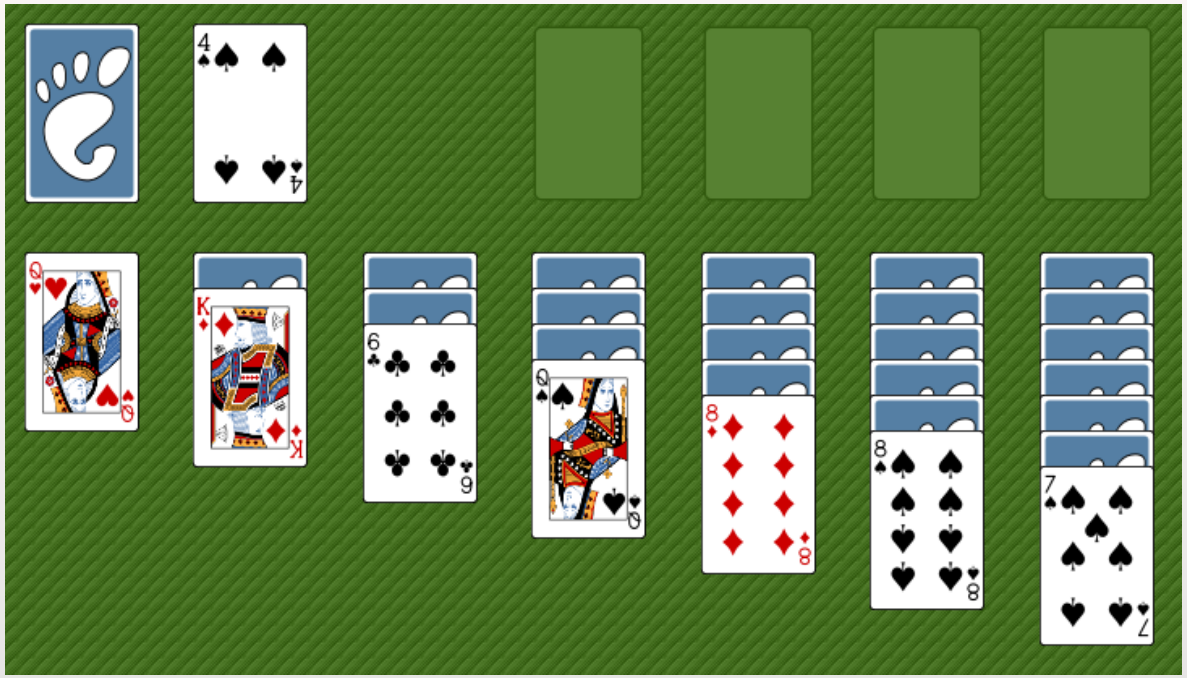
- Create a class called Country with attributes:
  - Population, size, national language
- Include a method called Greet, which outputs “Hello!”
- In main(), create objects from the country class called: UK, France, Spain
- Use constructors to initialise the attributes mentioned above
- Print all attributes 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
- Markov Chains

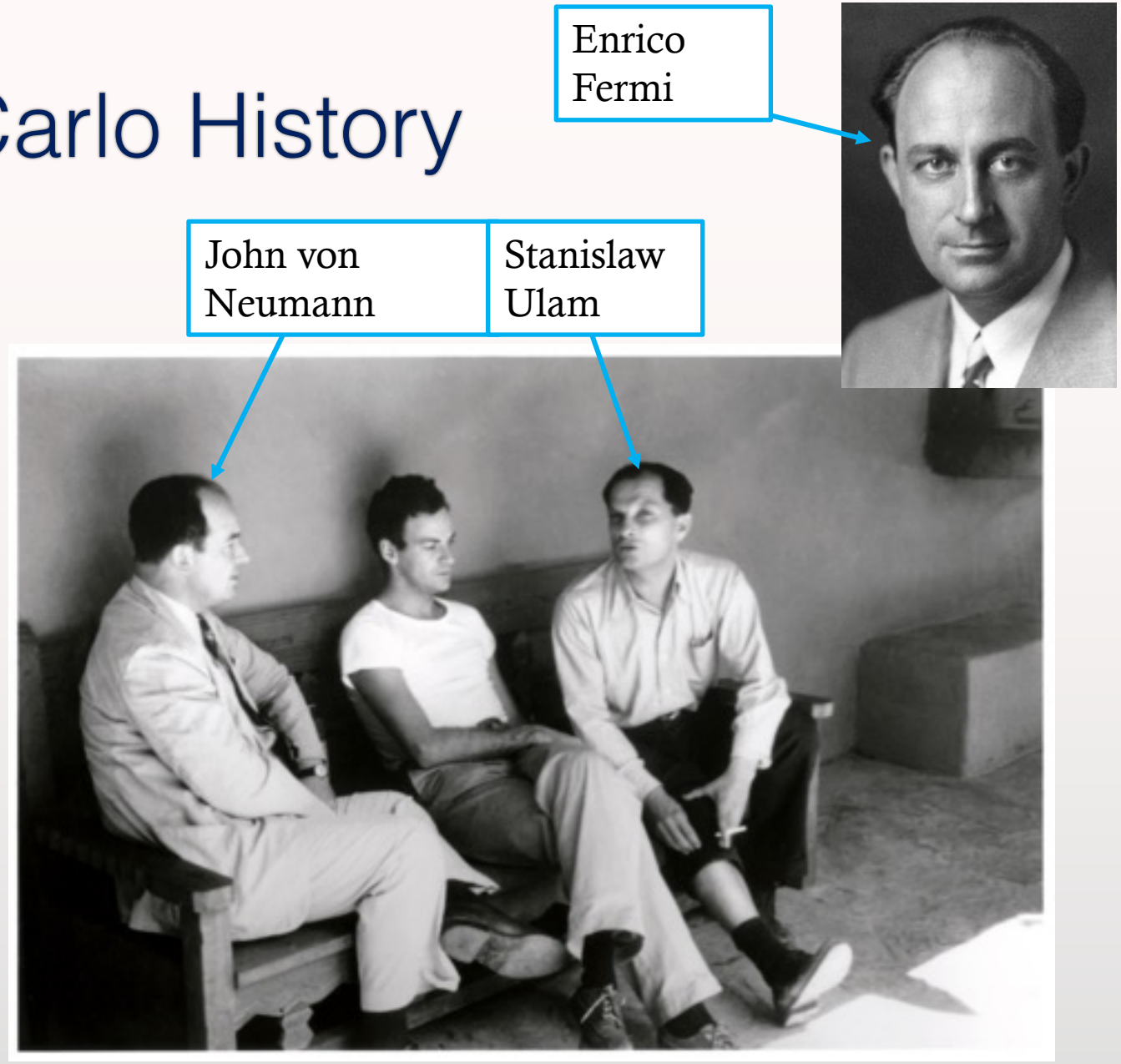
Monte Carlo History

- 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

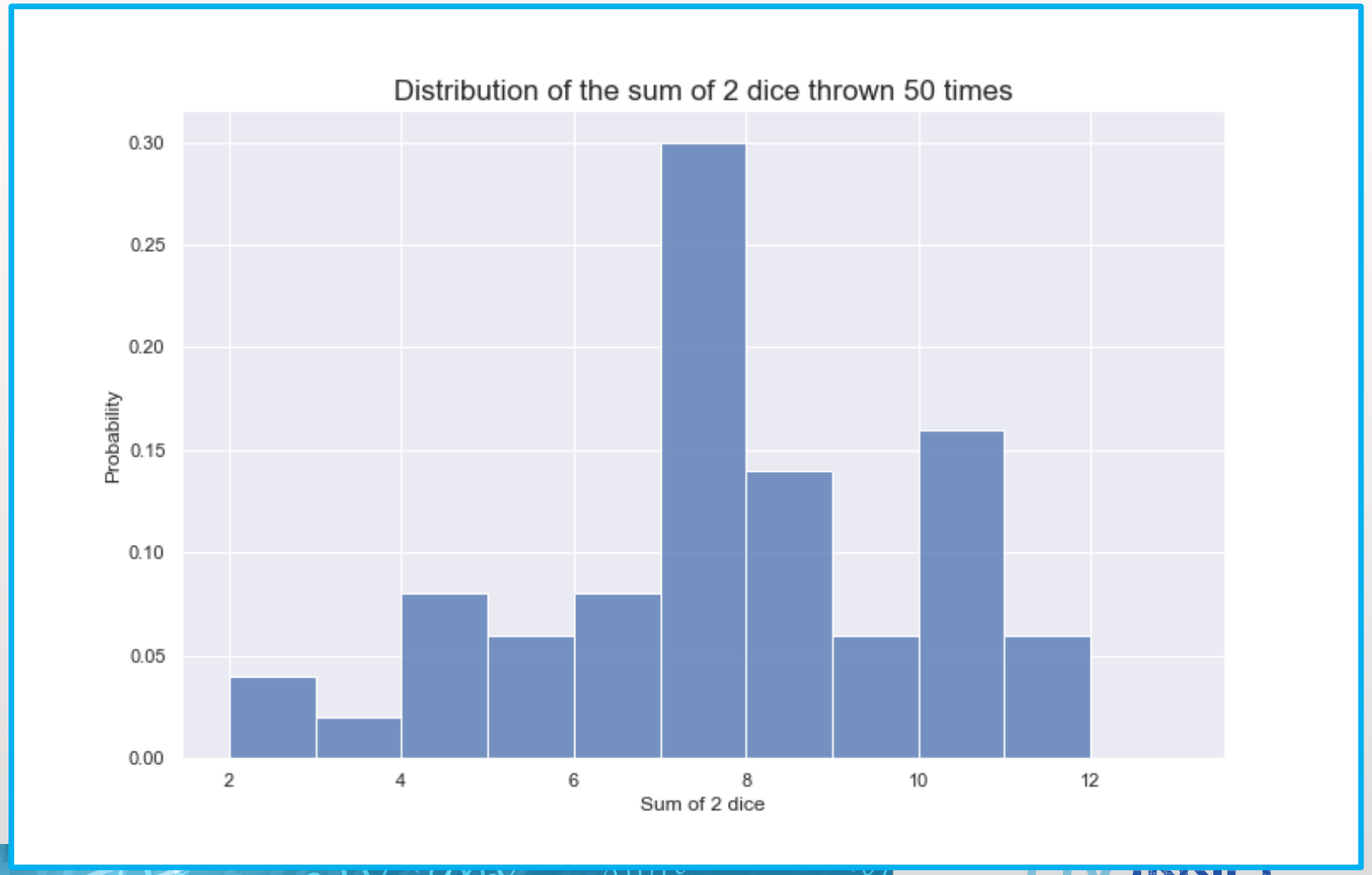
  - As the variance of the population increases, the more samples are needed to reach the same degree of confidence

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



Gambler's Fallacy

- This is the idea that following a statistically unlikely set of results (e.g. many instances of red in a row in roulette), people believe that the next sample will work to bring you back to the average
- Example in 1913 at Monte Carlo casino: after 15 blacks in a row, there was a rush to bet on red
- After 26 reds in a row, the casino had made a fortune

Regression to the Mean

- Following an extreme, independent random event, the next random event is likely to be less extreme
- Example: 10 reds in a row in roulette is $\frac{1}{2^{10}}$
- In the next 10 spins, it is likely that you will get fewer than 10 reds (E[red] = 5)
- For the 20 spins in total, you will get closer to the expected mean of 50% reds than you had in the first 10 spins
- Subtly different to the gambler's fallacy, in which you would expect to get more than five blacks to counteract the initial extreme event

Next week

- Monte Carlo with Classes