Lær å kode i Unity – del 3.1: Variabler

I denne delen av kurset skal vi se nærmere på verdenen innenfor programmering. Programmering er en svært viktig del av det å lage et spill. Det er der du finner alle instruksene til hva som skal skje i spillet. Dette kan sammenliknes med for eksempel en matoppskrift:

BROWNIES

INGREDIENSER

12
PORSJONER

- 200 g mørk sjokolade, 70% kakao
- 200 g smør
- 4 stk egg
- 3 dl sukker
- 4 dl hvetemel
- 1 ts bakepulver
- 2 dl grovhakkede valnøttkjerner

Til pynt:

- 1 kurv friske bringebær
- 2 ss melis

SLIK GJØR DU

1. Forvarm stekeovn til 150 °C.
2. Finhakk sjokolade og ha den i en glassbolle sammen med smør. Sett bollen over en liten kjele med vann som småkoker (vannbad) til sjokoladen har smeltet.
3. Sett bollen tilbake på kjøkkenbenken og ha i egg. Bruk en visp og rør godt slik at alt blander seg. Ha deretter i sukker.
4. Tilsett hvetemel og bakepulver til slutt, og rør alt godt sammen.
5. Smør en liten langpanne, ca. 25x35 cm. Kle så pannen med et bakepapir. Hell røren over i langpannen og stryk den utover.
6. Stek kaken i ca. 40 minutter. Avkjøl kaken og del den i biter.

Server gjerne brownies bitene med en iskaffe eller et glass kald melk. .

```
// Eksempel i kodespraaket "Java"
import java.util.Scanner;

class apples{
    public static void main (String[] args){

        // "Ingredienser:"
        Scanner input = new Scanner (System.in);
        int total = 0;
        int grade;
        int average;
        int counter = 0;

        // "Slik gjor du:"
        while (counter < 10){
        grade = input.nextInt();
        total = total + grade;
        counter++;
        }

        average = total/10;

        // "Resultat:"
        System.out.println("Average amount of apples: " + average);
    }
}
```

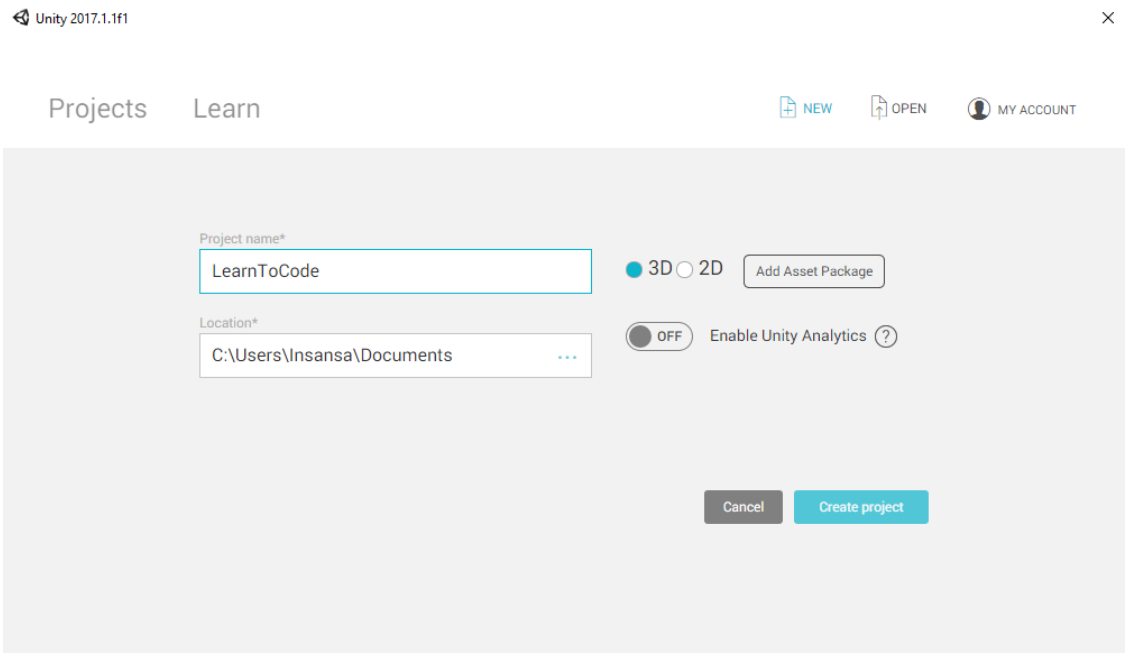
Du har en rekke med ingredienser (disse kaller vi for variabler i kodespråket), en instruksjon på hvordan ingrediensene skal brukes og til slutt får du et ferdig produkt.

Av og til når vi følger en matoppskrift får vi ikke alltid det resultatet vi ønsket å få. Dette kan skylles det at personen ikke klarte å følge oppskriften til punkt og prikke. Det funket kanskje for én person (eller på én datamaskin), men ikke for en annen. Kanskje var oppskriften vanskelig å forstå? Det er svært viktig å være nøye med «oppskriften»/koden vår når vi lager spill. Ved å skrive god kode hindrer vi at resultatet kan feiltolkes av datamaskinen underveis og vi unngår å få et dårlig spill fult av «bugs». Det er dermed svært viktig å være nøye med både gramatikk (Skal det være stor eller liten bokstav her? Har jeg brukt riktige tegn? Osv.) og logikk (Er koden skrevet på en logisk måte som datamaskinen forstår?)

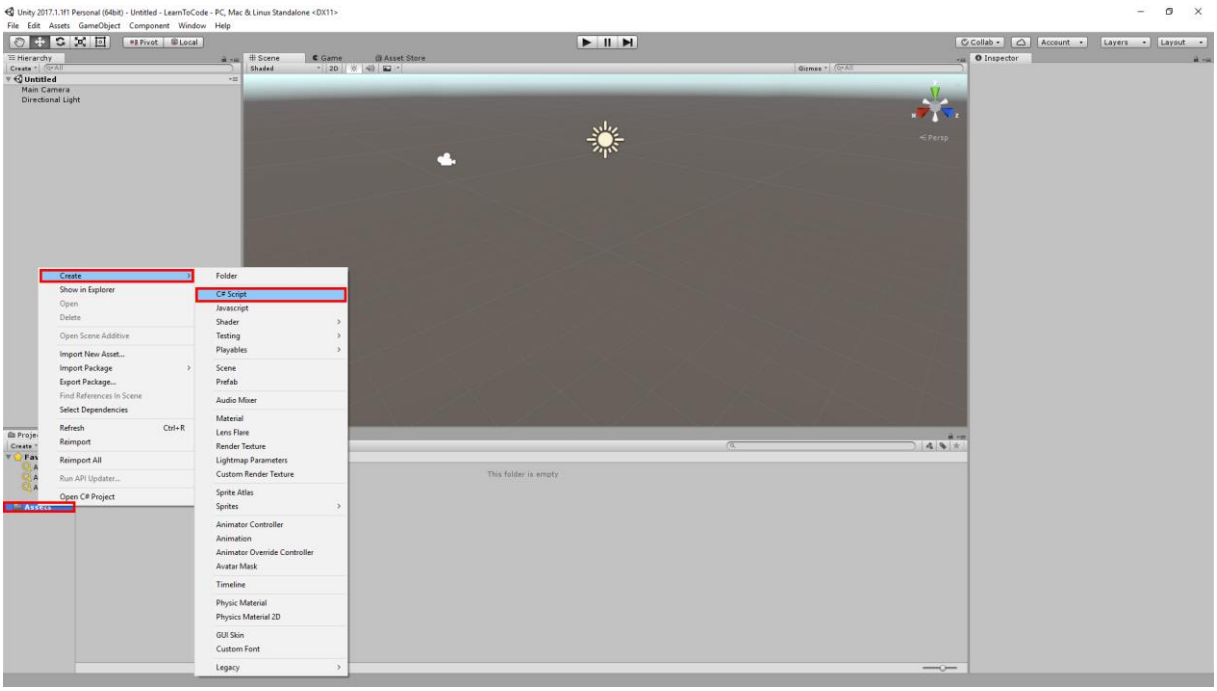
(Vil du at spillet ditt skal bli som kaken vist til venstre eller som kaken vist til høgre?)

Som sagt må «oppskriften» være skrevet på et forståelig språk dersom den skal kunne utføres. For oss mennesker kan dette være norsk, engelsk osv. En datamaskin har egne språk den forstår. Det finnes tusenvis av slike ulike språk, og hver av dem har sine områder de er gode på. Språket vi skal bruke her heter C# (uttales C sharp) og brukes mye til spillprogrammering, spesielt i Unity.

Åpne Unity. Klikk file – new project og kall prosjektet for «LearnToCode».

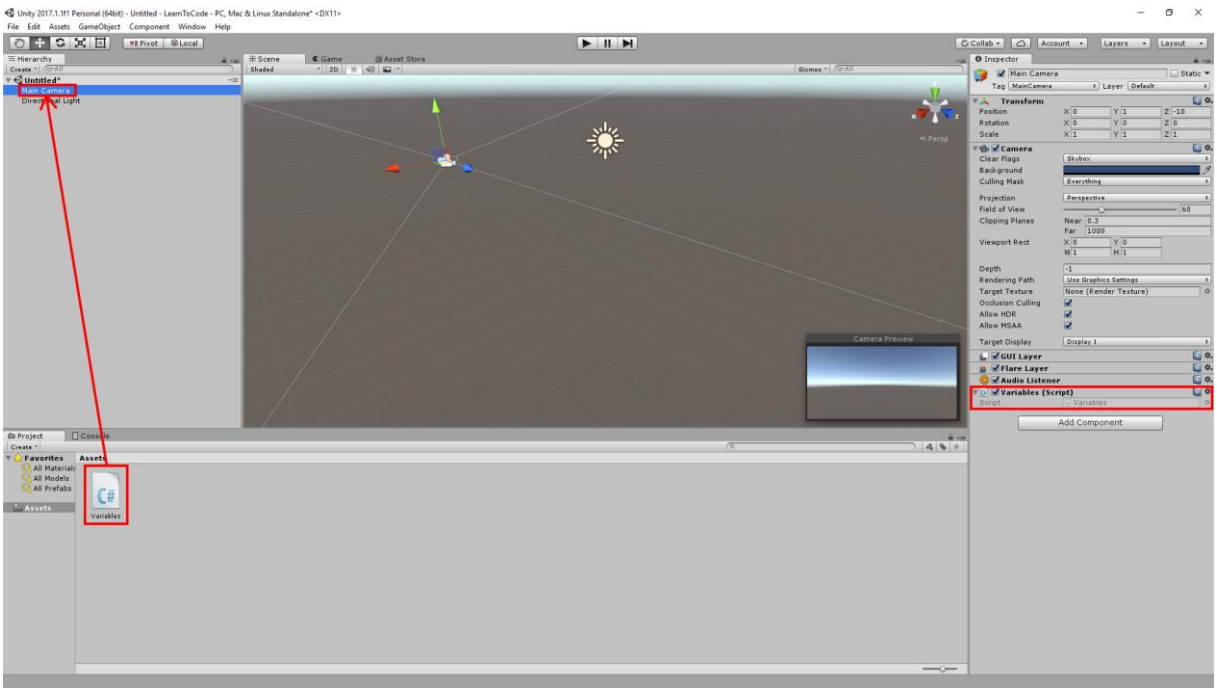


Klikk project – høyreklikk på assets – create – C# Script, kall den «Variables».

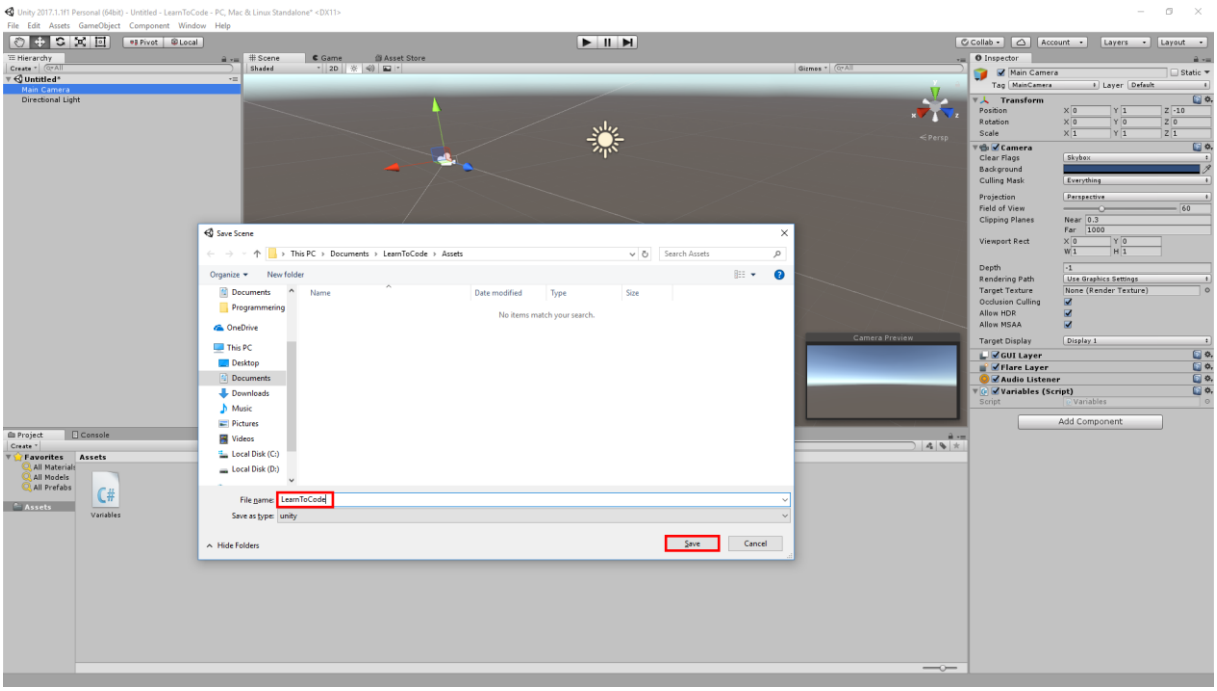


Når du jobber i Unity kommer du til å lage mange forskjellige «Scripts», og disse befinner seg inni prosjektet ditt slik at du lett kan dra dem over til objekter i spillet ditt. Så det første du skal gjøre er å dra skriptet ditt over til «Main Camera». Dette skriptet vil dermed starte når «Scene» starter. Konseptet med å feste kode til objekter er ulikt fra andre typer programmeringssystemer. Det er kun i Unity vi gjør dette, men selve kodedelen er den samme uansett hvilket program du bruker.

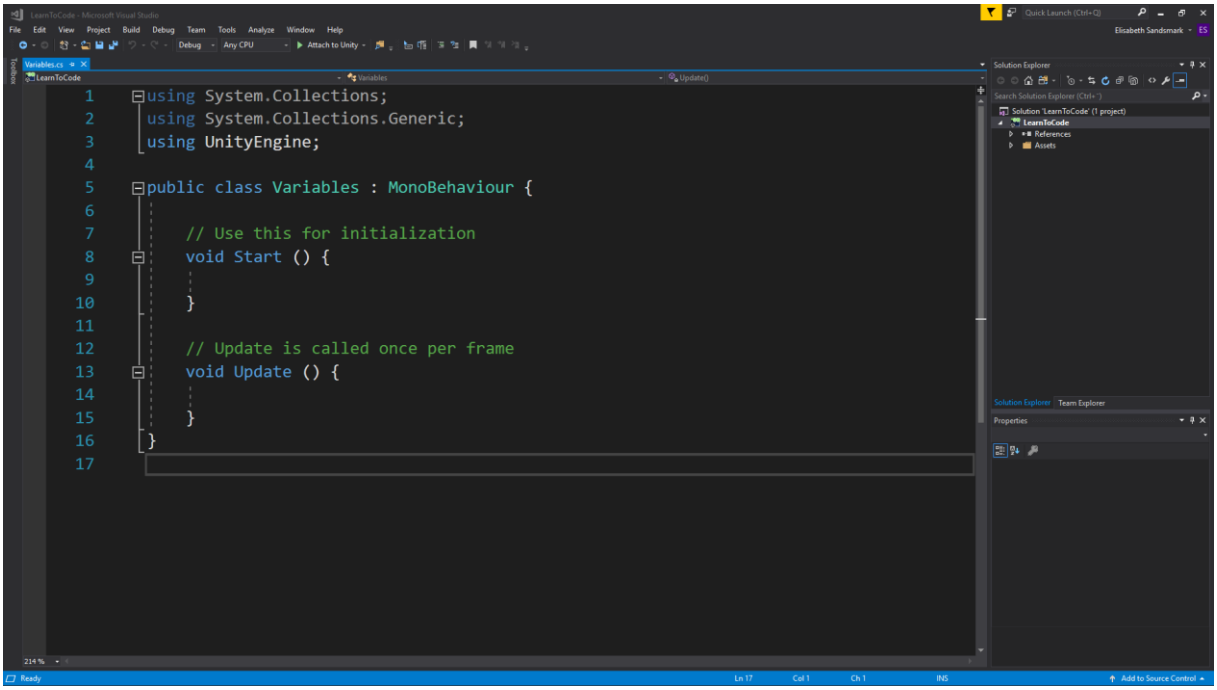
Hvis du nå klikker på Main Camera vil du kunne se skriptet ditt i «Inspector».



Lagre «Scene» (Ctrl + S) og kall den LearnToCode.



Åpne skriptet «Variables» (i MonoDevelop eller VisualStudio etc)



Legg merke til at det allerede befinner seg en «Start-funksjon» og en «Update-funksjon» i skriptet (mer om funksjoner senere). Start-funksjonen blir brukt når spillet starter, mens Update-funksjonen blir brukt om og om igjen gjennom hele spillet i en «loop»/sirkel.

Variabler

Det første vi skal se på er variabler («variables»). Forestill deg at datamaskinens minne består av et stort antall celler (lagerplasser) hvor hver av disse cellene har sin egen adresse og kan lagre én enkelt verdi. For å reservere en av disse cellene må vi gi verdien et navn. Variabler er navngitte verdier som brukes for å lagre informasjon. Vi skiller mellom ulike typer variabler, for eksempel string (lagrer tekst), int (lagrer heltall), double (lagrer reelle tall) osv.

Dette er en visualisering av hvordan du kan tenke deg at det ser ut når informasjonen blir lagret, ikke slik det egentlig ser ut:

Jeg har markert variabeltype som blå, variabelnavn som gul og lagret verdi som grønn fremover i dokumentet slik at det er lettere å se hva som er hva.

Adresse: 0x10001 Variabel type: int Variabelnavn: heroHP Verdi lagret: 100	Adresse: 0xB0004 Variabel type: string Variabelnavn: favoriteFruit Verdi lagret: Pear	Adresse: 0x80000 Variabel type: string Variabelnavn: favoriteDrink Verdi lagret: Milk
Adresse: 0xC0110 Variabel type: string Variabelnavn: heroName Verdi lagret: Link	Adresse: 0xB0000 Variabel type: string Variabelnavn: equippedWeapon Verdi lagret: Master Sword	Adresse: 0xA0000 Variabel type: int Variabelnavn: heroAge Verdi lagret: 17
Adresse: 0x10010 Variabel type: double Variabelnavn: gold Verdi lagret: 24.6	Adresse: 0x11111 Variabel type: string Variabelnavn: sidekickName Verdi lagret: Bob	Adresse: 0x00000 Variabel type: string Variabelnavn: favoriteColor Verdi lagret: green

Variabler – «String» (tekst)

Når du jobber med et spill kommer du til å lage mange variabler og det kan være lurt å gi dem gode navn slik at du selv forstår hva variabelen peker mot. La oss starte med vår aller første variabel under Start-funksjonen. I et spill (for eksempel en RPG) kommer vi blant annet til å ha en hovedperson eller en helt. Denne helten trenger et navn. Siden et navn består av tekst bruker vi string som variabeltype.

Kall variablen for heroName eller mainCharacter eller andre passende navn. Så lenge det inneholder kun bokstaver kan du gi det hvilket som helst navn. Dette navnet er kun for deg som programmerer og vil ikke være synlig for spilleren.

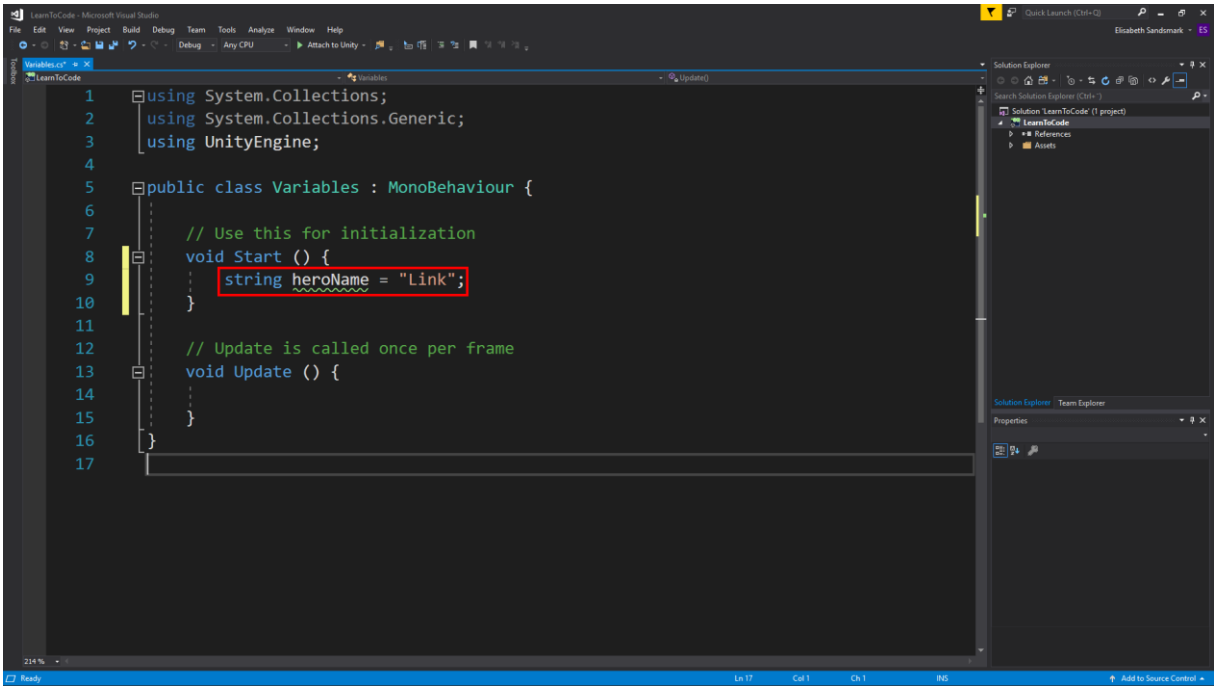
Gi helten et navn f.eks. Link eller Superman osv. Dette er dette navnet vi kommer til å se i spillet. (Vi bruker anførselstegn/sitattegn for å markere verdien til en string. Dette gjør at vi også kan bruke mellomromstegn dersom vi ønsker å skrive setninger. For eksempel «Link, Hero of Time.»)

```
string heroName = «Link»;
```

Dette gjør at informasjonen om navnet til helten vår blir lagret i minnet til datamaskinen, og vi kan bruke denne informasjonen seinere i koden vår når vi måtte ønske.

NB! Det er vanlig å starte variabelnavnet med liten bokstav. Stor bokstav brukes ofte på første bokstav i ord nummer 2, 3.. osv. dersom du har flere ord i samme variabelnavn (eks: name, hero, heroName, mainCharacter, mainCharacterName).

Merk deg også at alle deklarasjoner (og for øvrig alle andre instruksjoner) blir avsluttet med et semikolon ;.



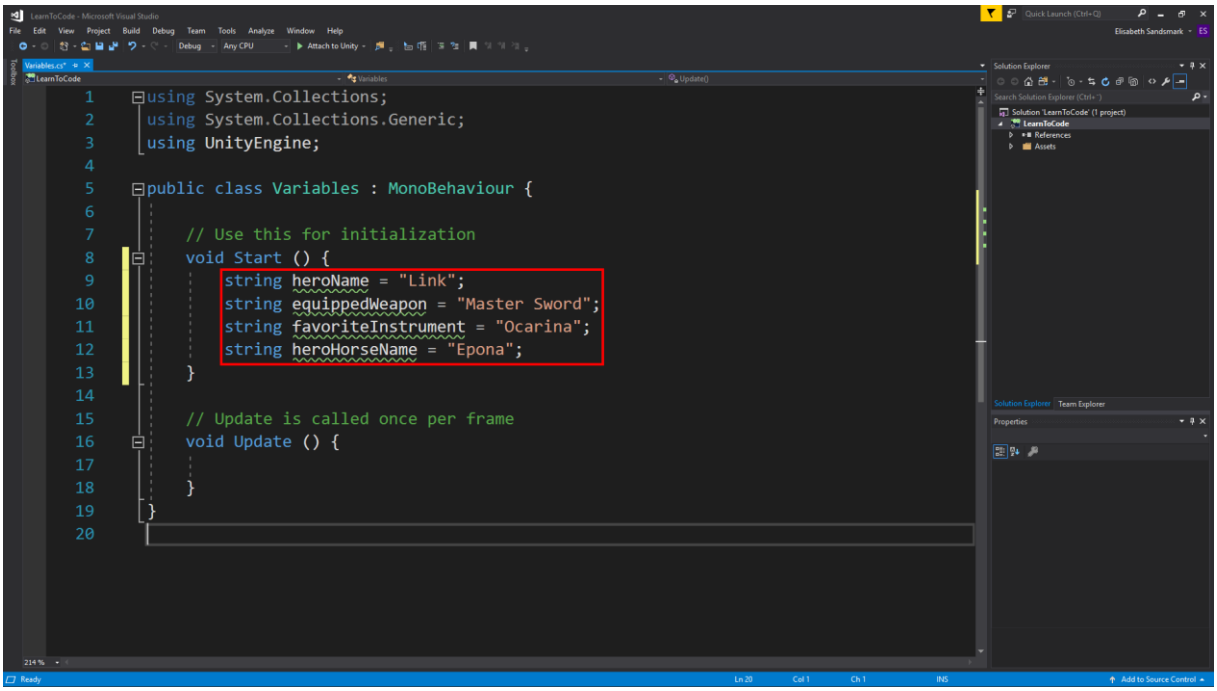
Neste vi kan lage er for eksempel et våpen til helten vår. (Gi dem hvilke som helst navn du ønsker)

```
string equippedWeapon = «Master Sword»;
```

Legg til så mange variabler du ønsker. Eksempler:

```
string favoriteInstrument = «Ocarina»;
```

```
string heroHorseName = «Epona»;
```



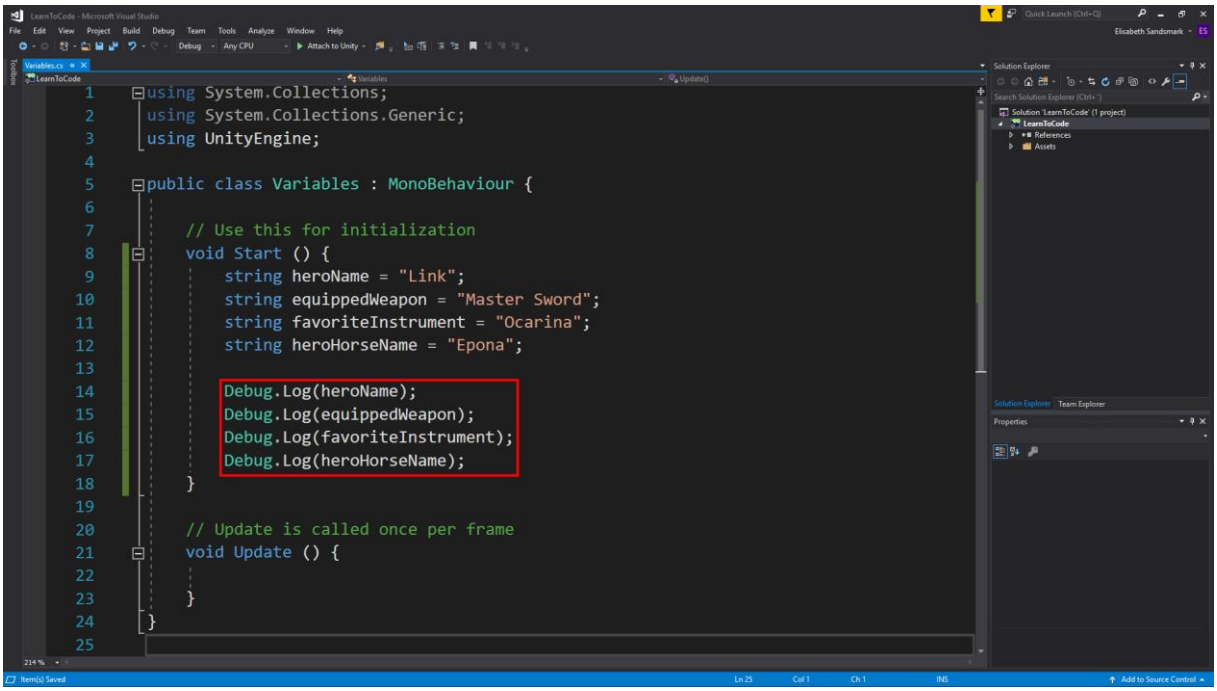
Vi kan nå for eksempel skrive ut disse navnene til skjermen. Dette gjøres på følgende måte:

Debug.Log (heroName);

Debug.Log (equippedWeapon);

Debug.Log (favoriteInstrument);

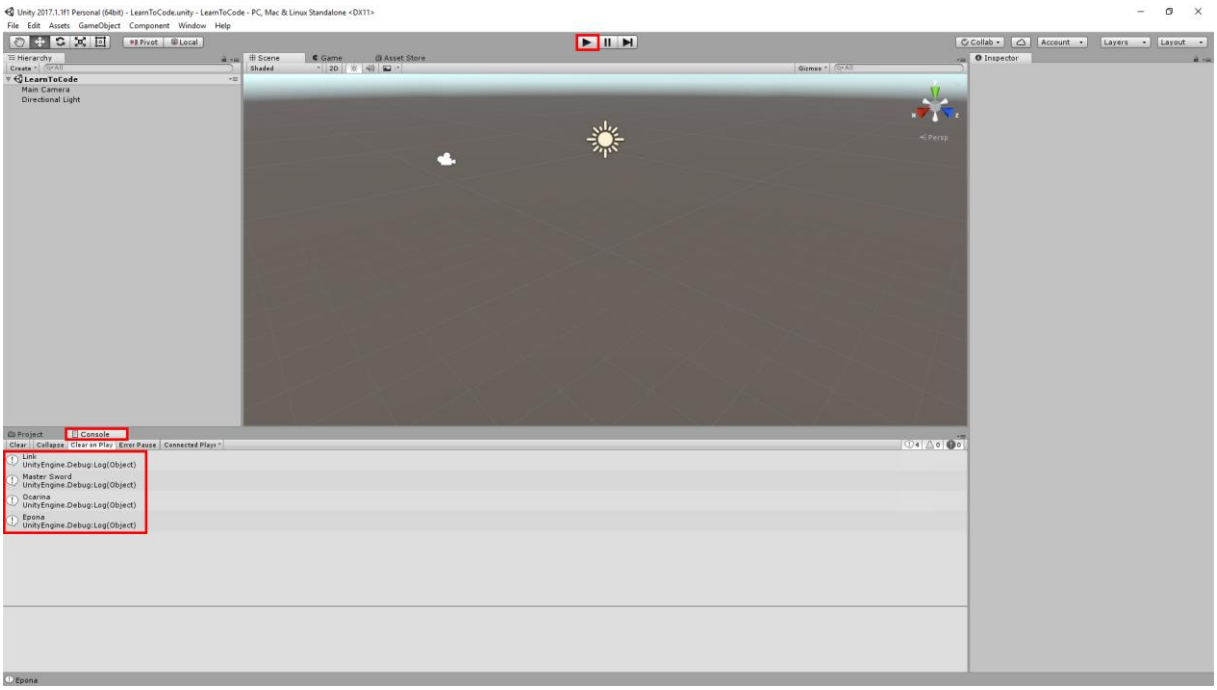
Debug.Log (heroHorseName);



Lagre dette og gå tilbake til Unity. Klikk på «Console» og klikk på play knappen for å kjøre spillet. (Klikk igjen for å avslutte kjøring av spillet.)

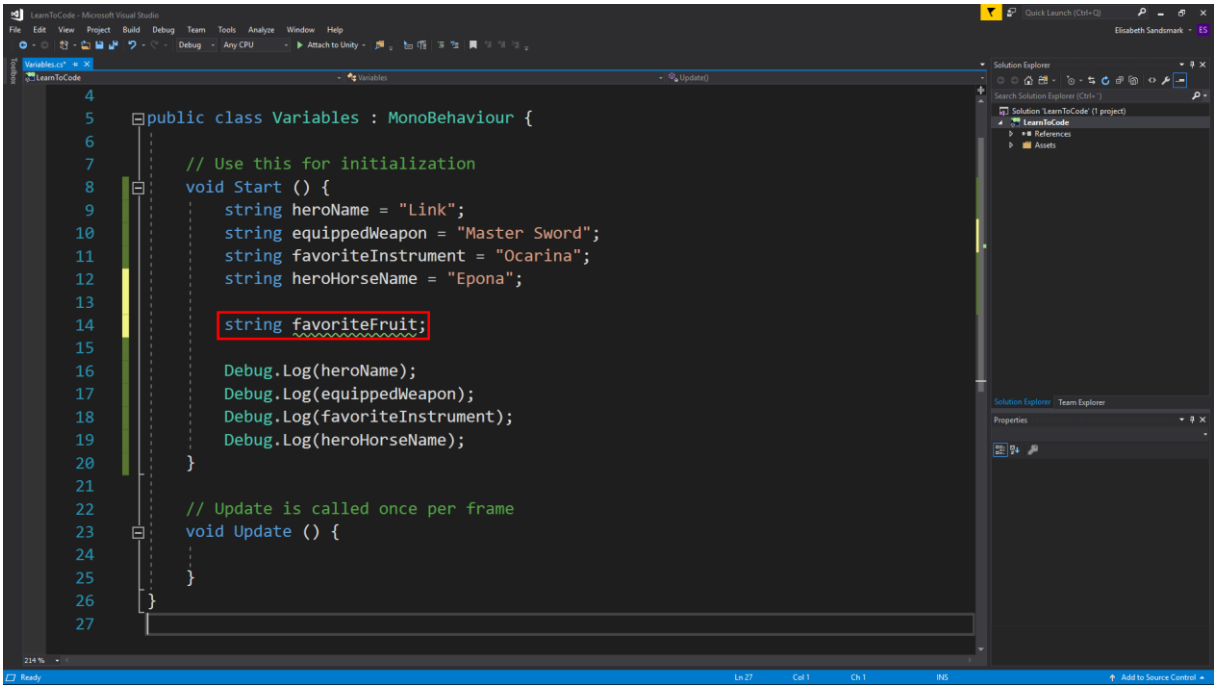
Legg merke til at det skrives ut alle variablene (Link, Master Sword, Ocarina og Epona dersom du ikke har laget dine egne variabler. Om du har laget dine egne variabler vil disse bli skrevet ut i stedet).

Kort oppsummert: Det du jobber med nå er data. Du spesifiserer en variabletype (eks. string), gir variablen et navn (eks. heroName) og gir den så en verdi (eks. Link).



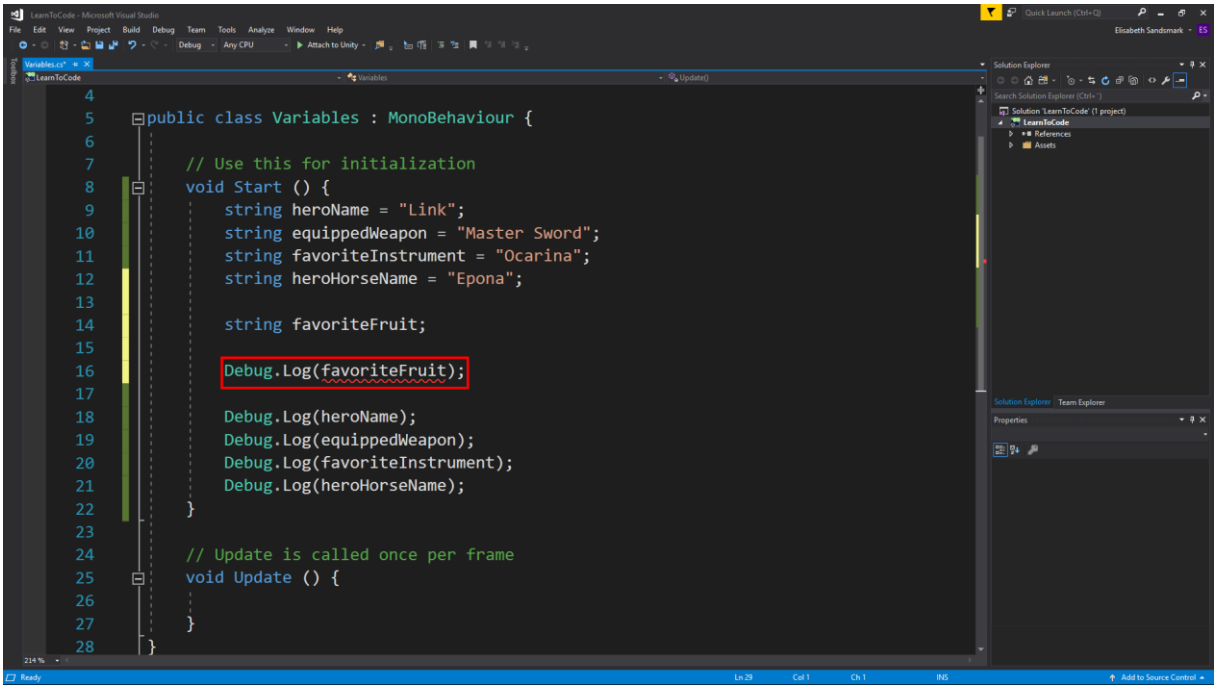
Du trenger ikke gi variablen en verdi med en gang. Kanskje har helten en favorittfrukt men vet ikke helt hvilken det er i starten. Eks:

```
string favoriteFruit;
```

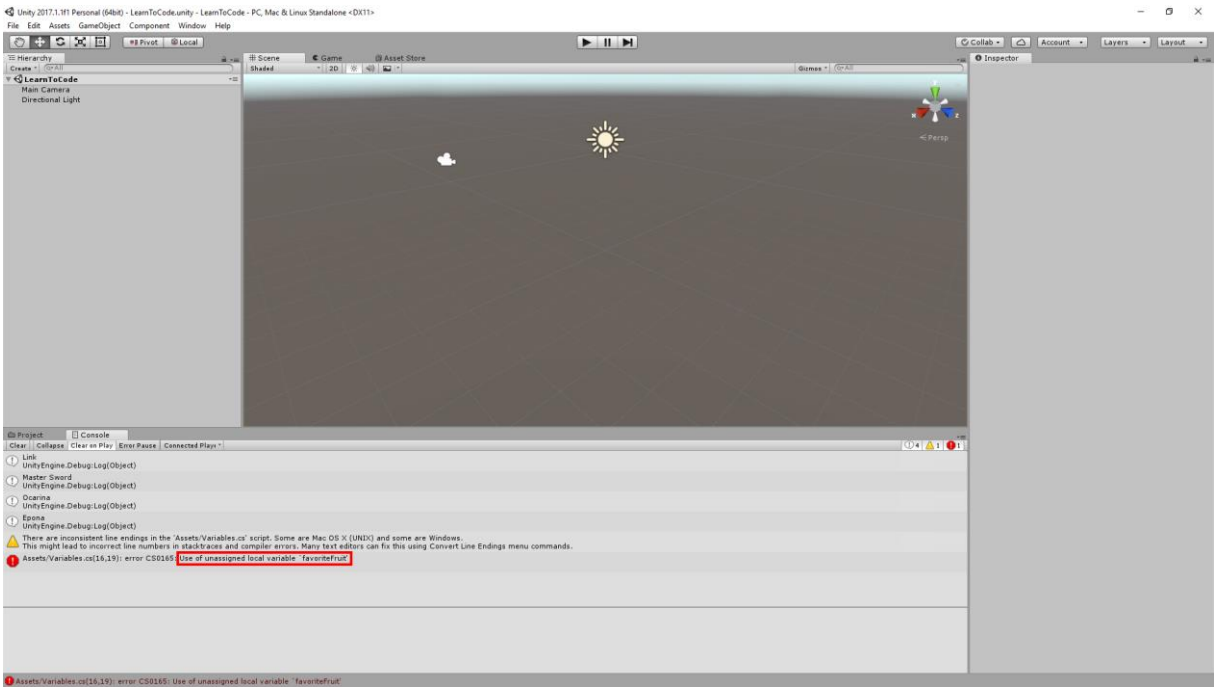


Du kan ikke hente denne informasjonen enda fordi vi ikke har gitt den en verdi. Hvis vi nå skriver:

Debug.Log(favoriteFruit);



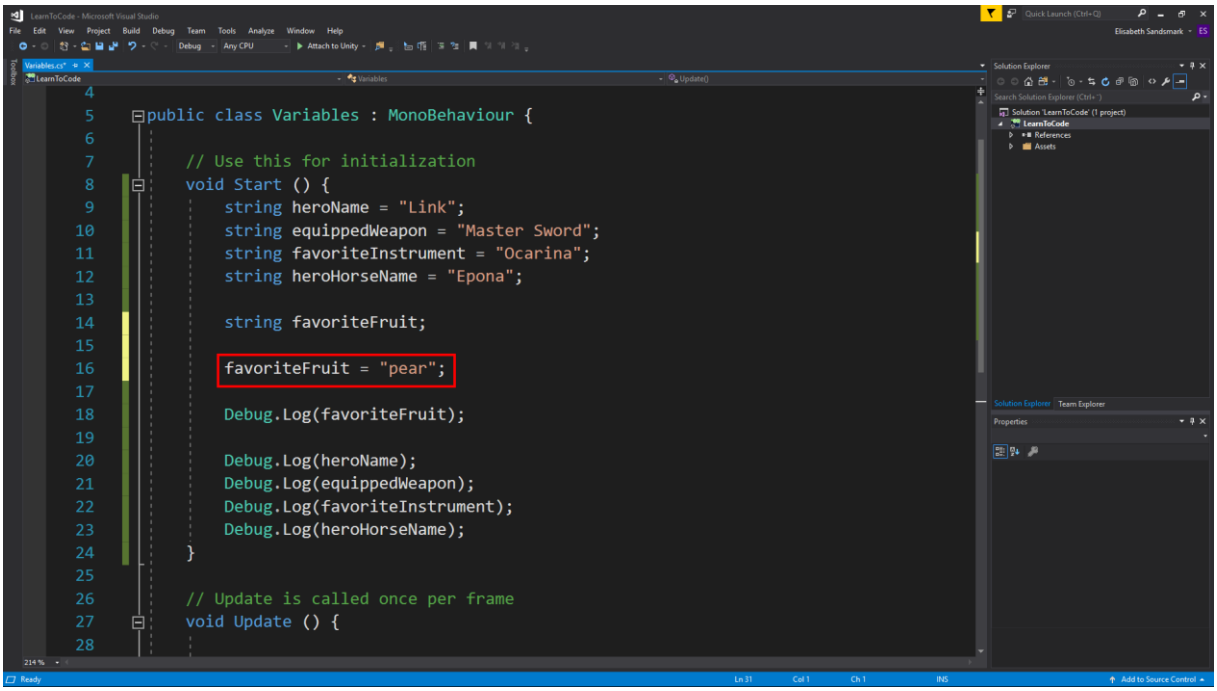
Vil vi få en feil «Use of unassigned local variable» dersom vi lagrer og prøver å kjøre programmet. Dersom du prøver å kjøre et spill som prøver å bruke en variabel som ikke har en verdi, vil spillet mest sannsynlig krasje eller ikke kjøre i det hele tatt. Maskinen blir forvirret fordi den leter etter en verdi som ikke eksisterer.



La oss si at vi seinere finner ut at denne favorittfrukten er pære (OBS: pass på å ikke bruke norske bokstaver eller andre tegn som ikke er i det engelske alfabetet a-z. Maskinen forstår som regel ikke hva du prøver å si og vi får feil når vi prøver å kjøre programmet).

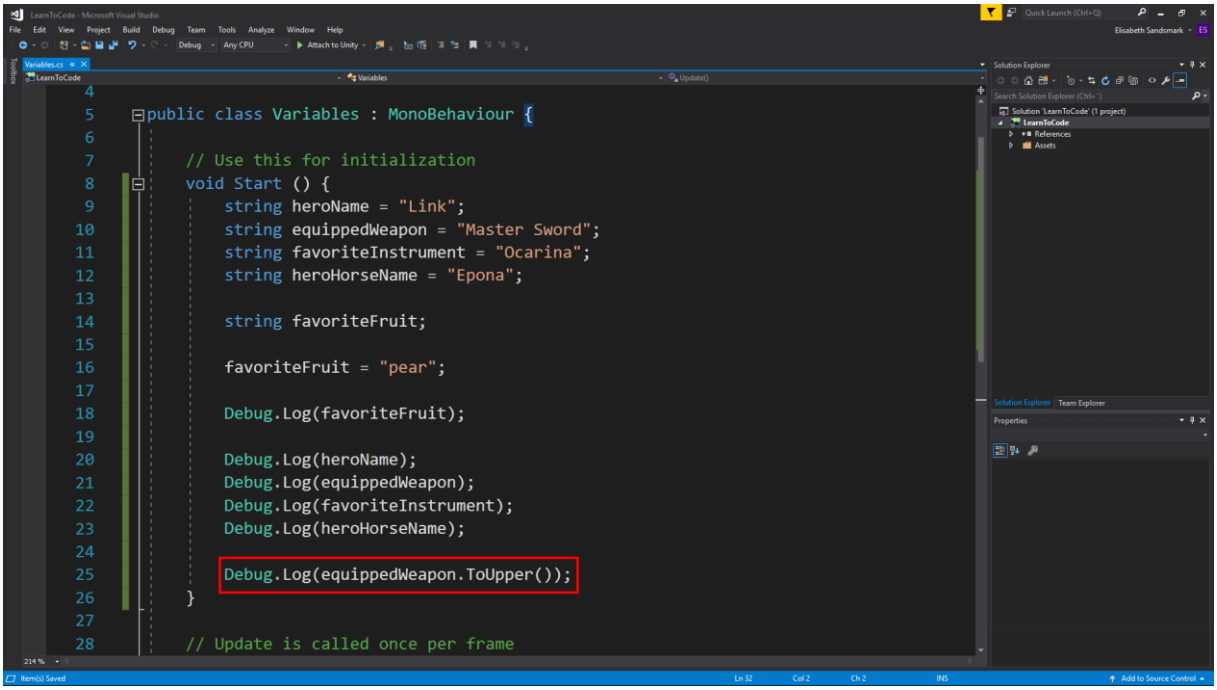
favoriteFruit = «pear»;

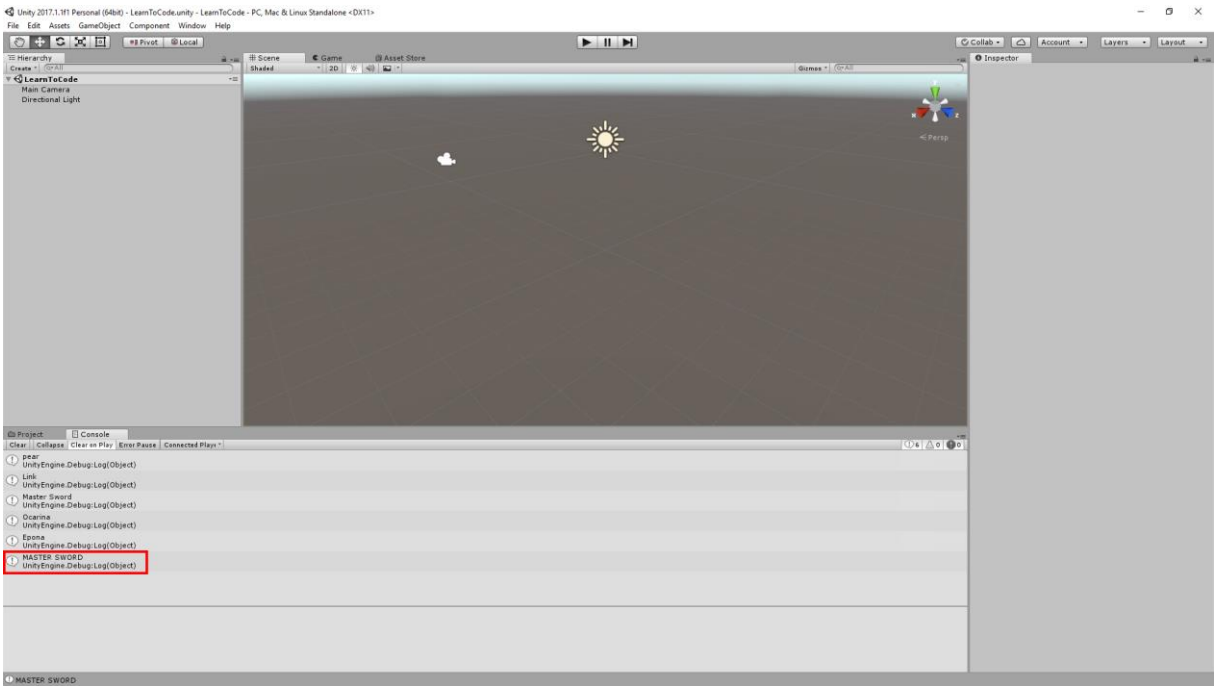
Dette må stå før *Debug.Log(favoriteFruit);* for at programmet skal kunne registrere at variablen faktisk har en verdi. Når maskinen leser kode, så leser den fra topp til bunn (linje 1 til linje x) med mindre vi spesifiserer noe annet underveis. Lagre og test at alt fungerer som det skal.



Det er mye vi kan gjøre med strings. For å ta et eksempel kan vi blant annet gjøre hele teksten i en string om til store bokstaver:

Debug.Log(equippedWeapon.ToUpper ());





Alt etter punktum «.» (eks .ToUpper, .IndexOf, .GetType) er forskjellige funksjoner du kan kalle på og bruke for å modifisere dine variabler. Prøv deg frem og sjekk på nettet hva de ulike funksjonene gjør. (Søk f.eks. på C# string ToUpper).

Full kode så langt til «Variables» (dersom du får problemer underveis):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Variables : MonoBehaviour {

        void Start () {
    string heroName = "Link";
    string equippedWeapon = "Master Sword";
    string favoriteInstrument = "Ocarina";
    string heroHorseName = "Epona";

    string favoriteFruit;

    favoriteFruit = "pear";

    Debug.Log(favoriteFruit);

    Debug.Log(heroName);
    Debug.Log(equippedWeapon);
    Debug.Log(favoriteInstrument);
    Debug.Log(heroHorseName);

    Debug.Log(equippedWeapon.ToUpper());
  }

        void Update () {

        }
}
```

Variabler – «int», «float» og «double» (tall)

Vi har nå sett en del på hvordan vi lager variabler som består av tekst, men dersom vi ønsker å lage variabler som består av tall må vi bruke en annen variabeltype. De mest brukte variabeltypene innenfor tall heter «int» (kort for integer), «float» og «double». «int» brukes kun for heltall, både positive og negative, mens «float» og «double» brukes begge for alle «reelle tall», altså både heltall og desimaltall, positive og negative.

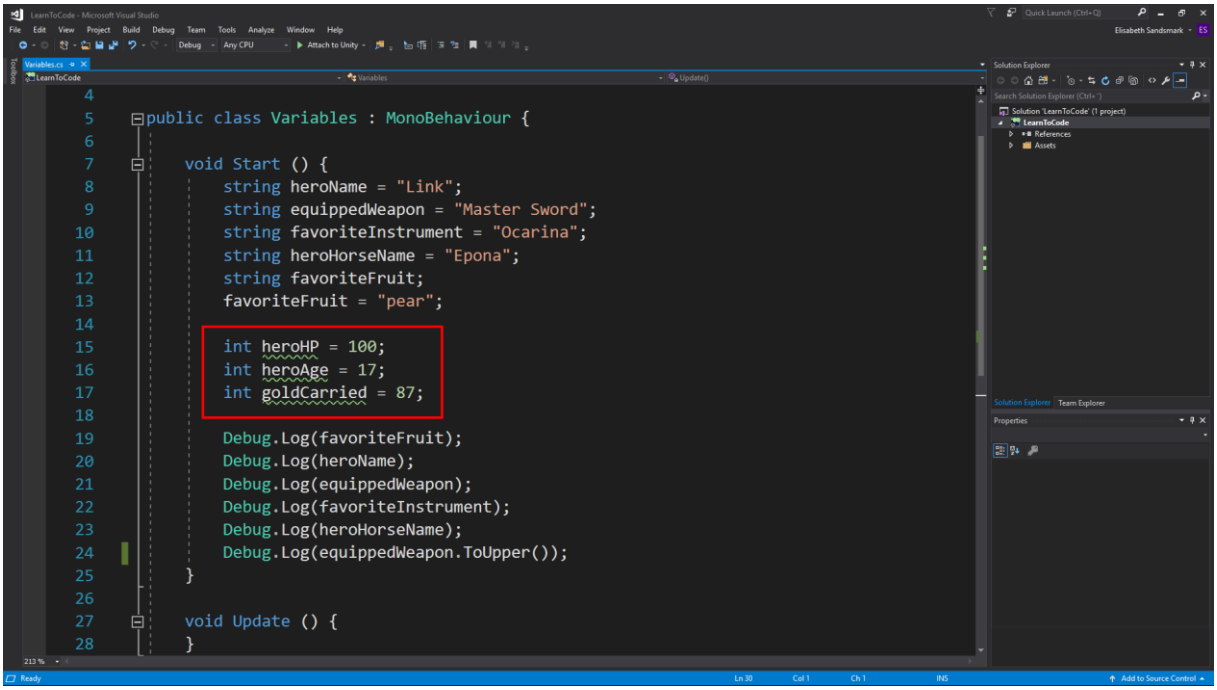
Hver av variabeltypene for tall har en grense for sitt høyeste og laveste tall. For «byte» vil dette være tall fra 0 til 255, mens for «int» strekker det seg helt fra -2,147,483,648 til 2,147,483,647. Det er derfor mer logisk å bruke int fremfor blant annet byte siden denne variabeltypen har en større bredde på høyeste og laveste tall vi kan bruke.

(Byte blir mer brukt i språk som C (ikke det samme som C#), i programmer hvor vi ønsker at ting skal gå så raskt som mulig. Siden byte har færre «adresser» å gå gjennom enn f.eks. en int har for å finne en lagret verdi i minnet vil programmet kunne kjøre rask! Til gjengjeld vil vi ikke kunne lagre like mange og like høye/lave verdier. Til spill bruker vi derfor som oftest int, float eller double i stedet. Dagens maskiner er raske nok til å takle dette uten store problemer!)

Vi legger til noen enkle «int»-variabler i koden vår (husk at variabler av typen int kun kan bestå av heltall):

```
int heroHP = 100;

int heroAge =17;

int goldCarried = 87;
```

(Husk at variablene er markert som følgende: variabeltype som blå, variabelnavn som gul og lagret verdi som grønn.)

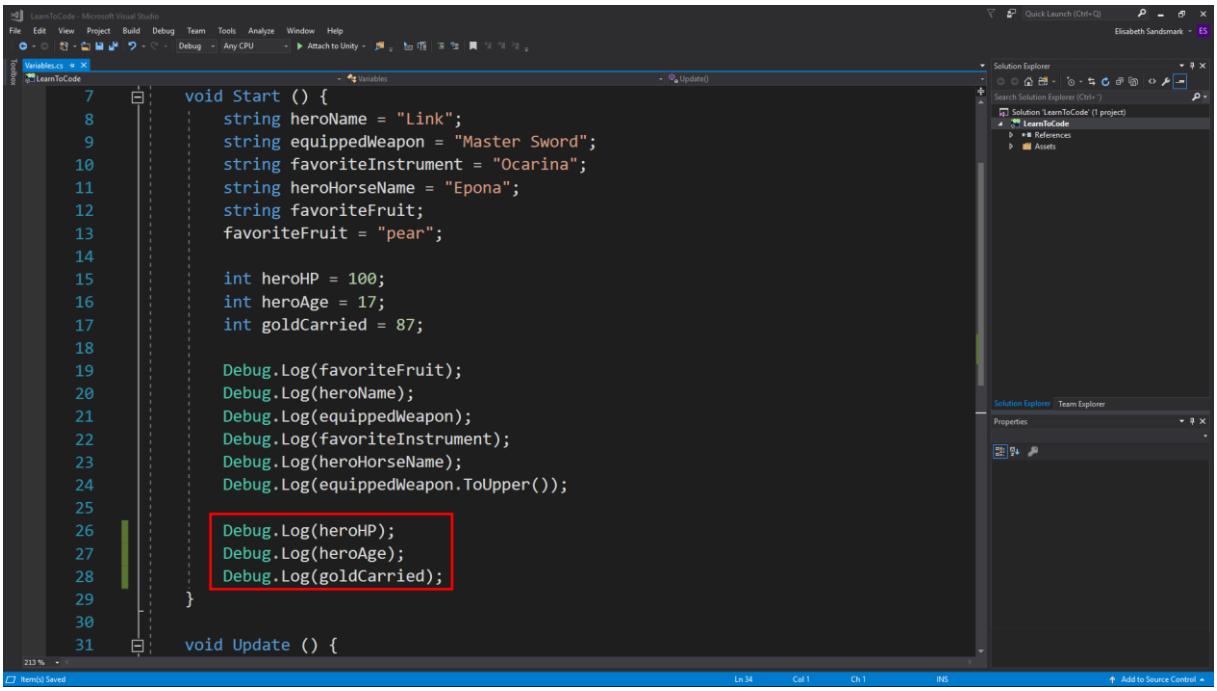


Legg til følgende for å skrive ut resultatet (3 tallverdier) til skjermen. Lagre og test.

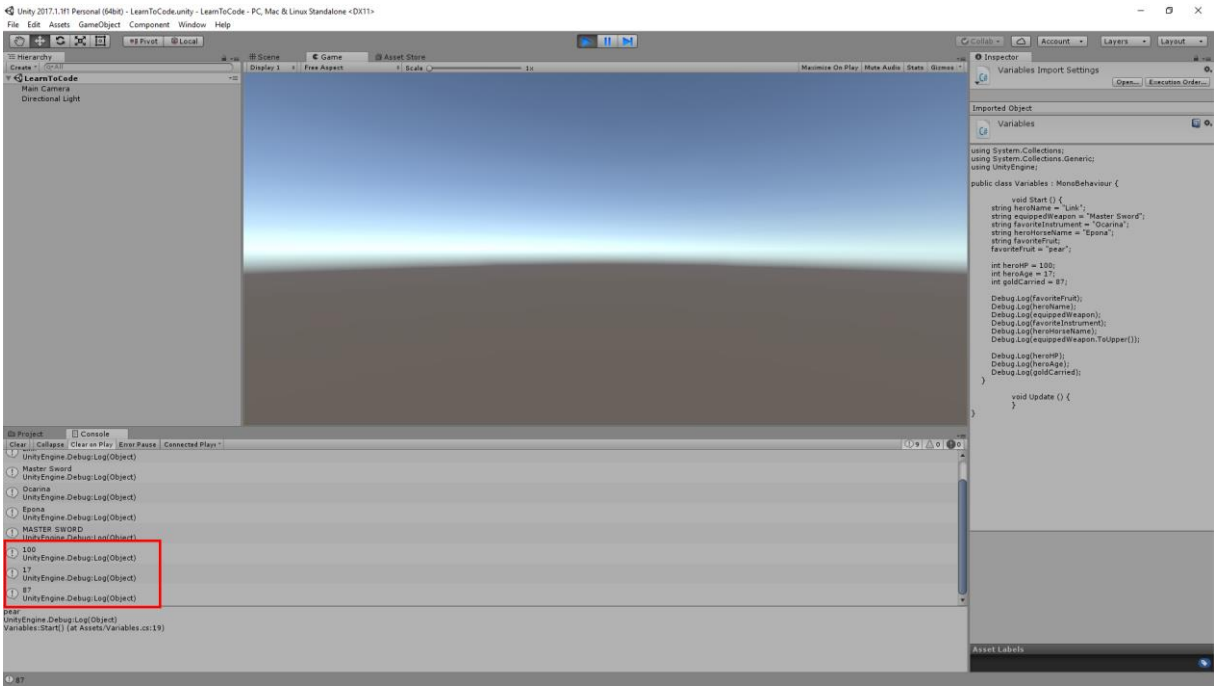
Debug.Log(heroHP);

Debug.Log(heroAge);

Debug.Log(goldCarried);



```
void Start () {
    string heroName = "Link";
    string equippedWeapon = "Master Sword";
    string favoriteInstrument = "Ocarina";
    string heroHorseName = "Epona";
    string favoriteFruit;
    favoriteFruit = "pear";

    int heroHP = 100;
    int heroAge = 17;
    int goldCarried = 87;

    Debug.Log(favoriteFruit);
    Debug.Log(heroName);
    Debug.Log(equippedWeapon);
    Debug.Log(favoriteInstrument);
    Debug.Log(heroHorseName);
    Debug.Log(equippedWeapon.ToUpper());

    Debug.Log(heroHP);
    Debug.Log(heroAge);
    Debug.Log(goldCarried);
}

void Update () {
```



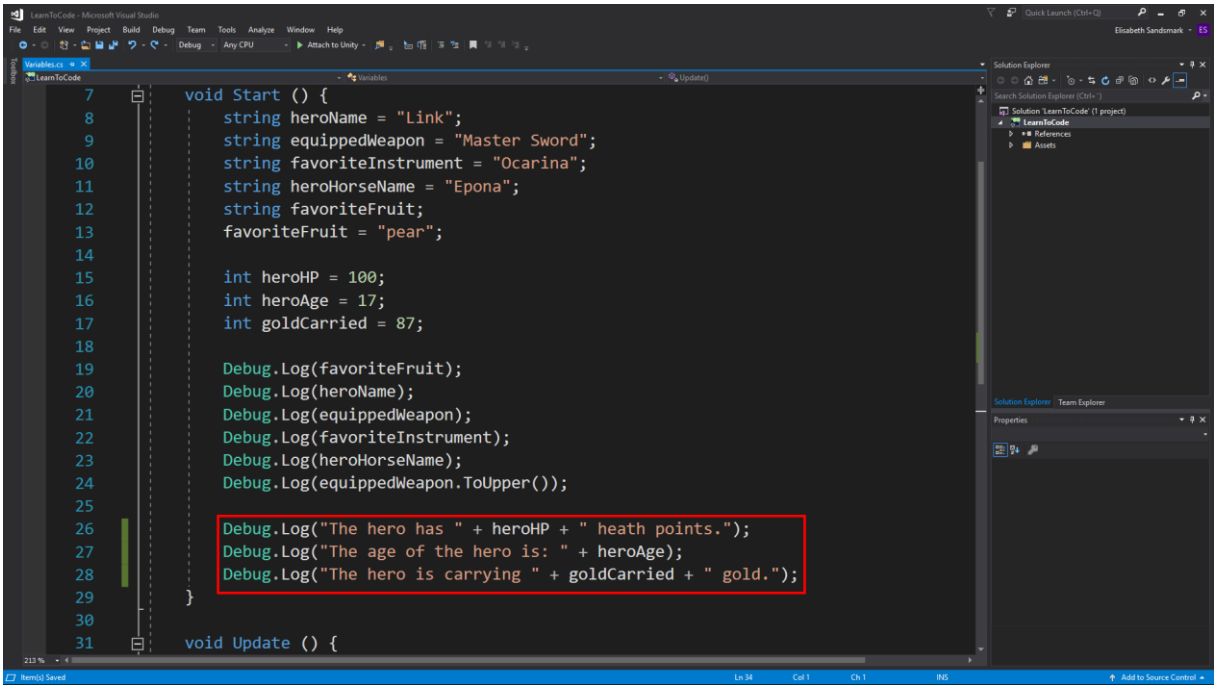
Vi kan også legge til litt tekst til det vi ønsker å skrive ut til skjermen slik at vi forstår hva tallene betyr. Endre teksten til følgende i koden din:

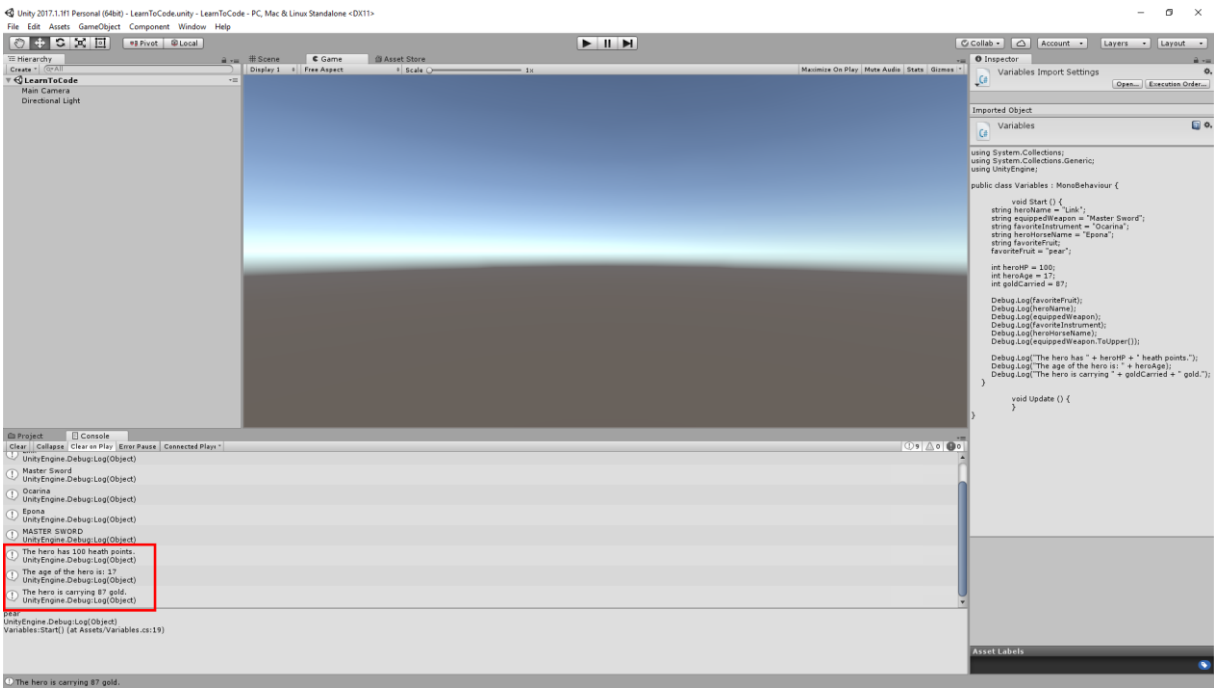
Legg merke til at all tekst må være skrevet i anførselstegn/sitattegn og at mellom teksten og variabelen må vi legge til et pluss tegn «+».

Debug.Log("The hero has " + heroHP + " heath points.");

Debug.Log("The age of the hero is: " + heroAge);

Debug.Log("The hero is carrying " + goldCarried + " gold.");





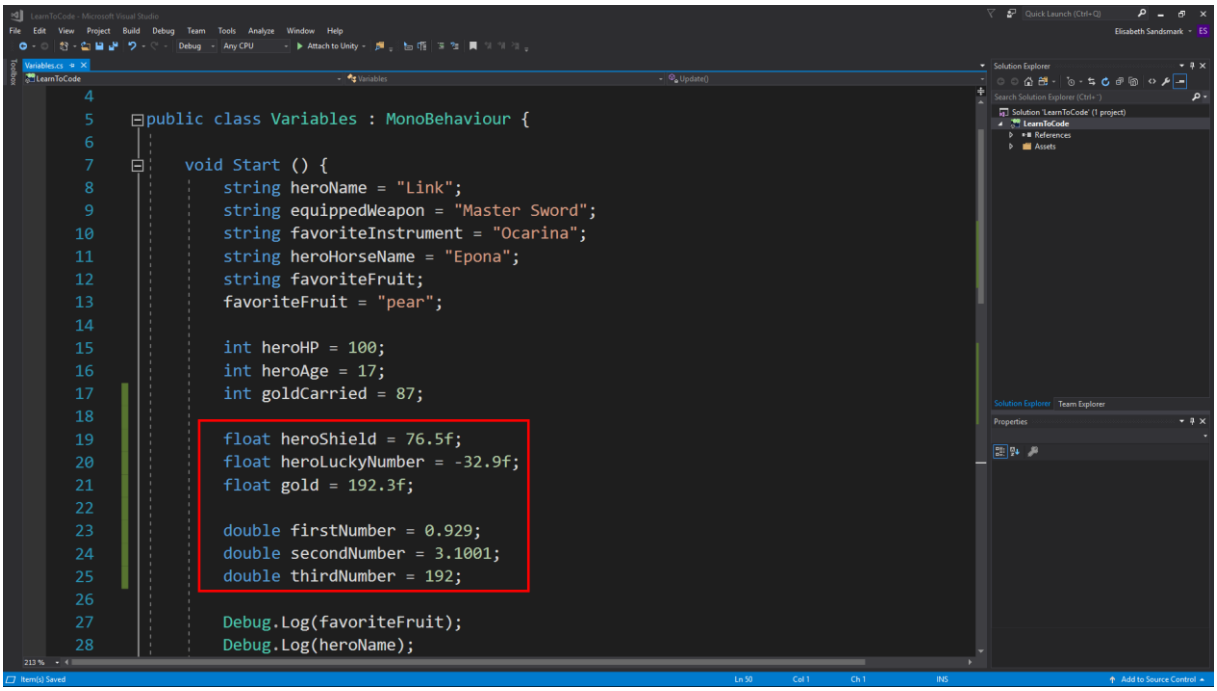
«float» og «double» er så å si det samme. Begge variabeltypene lagrer reelle tall (dvs. positive og negative heltall og desimaltall). Den største forskjellen er bare at «double» kan ha dobbelt så stor verdi som «float» kan. Grunnen til at jeg nevner begge er at mange som koder spill bruker «float» i stedet for eller i tillegg til «double», men det er helt ok å bruke kun «double». Svært mange bruker kun «double» også.

«float» kan ta tall fra -3.402823e38 til 3.402823e38 og
«double» kan ta tall fra -1.79769313486232e308 til 1.79769313486232e308.

En ting som er viktig å merke seg er at når du skriver desimaltall i koding, bruker vi punktum mellom heltallet og desimalen bak (eksempel: 2.45). På norsk bruker vi komma (eksempel: 2,45) og ikke punktum, men på de fleste andre språk brukes punktum! Komma brukes derimot for å skille mellom tusen (eks: 1,000,000 eller 1,000,000.00 = en million).

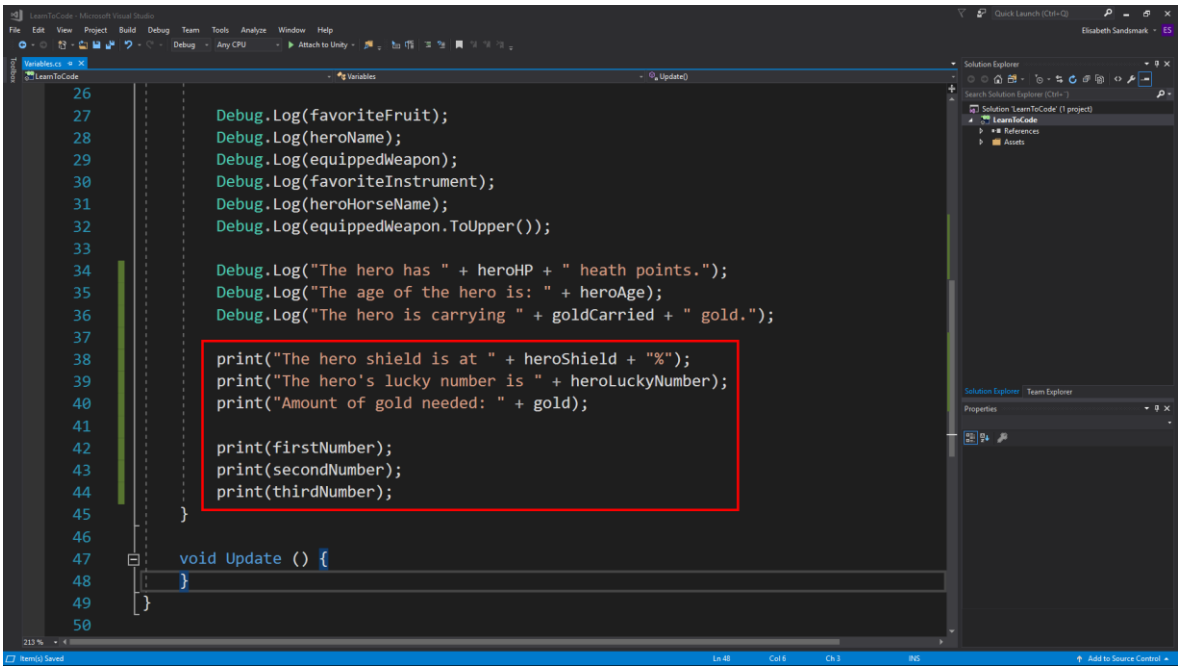
Her er noen eksempler på «float» og «double»:

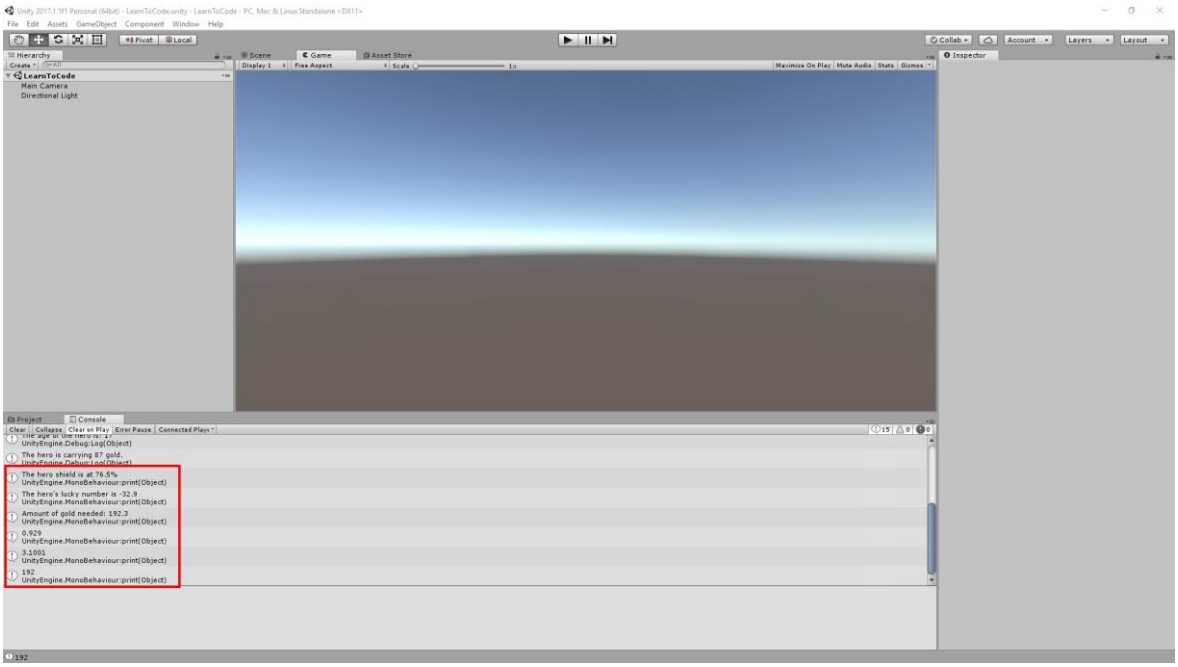
```
float heroShield = 76.5f;
float heroLuckyNumber = -32.9f;
float goldNeeded = 192.3f;

double firstNumber = 0.929;
double secondNumber = 3.1001;
double thirdNumber = 192;
```



Debug.Log brukes mye for å teste om du har «bugs» i spesielle deler av koden din og kan være svært nyttig. Et annet alternativ for å kun skrive ut f.eks. variabler fra koden din til konsollvinduet er «print»:

```
print("The hero shield is at " + heroShield + "%");
print("The hero's lucky number is " + heroLuckyNumber);
print("Amount of gold needed: " + goldNeeded);

print(firstNumber);
print(secondNumber);
print(thirdNumber);
```





Full kode så langt til «Variables» (dersom du får problemer underveis):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Variables : MonoBehaviour {

        void Start () {
    string heroName = "Link";
    string equippedWeapon = "Master Sword";
    string favoriteInstrument = "Ocarina";
    string heroHorseName = "Epona";
    string favoriteFruit;
    favoriteFruit = "pear";

    int heroHP = 100;
    int heroAge = 17;
    int goldCarried = 87;

    float heroShield = 76.5f;
    float heroLuckyNumber = 32.9f;
    float gold = 192.3f;

    double firstNumber = 0.929;
    double secondNumber = 3.1001;
    double thirdNumber = 192;

    Debug.Log(favoriteFruit);
    Debug.Log(heroName);
    Debug.Log(equippedWeapon);
    Debug.Log(favoriteInstrument);
    Debug.Log(heroHorseName);
    Debug.Log(equippedWeapon.ToUpper());

    Debug.Log("The hero has " + heroHP + " heath points.");
    Debug.Log("The age of the hero is: " + heroAge);
    Debug.Log("The hero is carrying " + goldCarried + " gold.");

    print("The hero shield is at " + heroShield + "%");
    print("The hero's lucky number is " + heroLuckyNumber);
    print("Amount of gold needed: " + gold);

    print(firstNumber);
    print(secondNumber);
    print(thirdNumber);
  }

        void Update () {
        }
}
```

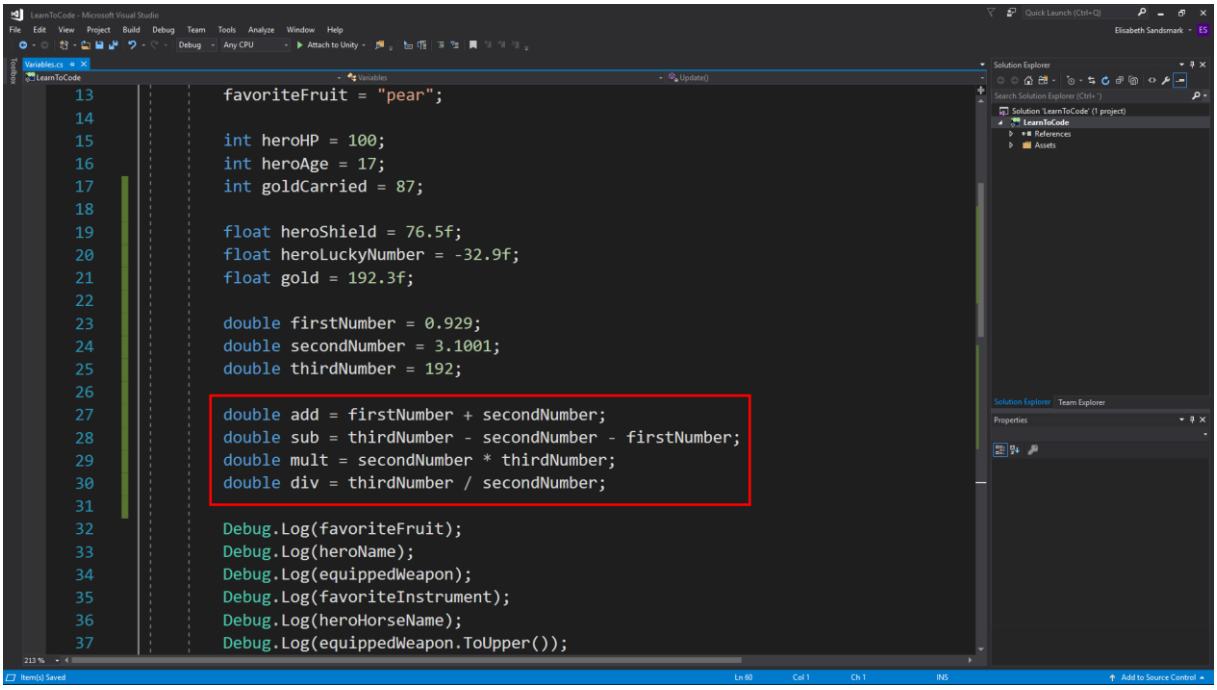
Videre kan vi bruke «double» variablene vi har til å gjøre litt enkel matte inne i koden. Du kan også lage dine egne variabler dersom du ønsker det, enten det er av typen «int», «float» eller annet. En ting som er verdt å merke seg er at dersom du legger sammen f.eks. en «int» og en «double» må du ha en «double» som svar for at koden skal fungere (1 + 2.43 = 3.43, eller «int» + «double» = «double»).

Her er noen eksempler ved bruk av «double»:

Vi må først deklarere variablene (dette har vi allerede gjort i forrige del eks. double firstNumber = 0.929;) Dermed kan vi bruke disse variablene til å addere, subtrahere, multiplisere og dividere på følgende måte (helt enkel matematikk).

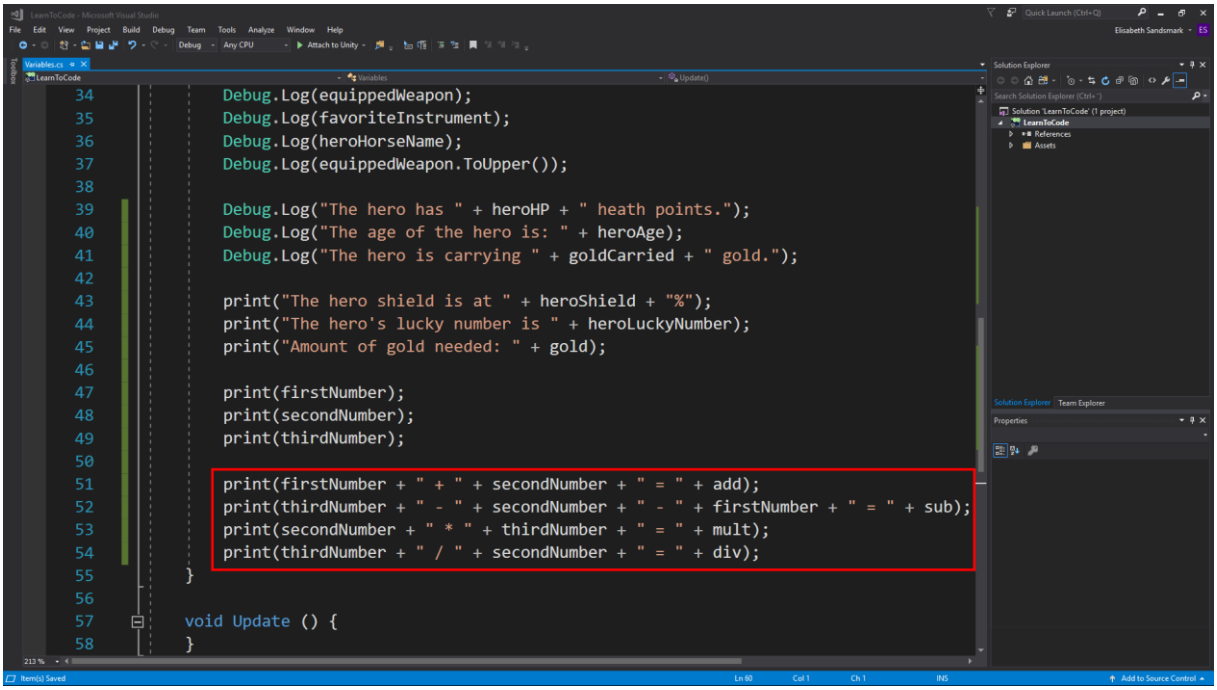
Vi kan kalle variabelen for addisjon for «add» for eksempel. Denne variabelen vil nå få den samlede verdien til «firstNumber» og «secondNumber»:

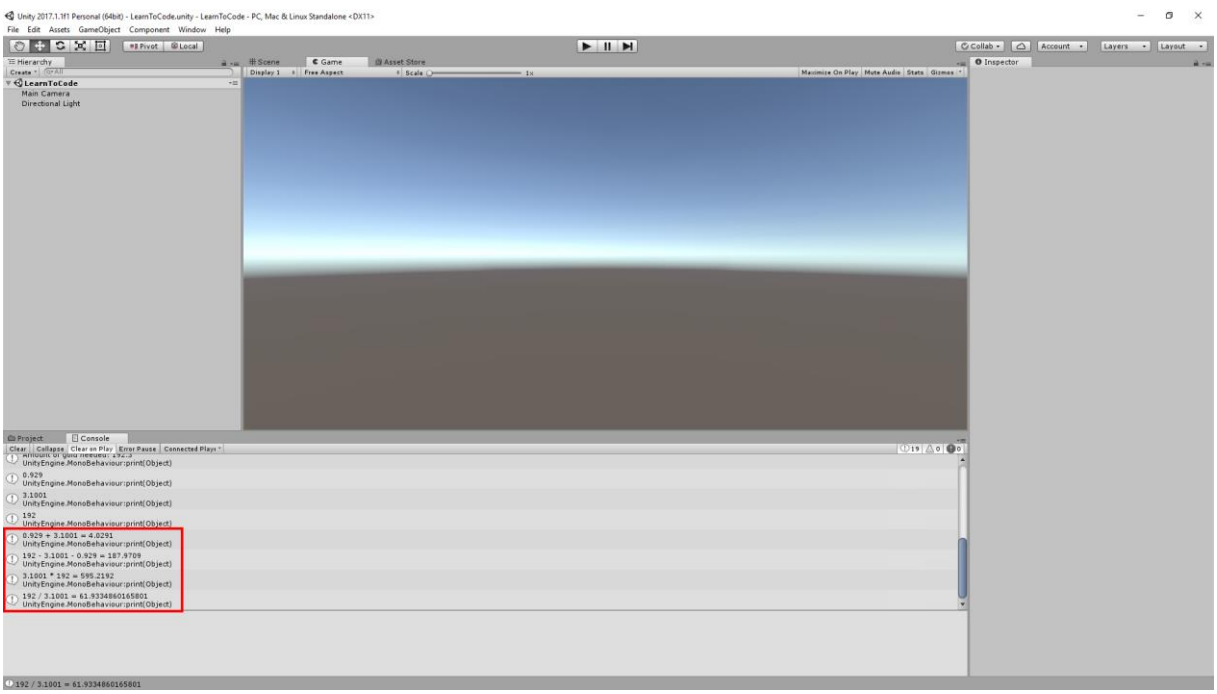
```
double add = firstNumber + secondNumber;
double sub = thirdNumber - secondNumber - firstNumber;
double mult = secondNumber * thirdNumber;
double div = thirdNumber / secondNumber;
```



Tegnene for regnestykkene (+, -, *, / og =) under «print» er der kun for at vi skal kunne se hele regnestykket og ikke bare tallene når vi skriver det ut til konsollvinduet. Dette gir oss bedre oversikt over hva som skjer.

```
print(firstNumber + " + " + secondNumber + " = " + add);
print(thirdNumber + " - " + secondNumber + " - " + firstNumber + " = " + sub);
print(secondNumber + " * " + thirdNumber + " = " + mult);
print(thirdNumber + " / " + secondNumber +  " = " + div);
```





Full kode så langt til «Variables» (dersom du får problemer underveis):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Variables : MonoBehaviour {

        void Start () {
    string heroName = "Link";
    string equippedWeapon = "Master Sword";
    string favoriteInstrument = "Ocarina";
    string heroHorseName = "Epona";
    string favoriteFruit;
    favoriteFruit = "pear";

    int heroHP = 100;
    int heroAge = 17;
    int goldCarried = 87;

    float heroShield = 76.5f;
    float heroLuckyNumber = -32.9f;
    float gold = 192.3f;

    double firstNumber = 0.929;
    double secondNumber = 3.1001;
    double thirdNumber = 192;

    double add = firstNumber + secondNumber;
    double sub = thirdNumber - secondNumber - firstNumber;
    double mult = secondNumber * thirdNumber;
    double div = thirdNumber / secondNumber;

    Debug.Log(favoriteFruit);
    Debug.Log(heroName);
    Debug.Log(equippedWeapon);
    Debug.Log(favoriteInstrument);
    Debug.Log(heroHorseName);
    Debug.Log(equippedWeapon.ToUpper());

    Debug.Log("The hero has " + heroHP + " heath points.");
    Debug.Log("The age of the hero is: " + heroAge);
    Debug.Log("The hero is carrying " + goldCarried + " gold.");

    print("The hero shield is at " + heroShield + "%");
    print("The hero's lucky number is " + heroLuckyNumber);
    print("Amount of gold needed: " + gold);

    print(firstNumber);
    print(secondNumber);
    print(thirdNumber);

    print(firstNumber + " + " + secondNumber + " = " + add);
    print(thirdNumber + " - " + secondNumber + " - " + firstNumber + " = " + sub);
    print(secondNumber + " * " + thirdNumber + " = " + mult);
    print(thirdNumber + " / " + secondNumber + " = " + div);
  }

        void Update () {
        }
}
```

Flere variabler /data typer (data types):

byte (0 til 255)
sbyte (-128 til 127)
int (-2,147,483,648 til 2,147,483,647)
uint (0 til 4294967295)
short (-32,768 til 32,767)
ushort (0 til 65535)
long (-9223372036854775808 til 9223372036854775807)
ulong (0 til 18446744073709551615)
float (-3.402823e38 til 3.402823e38)
double (-1.79769313486232e308 til 1.79769313486232e308)
char («Unicode» symboler brukt i tekst)
bool (True eller false)
object ()
string ()
decimal (±1.0 × 10e−28 to ±7.9 × 10e28)

For mer informasjon kan du f.eks. sjekke:
https://msdn.microsoft.com/en-us/library/ms228360%28v=vs.90%29.aspx?f=255&MSPPError=-2147217396