Lecture 2: Pythonic Movement

Bart Iver van Blokland
(Rune Sætre)

JOIN ME

We have cookies

The reference group needs you

It is your desssstiny

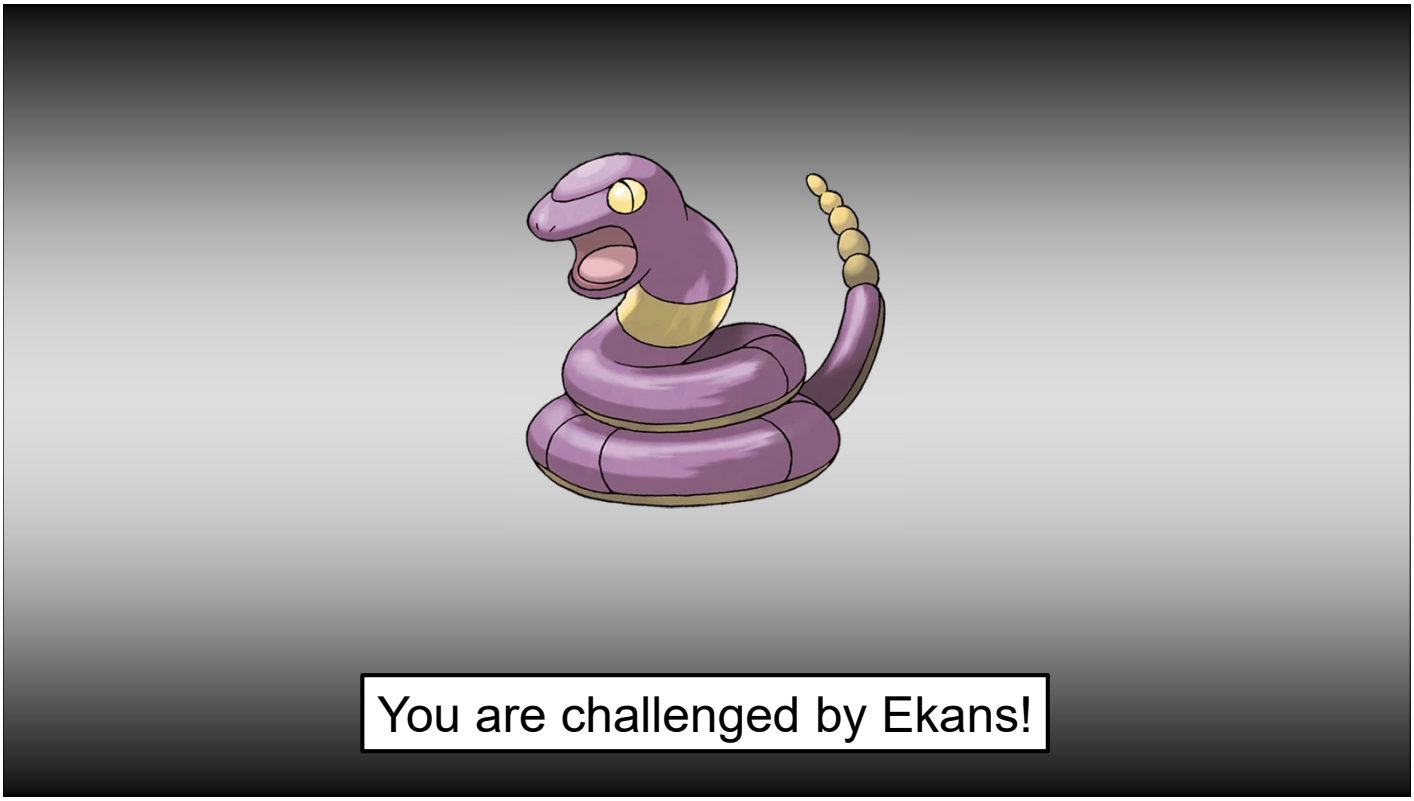
tdt4102-fagans@idi.ntnu.no

Vi hører gjerne hva dere synes om faget. Vi leter etter dere som kan ha møte med oss et par ganger gjennom semesteret, for å fortelle hva dere og deres medstudenter synes om faget!

Last time..

- Project configuration
- Program compilation
- The main function
- Writing to the terminal using cout
- Debugging

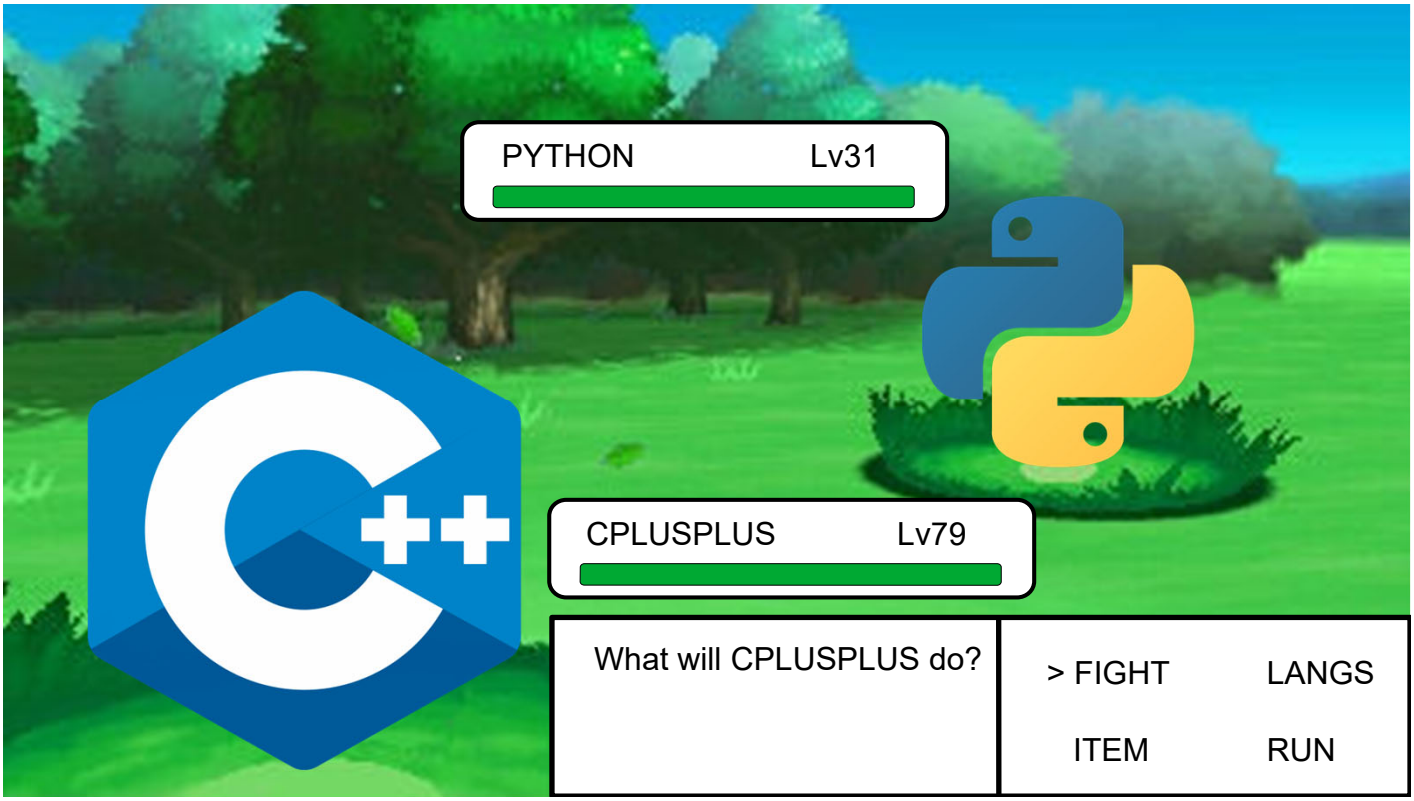
Dette ble behandlet på forrige forelesning.

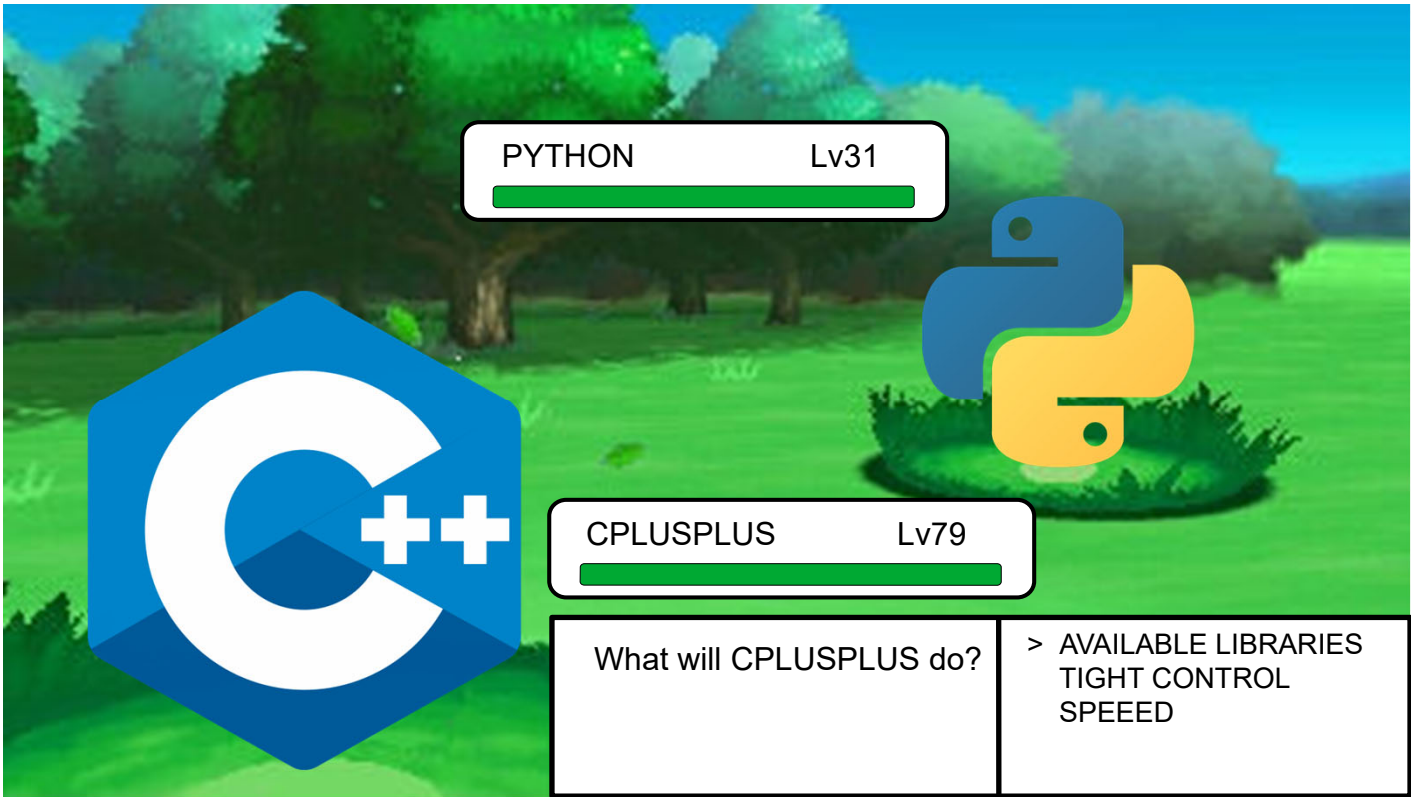


Å nei, deg igjen?

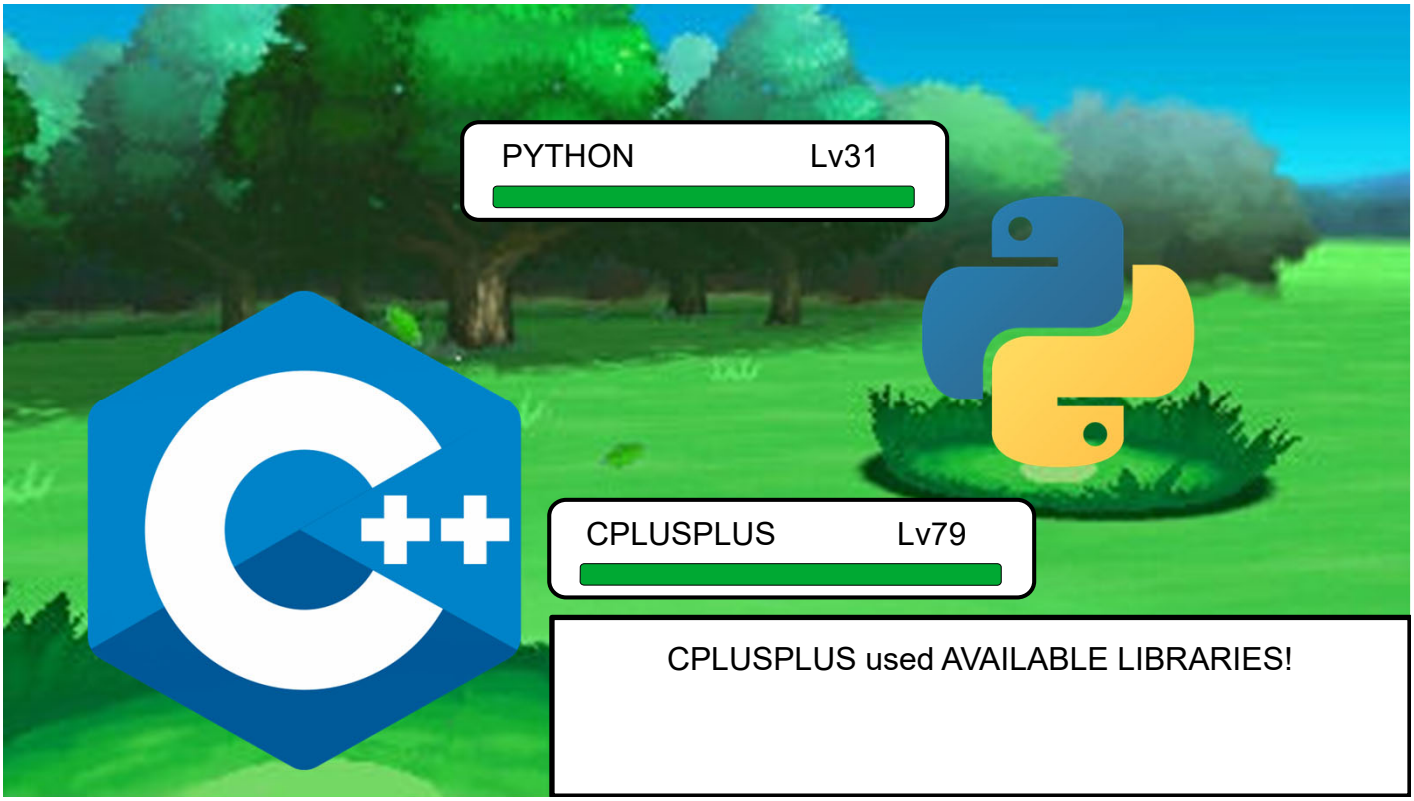
Ja da, den synes at Python er det aller beste programmeringsspråket som finnes.

La oss se hva C++ har å si om det..

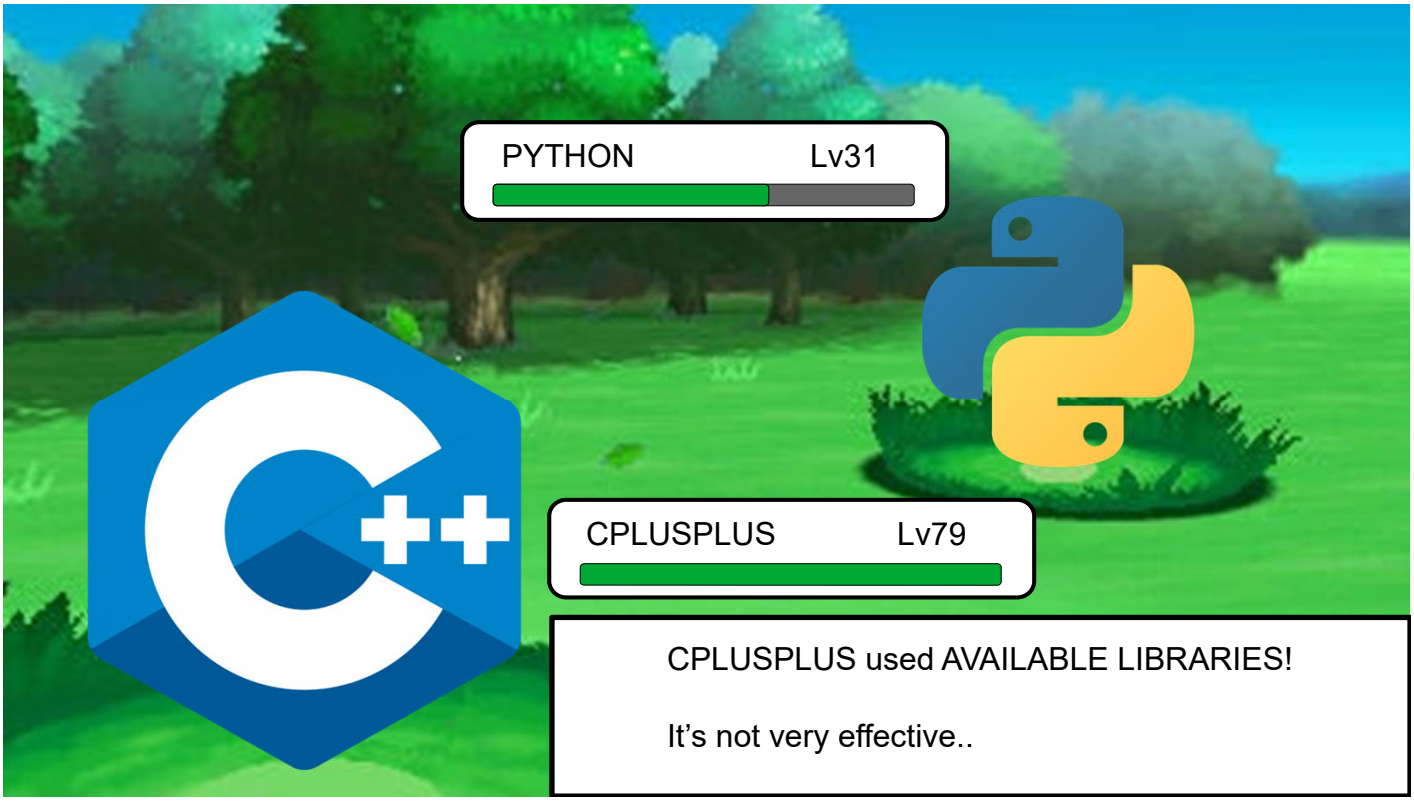

PYTHON Lv31
CPLUSPLUS Lv79
What will CPLUSPLUS do?
> FIGHT LANGS
ITEM RUN

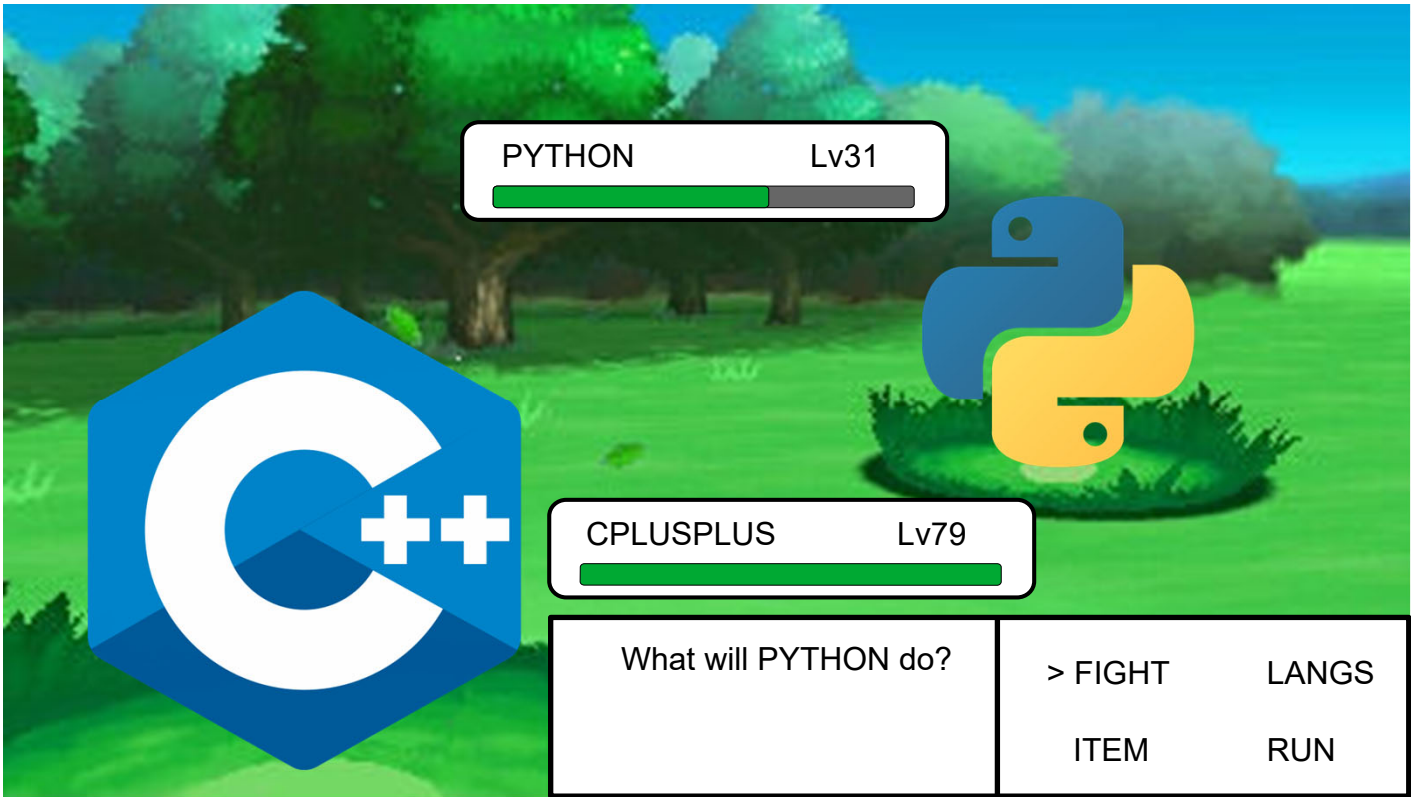


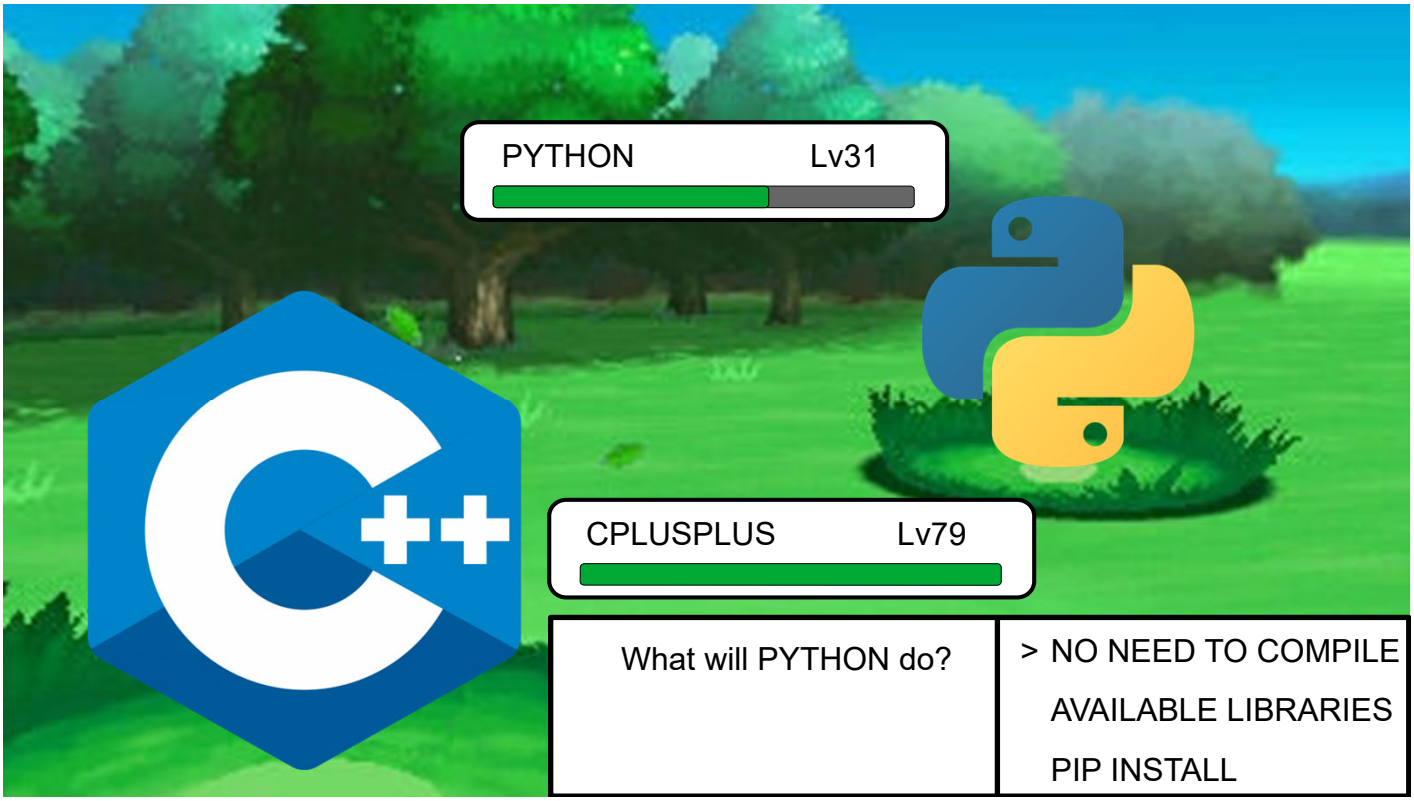
Vi har forskjellige muligheter her. Som i C har du stor kontroll over hvordan et program skal fungere. Det gir deg stor kontroll over ressurser som dataminne og prosessoren. Det er ikke mange språk som gir deg slik kontroll. I tillegg finnes det masse biblioteker som andre har skrevet for å gjøre alt mulig.



Dessverre har Python også en god del biblioteker tilgjengelig, så det var ikke en god argument.

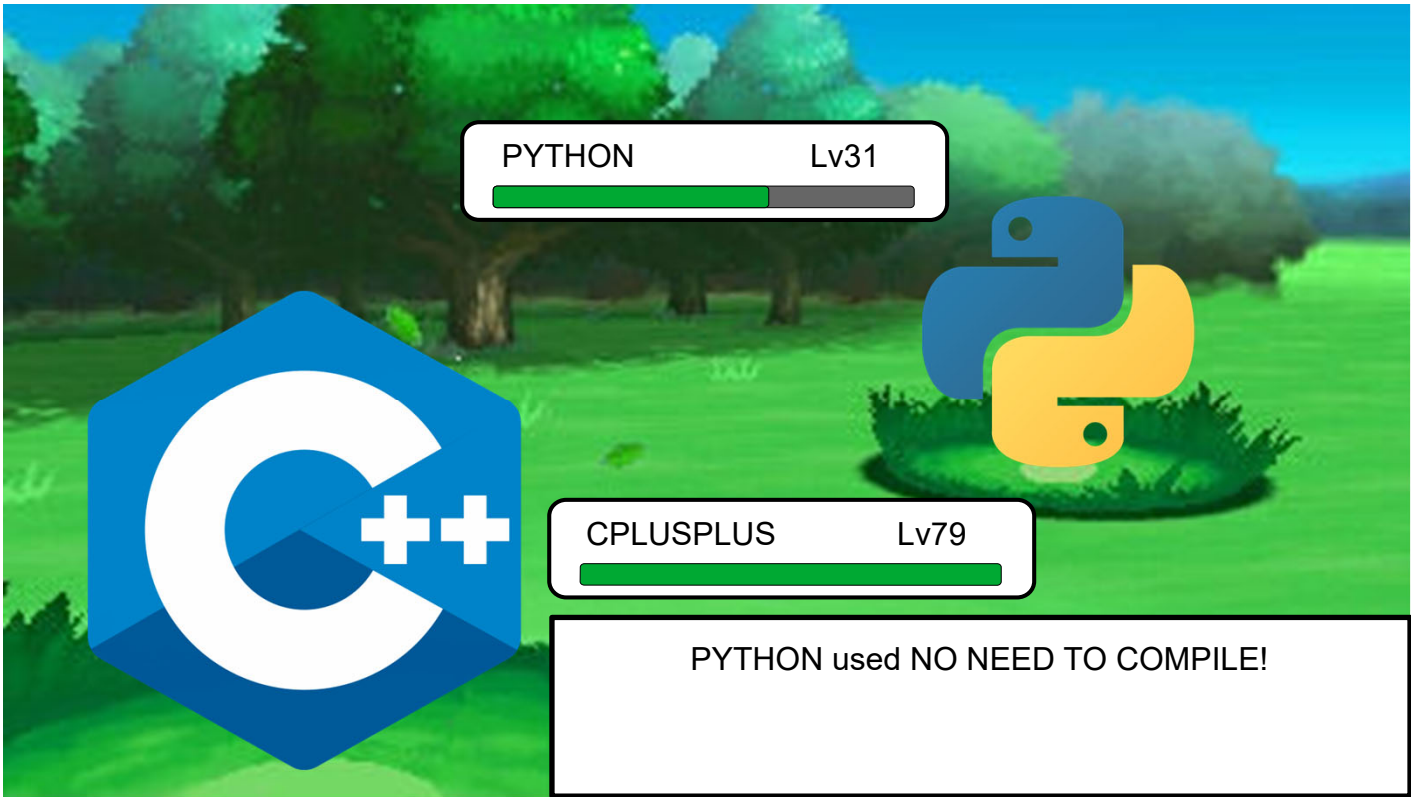

PYTHON Lv31
CPLUSPLUS Lv79
CPLUSPLUS used AVAILABLE LIBRARIES!
It’s not very effective..

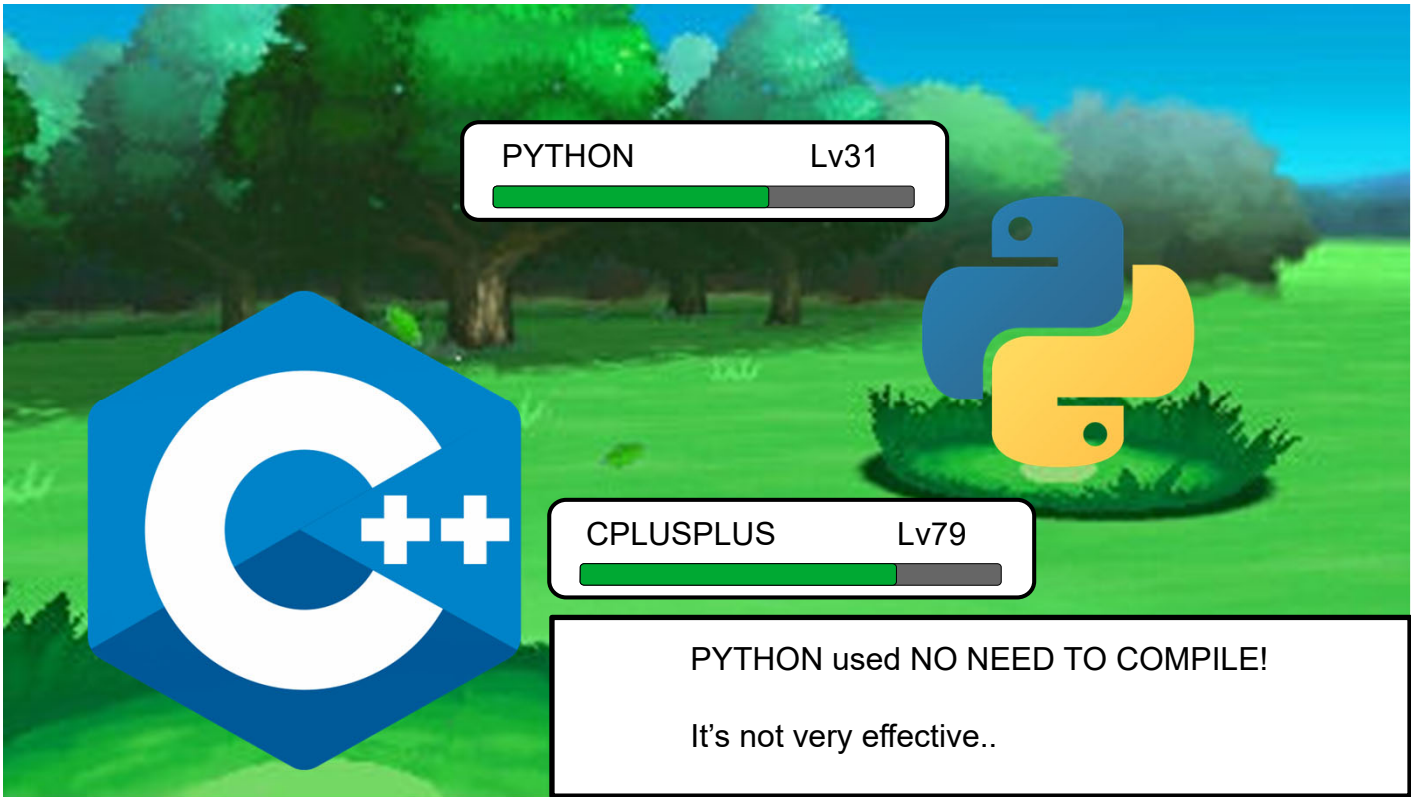

PYTHON Lv31
CPLUSPLUS Lv79
What will PYTHON do?
> FIGHT
LANGS
ITEM
RUN

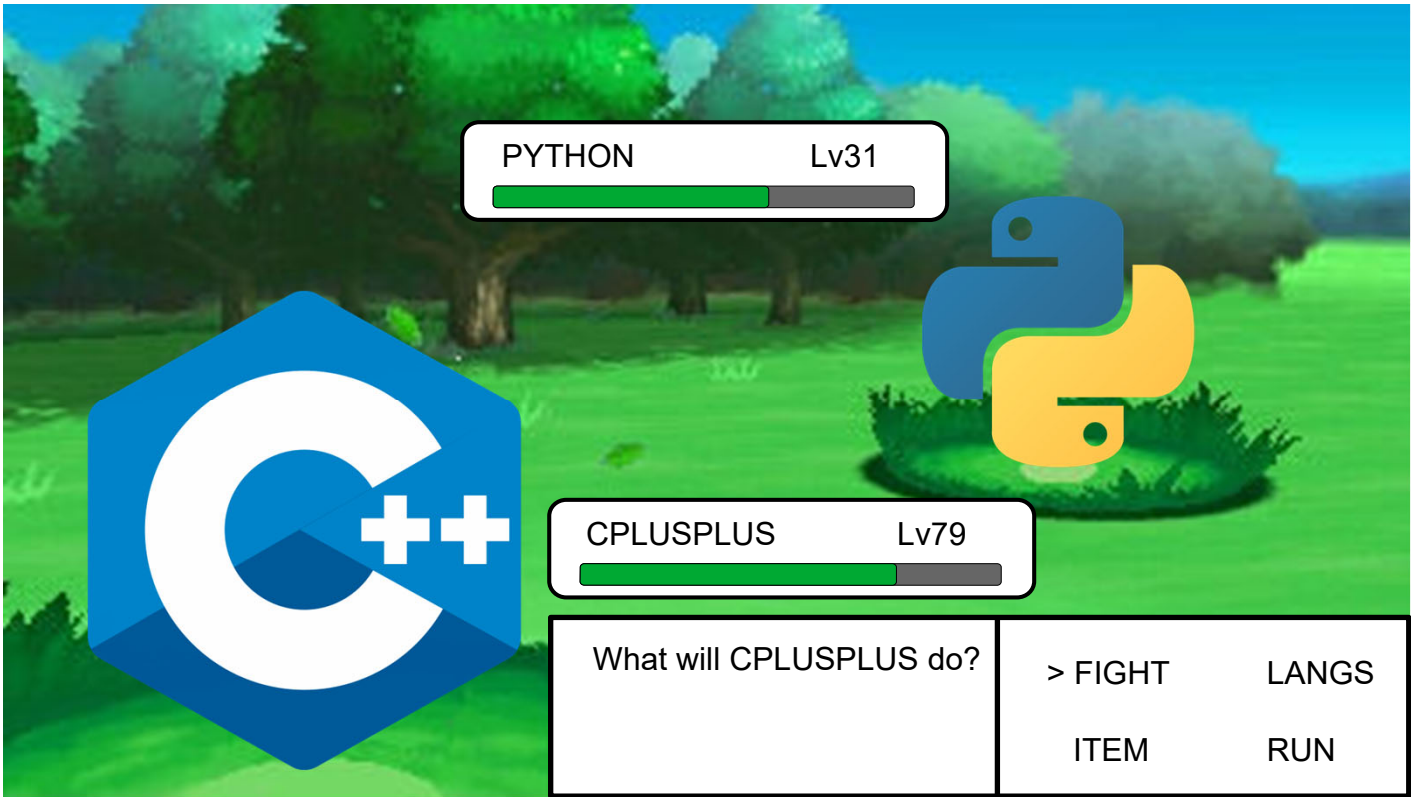


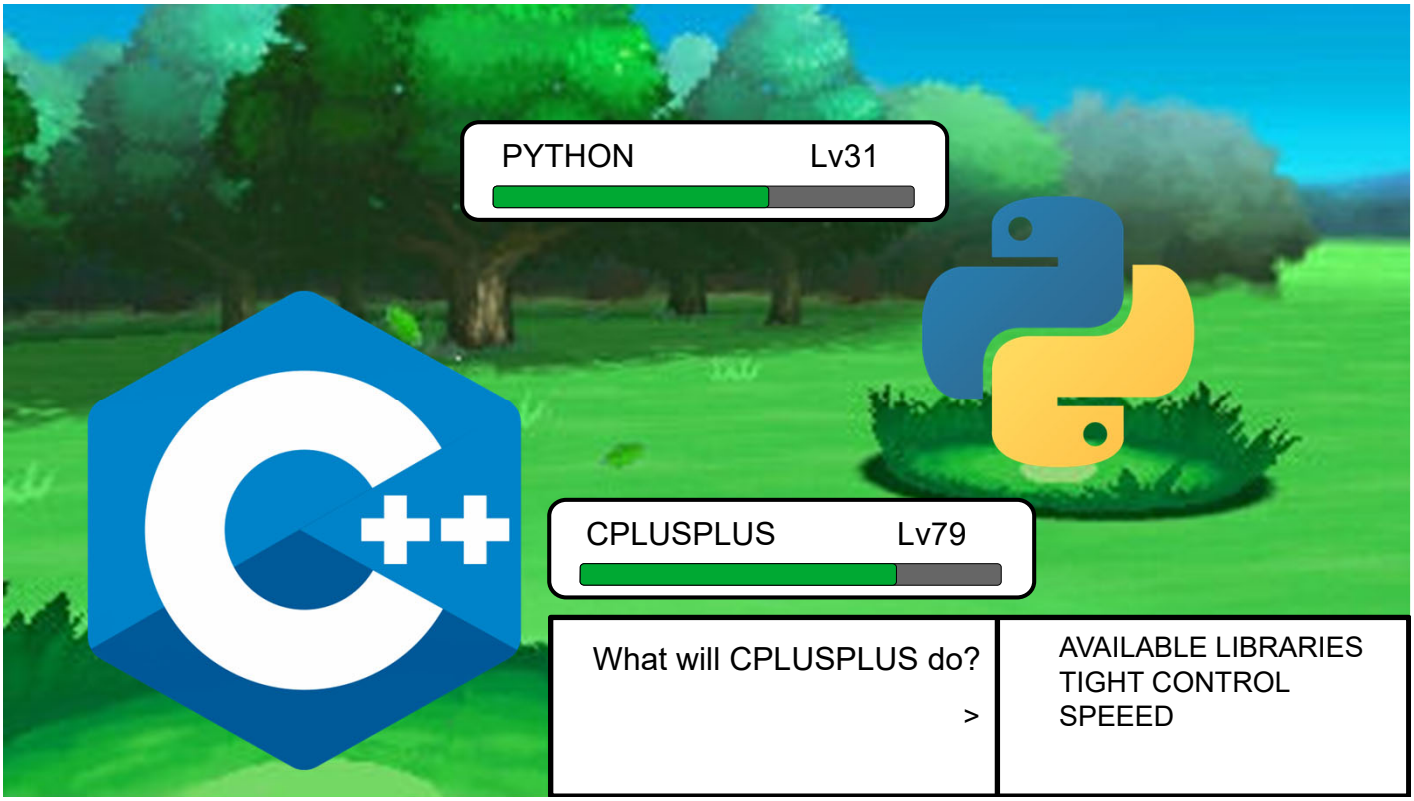
Python er derimot ikke et dårlig språk. Man trenger ikke å kompilere et Python program, og som i C++ finnes det en stor mengde biblioteker.

I tillegg har Python pip-verktøyet, som lar deg lett installere en stor mengde med forskjellige biblioteker. Et slikt 'universell' verktøy finnes ikke for C++.

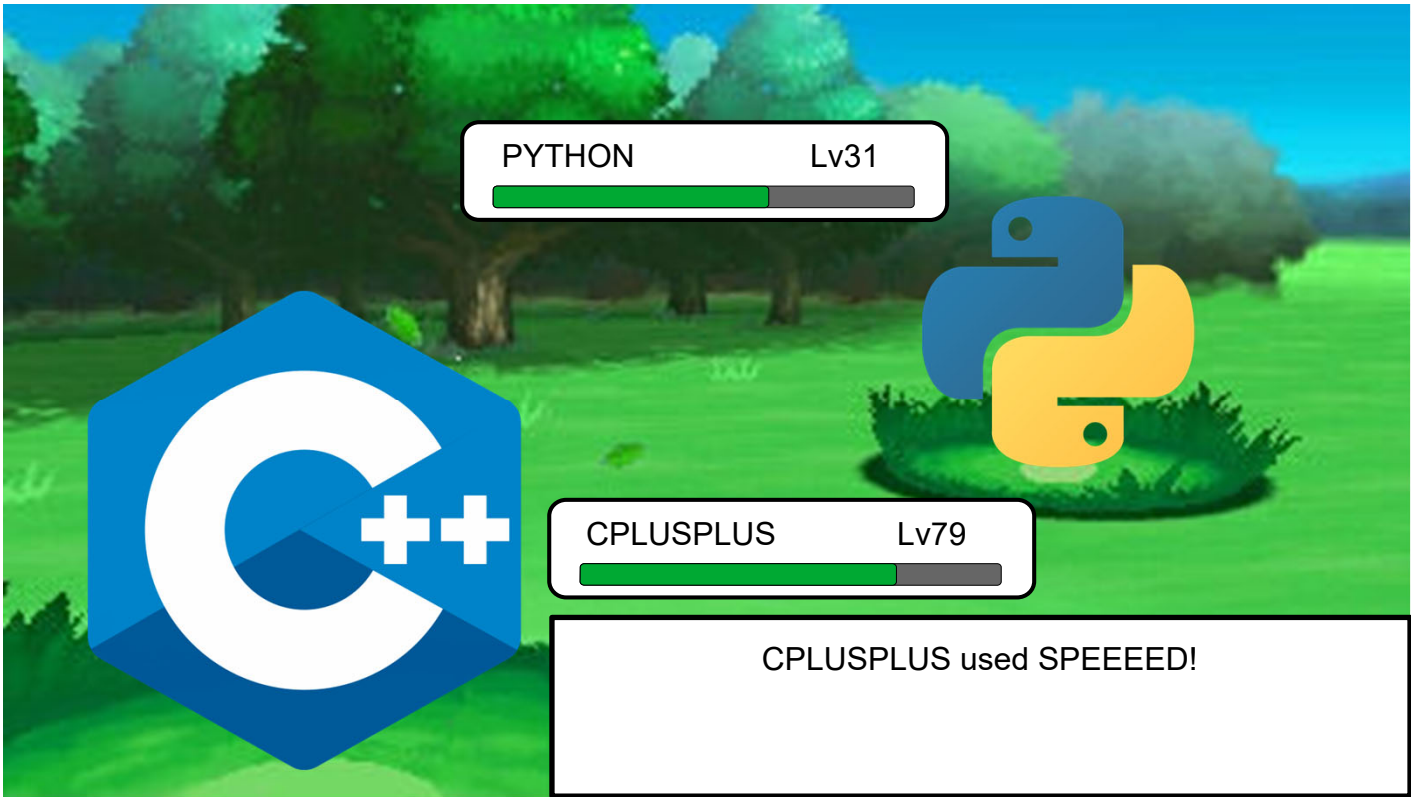

PYTHON Lv31
CPLUSPLUS Lv79
PYTHON used NO NEED TO COMPILE!

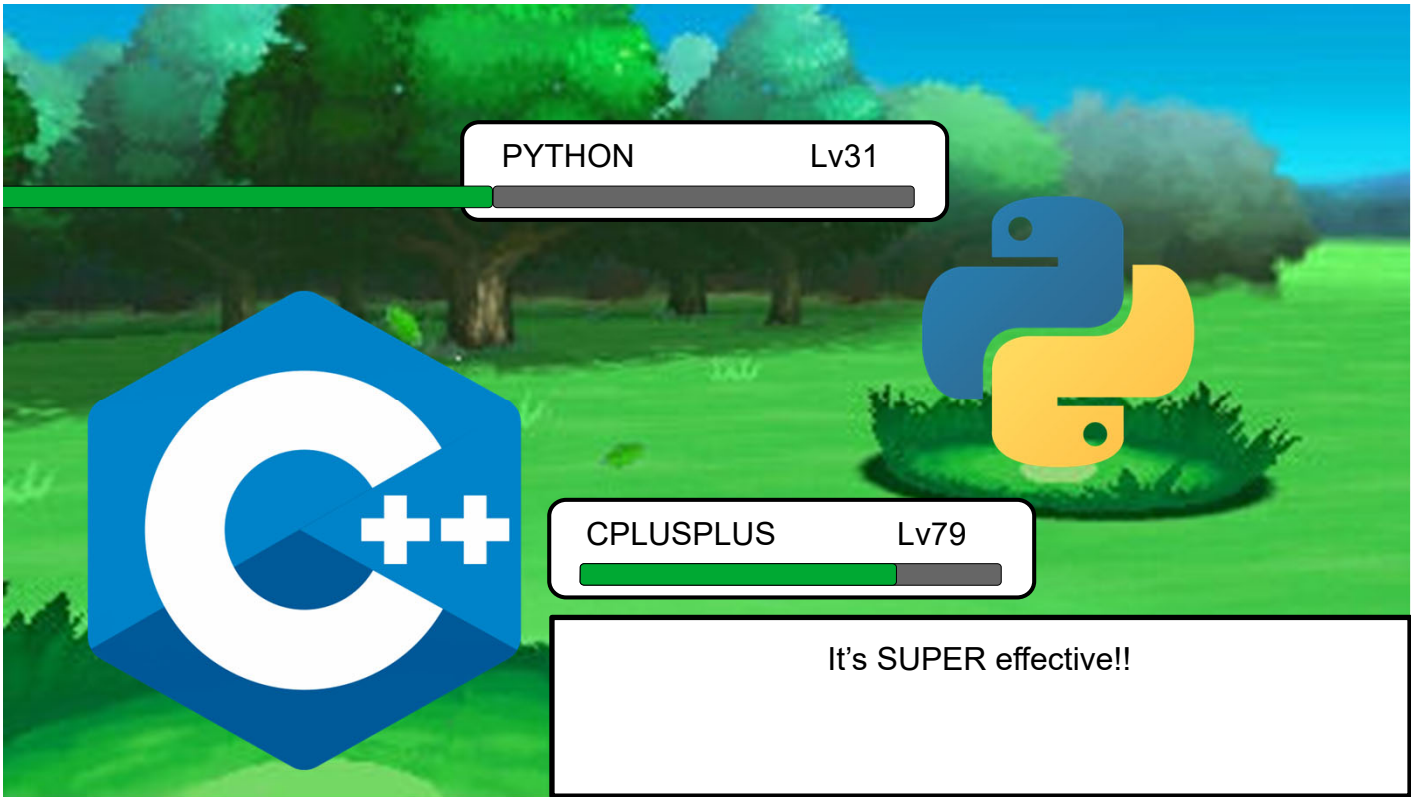

PYTHON Lv31
CPLUSPLUS Lv79
PYTHON used NO NEED TO COMPILE!
It’s not very effective..

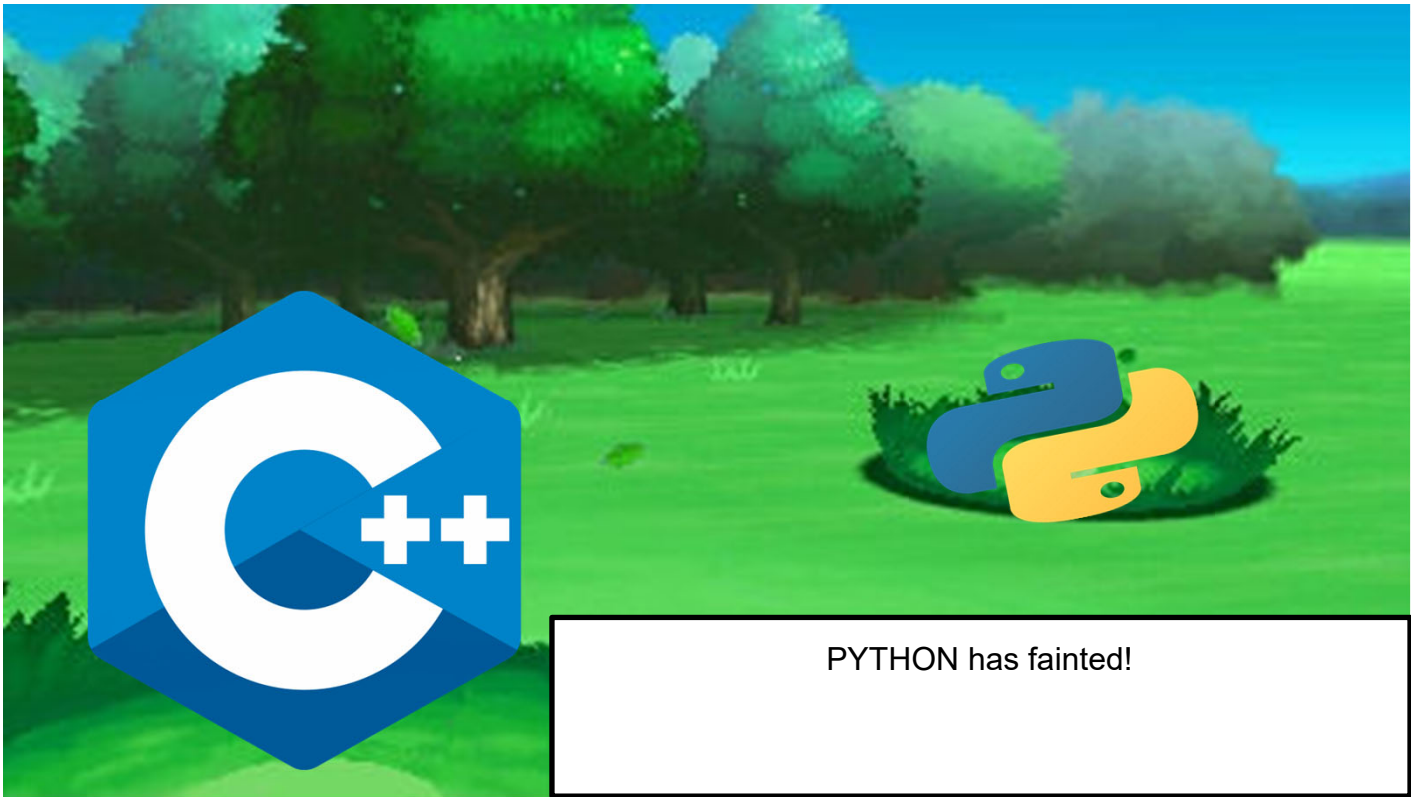

PYTHON Lv31
CPLUSPLUS Lv79
What will CPLUSPLUS do?
> FIGHT
LANGS
ITEM
RUN



C++ forhåndskompileres til kode som kjører på prosessoren din, uten noen abstraksjoner i mellom (slik som Python). Resultatet er at språket produserer program som er langt raskere enn det du kan produsere med Python.


PYTHON Lv31
CPLUSPLUS Lv79
CPLUSPLUS used SPEEEED!


PYTHON Lv31
CPLUSPLUS Lv79
It's SUPER effective!!


C++
PYTHON has fainted!

Today

Basics of the C++ Language

- Variables
- Data types
- Type casting
- Operators
- Text input
- If statements
- Loops
- Functions

(this will be all you need to complete assignment 1)

Today

Basics of the C++ Language

- **Variables**
- Data types
- Type casting
- Operators
- Text input
- If statements
- Loops
- Functions

(this will be all you need to complete assignment 1)

Variables

```
int variableName;
```

Name

Data type (this one is an integer)

```
int variableName = 5;
```

Can optionally be initialised to a value (Recommended!)

- Variables allow you to store values
- Used all the time!
- Works like in Python

En variabel tar vare på en verdi som du kan bruke senere i programmet. Akkurat som i Python har variabler i C++ en datatype, men i C++ kreves det at du selv markerer datatypen på alle variabler eksplisitt i programmene dine.

Variables

```
int variableName;
cout << "The value of variableName is: " << variableName << endl;
```

- Unlike Python, it is possible to create a variable without assigning a value to it

- The contents will be undefined!
  - Best practice: always initialise your variables!
  - The compiler will warn you about this:

```
[6/13] Compiling C++ object program.p/main.cpp.o
../main.cpp: In function 'int main()':
../main.cpp:9:17: warning: 'variableName' is used uninitialized [-Wuninitialized]
    9 |         cout << variableName << endl;
      |                 ^~~~
```


Det er mulig å lage en variabel i C++ uten å spesifisere en verdi, men de får ikke alltid automatisk en standardverdi tilordnet, så det kan skape problemer i programmet.

Det er god praksis å alltid gi en variabel en verdi med en gang du deklarerer den. Deklarer og definer samtidig.

Today

Basics of the C++ Language

- Variables
- **Data types**
- Type casting
- Operators
- Text input
- If statements
- Loops
- Functions

(this will be all you need to complete assignment 1)

Data Types

Category	Data Type	Lowest value	Highest value
Integers	`char`	-128	127
	`unsigned char`	0	255
	`short`	-32,768	32,767
	`unsigned short`	0	65,535
	`int`	-2,147,483,648	2,147,483,647
	`unsigned int`	0	4,294,967,295
	`long long`	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
	`unsigned long long`	0	18,446,744,073,709,551,615
Real numbers	`float`	-∞	∞
	`double`	-∞	∞
Miscellaneous	`bool`	not applicable	not applicable
	`enum`	not applicable	not applicable
	`string`	not applicable	not applicable
	(objects)	not applicable	not applicable
	`void` (no data type)	not applicable	not applicable

Tabellen gir en oversikt over alle «vanlige» datatyper som finnes i C++.

Tabellen viser også at C++ er et mer “lav-nivå” språk enn Python, for det er forskjellige datatyper som alle representerer heltall.

Data Types

4 questions:

- Why are there different integer and real types?
- Why are there signed and unsigned integers?
- What happens if the integer values go out of range?
- How do I decide what data type to use?

Data Types

4 questions:

- **Why are there different integer and real types?**
  - We sometimes work with data which uses a specific number of bytes per value.
  - Smaller data types may result in better performance
- Why are there signed and unsigned integers?
- What happens if the integer values go out of range?
- How do I decide what data type to use?

Det finnes en rekke datatyper som bare brukes for å ta vare på heltall. Det er fordi noen applikasjonsområder krever at det brukes verdier med en spesifikk størrelse. For eksempel, det brukes ofte 16bit-verdier i lydopptak og medisinske bilder som CT og MRI. I tillegg tar mindre datatyper mindre plass i dataminnet og på harddisken, og det kan være attraktivt når det skal behandles svært store mengder med data.

Data Types

4 questions:

- Why are there different integer and real types?
- **Why are there signed and unsigned integers?**
  - Modern processors support both
  - Use signed for day-to-day computations
  - Use unsigned for manipulating bits,
  - or to show a value is always positive
- What happens if the integer values go out of range?
- How do I decide what data type to use?

I C++ finnes det heltall med og uten fortegn. Heltall med fortegn brukes for å ta vare på heltall i beregninger, mens varianten uten fortegn er nyttig når man ønsker å modifisere enkelte bits, eller indeksere en tabell.

Data Types

4 questions:

- Why are there different integer and real types?
- Why are there signed and unsigned integers?
- **What happens if the integer values go out of range?**
  - The value “loops around”, so probably bad things!
- How do I decide what data type to use?

Hva skjer når verdien er for stor eller for liten til å kunne lagres i en bestem type heltalls-variabel?
Hvis tallet er større enn høyeste verdien fortsetter den fra laveste verdi, og vice versa.

Data Types

4 questions:

- Why are there different integer and real types?
- Why are there signed and unsigned integers?
- What happens if the integer values go out of range?
- **How do I decide what data type to use?**
  - Need an integer? You usually want to use `int`
  - Need a real number? You usually want to use `double`
  - `bool` and `string` should be obvious

Hvordan skal du bestemme deg for hvilken datatype skal du bruke?
Om du trenger et heltall pleier det vanligvis å være int, double for flyttall, og bool eller string der de passer.

The effects of data types: Minecraft's far lands

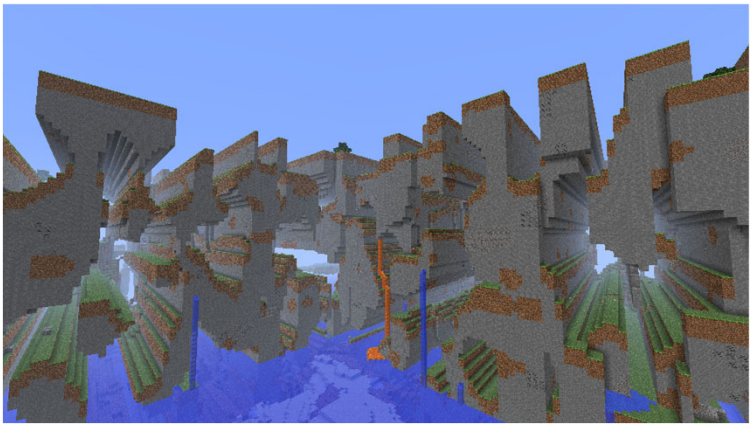
The effects of data types: Minecraft's far lands

The far lands in beta 1.7.3 were caused by an integer overflow.

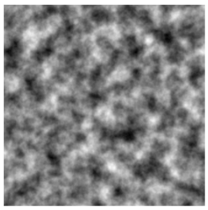
The terrain generator multiplies all block coordinates by approximately 171.

A part of this algorithm converts that number into an integer.

After 12,550,224 blocks this causes an integer overflow!



The far lands



Perlin noise, used in early versions of minecraft

Suboptimal valg av datatype kan ha konsekvenser. Dette er et eksempel fra en gammel versjon av MineCraft, som hadde en bug i terrenggeneratoren (algoritmen som genererer verden), hvor koordinatene til spilleren i verden ble ganget med litt over 171, og det resulterte i at når spilleren gikk 12.55 millioner blokker ut fra verdens nullpunkt ble terrenget helt kaotisk. Dette skyldes at verdier over 12.55 millioner ganget med 171 ble større enn maksverdien som kan lagres i en 32-bit integer.

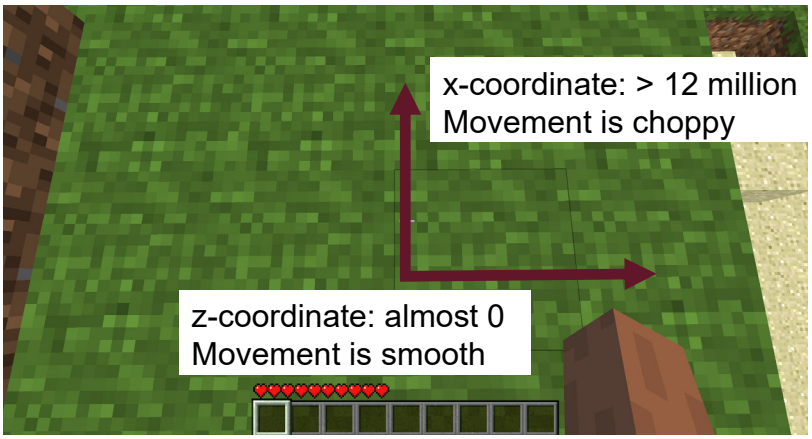
The effects of data types: Minecraft's far lands

Movement was choppy in one direction but not in the other.

Why?

The higher a floating point number gets, the larger the distance between numbers they can represent.

Around 12 million, 32-bit floats can only represent integers



Noe annet som skjer når man kommer seg så langt ut fra verdens nullpunkt viser forskjellen mellom float og double verdier. Spillerkoordinatene er lagret som double, mens når verden skal tegnes blir de oversatt til en float. Float verdier har bare mulighet å representere heltall (0 desimaler) når de blir større enn 12.5 millioner. Dette viser seg når man beveger seg i spillet: z-koordinaten i demonstrasjonen var omtrent 0, og det var ikke noe "hakkete bevegelser», men når man beveger seg langs x-aksen ser det ut som man "hopper" framover.

Today

Basics of the C++ Language

- Variables
- Data types
- **Type casting**
- Operators
- Text input
- If statements
- Loops
- Functions

(this will be all you need to complete assignment 1)

Type casting

- Casting is conversion of data types
  - This can lead to precision or data loss, but the compiler can warn you
- Types are converted implicitly if no accuracy is lost
  - For example, `char` to `int` or `float` to `double`
- Types can be converted explicitly using static_cast<>():
  - `int anInteger = 5;`
  - `float floatValue = static_cast<float>(anInteger);`
- For converting between numeric types, you can also use the type as a function:
  - `int anInteger = 5;`
  - `float floatValue = float(anInteger);`

"casting" er oversettelse mellom datatyper. Som vi så i demonstrasjonen kan dette føre til et tap av presisjon, eller verdien blir fullstendig annerledes. Ved alle oversettelser hvor dette ikke kan bli et problem (for eksempel fra float til double), gjør C++ oversettelsen automatisk for deg. Om du likevel ønsker å oversette potensielt problematiske verdi x, kan du bruke static_cast<>(x), eller ved enkle datatyper som float og int kan du også bruke navnet på datatypen som et slags "funksjon" int(x).

The name «casting» comes from metallurgy!

Oversetting av datatyper heter “casting” på engelsk, og det er avledet av ordet for metallstøping

Type casting

- Why explicit type conversion when much of it happens implicitly?
  - Types can not always be converted implicitly
  - Explicitly converting a type shows the conversion is intentional
    - And a loss of accuracy is acceptable
  - For numeric types, some calculations require type conversions to produce correct results

- Converting numbers to and from strings is a bit different.
  - We'll look at those in a future lecture.

Det er ikke alltid mulig å oversette datatyper automatisk. Og det er uansett alltid god praksis å gjøre oversettelser av datatyper eksplisitt, slik at du viser at du gjør oversettelsen med vilje. Det er også mulig at du trenger å oversette datatyper for å endre resultatet av en beregning (Eksempel: heltalls-divisjon, eller «vanlig divisjon»).

Å oversette fra og til tekststrenger (string) fungerer annerledes, og vi kommer til å behandle det i en senere forelesning.

Today

Basics of the C++ Language

- Variables
- Data types
- Type casting
- **Operators**
- Text input
- If statements
- Loops
- Functions

(this will be all you need to complete assignment 1)

Operators

- Roughly two main categories (that we use in this course):
  - Arithmetic operators: compute stuff
  - Logic operators: compare stuff

Vi har fram til nå sett hvordan vi kan lagre og oversette datatyper, men ikke enda hvordan de _brukes_. Det gjør vi med såkalte operatorer. Det er mange slike operatorer (vi kommer ikke til å behandle alle i emnet), men de som brukes mest kan fordeles i to grupper: logiske og aritmetiske operatorer.

Arithmetic operators

Calculation

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo

Assignment

Operator	Description
+=	Increment by
-=	Decrement by
*=	Multiply by
/=	Divide by
%=	Remainder of
=	Assignment
++	Increment by 1
--	Decrement by 1

Examples

```
int sum = 5 + 3;
cout << sum << endl; // 8

sum++;
cout << sum << endl; // 9

sum *= 3;
cout << sum << endl; // 27
```


Aritmetiske operatorer brukes for å beregne verdier. Det er mulig å bruke en kortform som erstatning for flere operatorer (+ og = for eksempel)

Logic operators

Value comparison

Operator	Description
<	Is less than
<=	Is less than or equal to
==	Is equal to
!=	Is not equal to
>=	Is greater than or equal to
>	Is greater than

Boolean

Operator	Description
!	Boolean NOT
&&	Boolean AND
||	Boolean OR

Examples

```
bool lessThan = 5 < 4;
cout << lessThan << endl; // 0

bool notEqual = 3 != 9;
cout << notEqual << endl; // 1
```

```
bool inverse = !(3 > 2);
cout << inverse << endl; // 0
```

Operators

- C++ does not have Python's // operator
  - When both operands are integers, integer division is used. Otherwise, always float division.
- A float in Python is always 64-bit. In C++ that would be called a double.
- The order in which operators are evaluated follows algebraic rules.
- Good practice: do not use the == and != operators with floating point numbers.

Today

Basics of the C++ Language

- Variables
- Data types
- Type casting
- Operators
- **Text input**
- If statements
- Loops
- Functions

(this will be all you need to complete assignment 1)

There's a way cin..

- Just like cout, there's a cin
- Uses >> to read a value into a variable
- The variable you read into can be of any of the common data types.

```
cout << "What is your favourite food?  " << endl;
string food;
cin >> food;
cout << "That's awesome, " << food << " is also my favourite!" << endl;
```


Ser du problemet med programmet som er vist? (svaret er på neste slide)

There's a way cin..

- `cin` only reads one word or one number at a time!
  For example: entering "1.5 2.3 3.8 4.4" here will print "1.5":

```
double value;
cout << "Write some numbers: ";
cin >> value;
cout << "Value: " << value << endl;
```

- If you use cin again in this example, "value" becomes 2.3

- You can use the getline() function to read a line:

```
string message;
cout << "Write a message: ";
getline(cin, message);
cout << "The message was: " << message << endl;
```


(svaret fra forrige slide: om ditt favorittmat består av flere ord blir bare første ordet skrevet ut)

Today

Basics of the C++ Language

- Variables
- Data types
- Type casting
- Operators
- Text input
- **If statements**
- Loops
- Functions

(this will be all you need to complete assignment 1)

If statements

```
bool condition = true;

if(condition) {
    cout << "condition is true!" << endl;
} else {
    cout << "condition is false." << endl;
}
```

- If statements allow you to run a piece of code based on a boolean true/false condition

Kanskje husker dere det vi trenger for valg: en 'if' setning. Setningen tar en Boolean betingelse, og når den er sant, kjører koden mellom krøllparentesene { } øverst, og ellers den anvist med "else". Merk at "else" og etterfølgende kodeblokken (mellom {} krøllparentesene) er ikke obligatorisk (du kan se dette på neste slide).

If statements

```
bool condition = true;
bool alternateCondition = true;

if(condition) {
    cout << "condition is true!" << endl;
} else if(alternateCondition) {
    cout << "condition is false." << endl;
    cout << "alternateCondition is true!" << endl;
} else {
    cout << "condition is false." << endl;
    cout << "alternateCondition is also false." << endl;
}
```

- Multiple statements can be chained

Kanskje husker dere det vi trenger for valg: en 'if' setning. Setningen tar en Boolean betingelse, og når den er sant, kjører koden mellom krøllparentesene { } øverst, og ellers den anvist med "else". Merk at "else" og etterfølgende kodeblokken (mellom {} krøllparentesene) er ikke obligatorisk (du kan se dette på neste slide).

Today

Basics of the C++ Language

- Variables
- Data types
- Type casting
- Operators
- Text input
- If statements
- **Loops**
- Functions

(this will be all you need to complete assignment 1)

Loops

- There are three different loop variations in C++
  - The for loop:

```
for(int i = 0; i < 10; i++) {

}
```

  - The while loop:

```
int i = 0;
while(i < 10) {
    i++;
}
```

  - The do while loop:

```
int i = 0;
do {
    i++;
} while(i < 10);
```

- Loops are interchangeable: each can be rewritten as another type.
- For example, all the shown loops repeat their loop body 10 times!

Loops

- Where should you use each loop type?
  - The for loop:
    - When you know in advance how many iterations you need
  - The while loop:
    - When you don't know in advance how many iterations you need
  - The do while loop:
    - When you need the loop body to be executed at least once

Loops

```
for(int i=0;…){}
```

```
int i = 0;
while(i < 10) {
    i++;
}
```

In practice, the for loop is the one used most (by far)

Av disse forskjellige løkker brukes for løkken langt mer enn de andre.

Today

Basics of the C++ Language

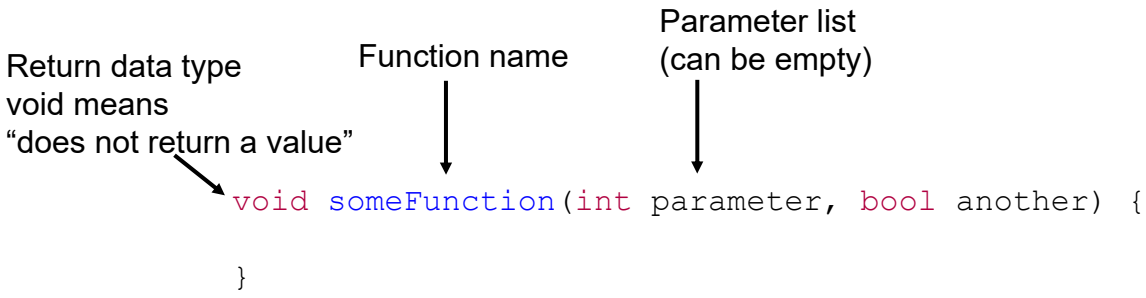
- Variables
- Data types
- Type casting
- Operators
- Text input
- If statements
- Loops
- **Functions**

(this will be all you need to complete assignment 1)

Functions

- Functions are pieces of code that can be run on demand
- Functions allow reusing functionality across your program
  - And if you fix a bug in a function used in multiple places, you have fixed it everywhere!
- Functions allow abstraction: as long as the function behaves the same way, you can change how it works.

Functions


Return data type
void means
“does not return a value”
Function name
Parameter list
(can be empty)
void someFunction(int parameter, bool another) {
}

Functions

Values are returned using a return statement.

```
int add(int a, int b) {
        return a + b;
}
```

Functions are called by writing the name, followed by any parameter values separated using commas.

```
void printAnswer() {
        cout << add(21, 21) << endl;
}
```

Functions

- Declaration versus Definition:
  - Declaration: what does the function look like?
    - Does not have code surrounded by { }

```
void functionA();
```

  - Definition: what does the function do?
    - Contains code surrounded by { }

```
void functionA() {
}
```

Functions

Functions must be declared before they can be used.

Order matters: the program is analysed top to bottom

```
#include "std_lib_facilities.h"

void functionA() {
    cout << "I am function A" << endl;
}

void functionB();

int main() {
    functionA();
    functionB();
    return 0;
}

void functionB() {
    cout << "I am function B" << endl;
}
```

Example X

Det er verdt å nevne at som i Python kreves det at en funksjon er definert før den kan brukes.

That’s all for assignment 1!

The earlier you start, the more time you have to get help from us when you get stuck!

- Images used:

https://www.weld2cast.com/sand-casting/
http://vignette1.wikia.nocookie.net/antagonists/images/f/f7/Darth_Vader.png/revision/latest?cb=20141211094955https://www.minecraftforum.net/forums/minecraft-java-edition/discussion/2391376-recreating-the-far-lands

Og dette er lenker til bildene jeg har brukt i forelesningen