Introduktion til programmering

Faglige læringsmål

* Kendskab til Visual Studio 2019
* At kunne lave mit første simple program
* At kunne oversætte og køre et C#-program i debug mode

Konkrete elementer

* Opsætning af Visual Studio
* Mit første C#-program
* Afvikling af programmer med debugger

Forudsætninger

* Eleven er indstillet på at lære at arbejde med C# og Visual Studio

Indhold

[1. I gang med Visual Studio 3](#_Toc529513404)

[1.1. Tilføj til proceslinje 3](#_Toc529513405)

[1.2. Første gang du starter Visual Studio 3](#_Toc529513406)

[2. Opret nyt projekt 4](#_Toc529513407)

[2.1. Indstilling af default folder 6](#_Toc529513408)

[3. Find rundt i det nye projekt 7](#_Toc529513409)

[3.1. Hello world og ‘;’ 8](#_Toc529513410)

[3.2. Afbryd programmet 9](#_Toc529513411)

[3.3. Find Solution Explorer (igen) 9](#_Toc529513412)

[4. Introduktion til debug 10](#_Toc529513413)

[4.1. Breakpoints 11](#_Toc529513414)

[4.2. At steppe sig gennem programmet 11](#_Toc529513415)

[4.3. Vigtige debug-vinduer 12](#_Toc529513416)

[4.3.1. Vinduet Autos 12](#_Toc529513417)

[4.3.2. Vinduet Locals 14](#_Toc529513418)

[4.3.3. Watch-vinduerne 15](#_Toc529513419)

[4.3.4. Vinduet Call Stack 15](#_Toc529513420)

# Version 2017 vs. 2019

Denne introduktion til Visual Studio er baseret på Visual Studio 2017. Den burde benytte 2019 (jeg retter den til når jeg får tid). Det betyder at udklippene ser lidt anderledes ud i dette dokument end det gør i Visual Studio 2019, så prøv at abstrahere fra det. Bed om hjælp hvis du bliver forvirret.

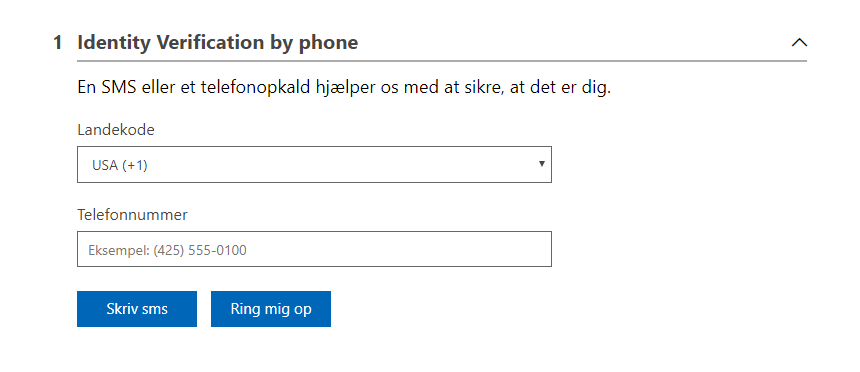
# Installation af Visual Studio 2019

Undersøg først om *Visual Studio 2019 Enterprise edition* er installeret på din computer. Det er ikke sikkert da maskinerne tidligere som standard blev installeret med 2017.

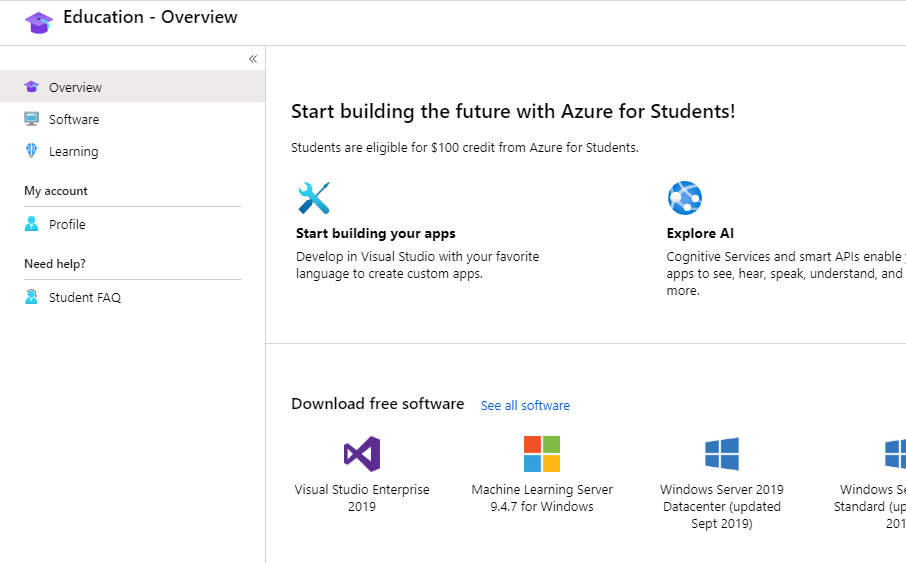
Hvis du har den nye version, så spring dette afsnit over, ellers skal du have installeret *Visual Studio 2019 Enterprise edition.*

1. Gå ind på *dreamspark.campusvejle.dk*. Du bliver videredirigeret til *Microsoft Azure*.
2. Sign in med din campusvejle login.

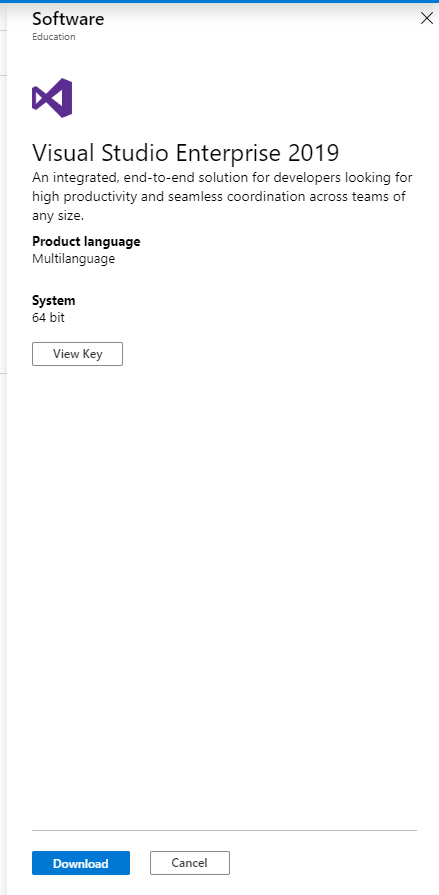
Du bliver bedt om et telefonnummer til en SMS-verifikation (sikkerhed for Microsoft).



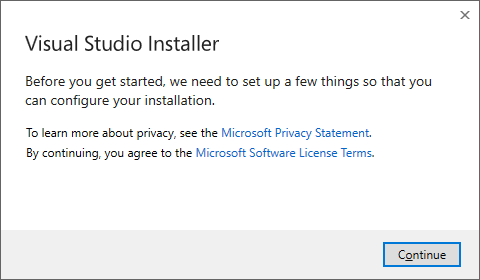
1. Vælg *Visual Studio Enterprise 2019*



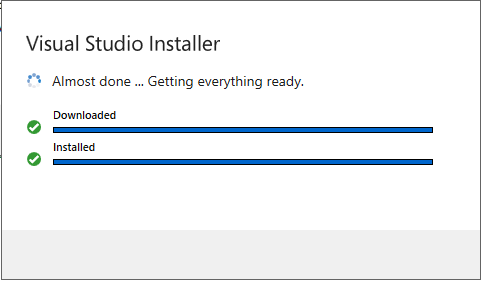
1. Kopiér nøglen og tryk på Download



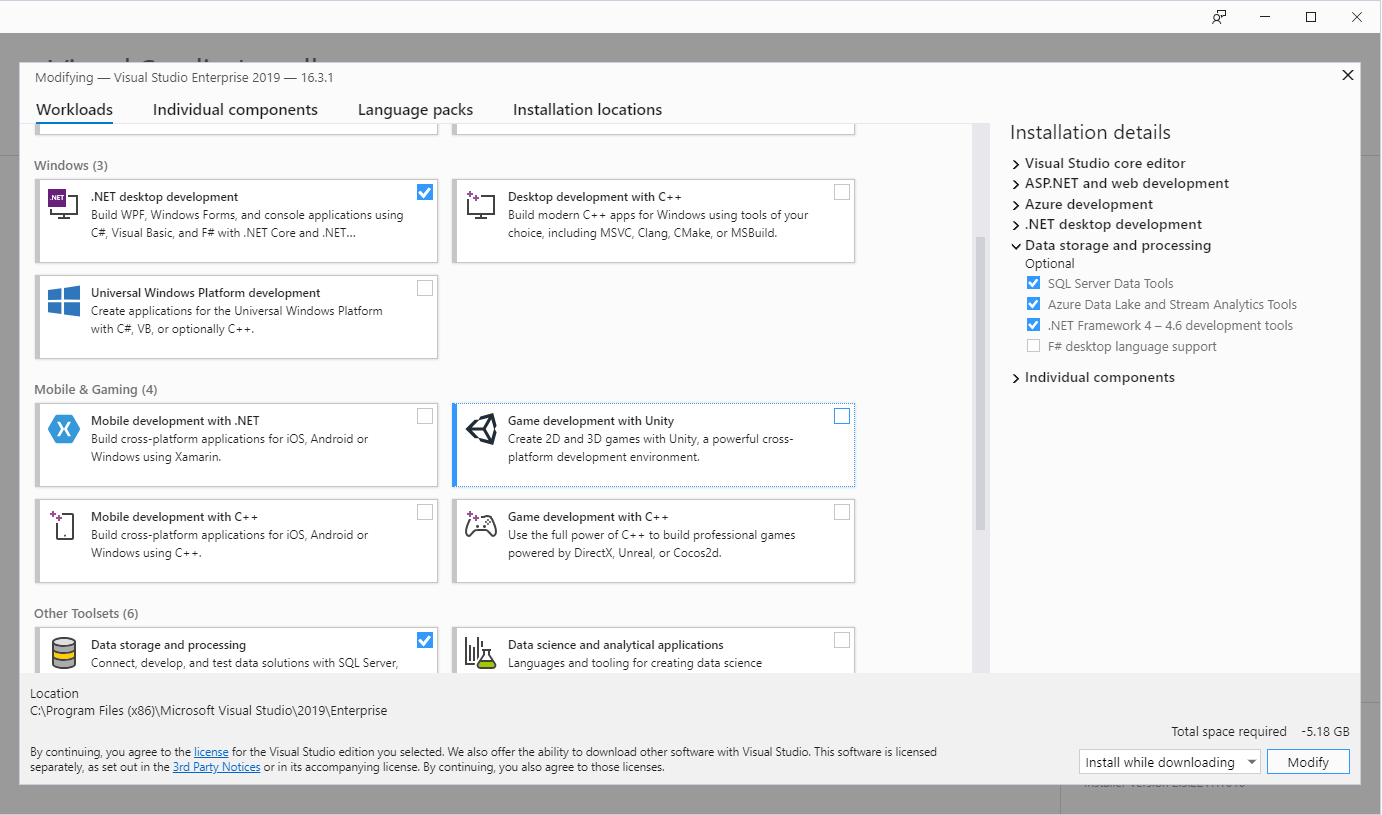
1. Kør installationsprogrammet



Tryk *Continue*.

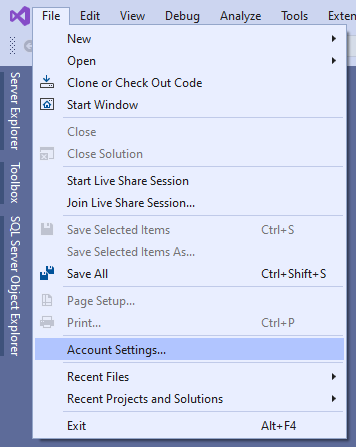


1. Vælg at installere *.NET desktop development* og *Data storage and processing*.[[1]](#footnote-1)

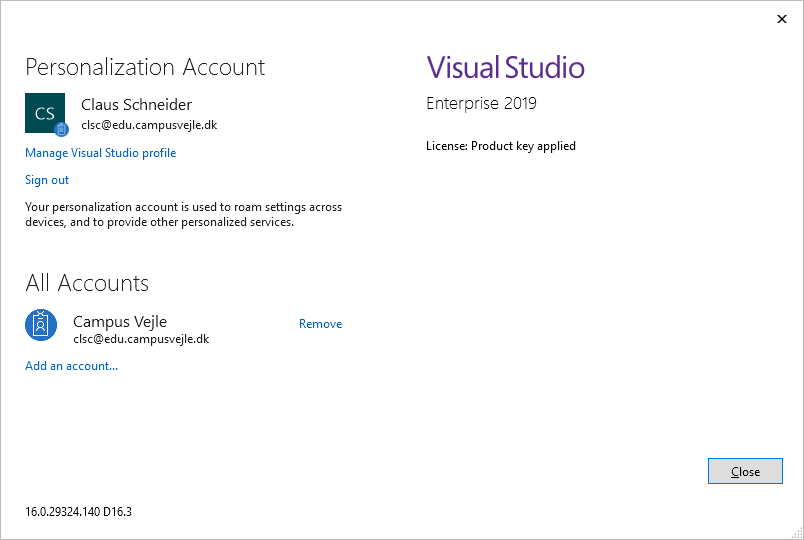


1. Start Visual Studio 2019 op

Vælg *Account Settings …*



I højre side af dialogen kan licens-nøglen indsættes. I dialogen nedenfor er det allerede sket.

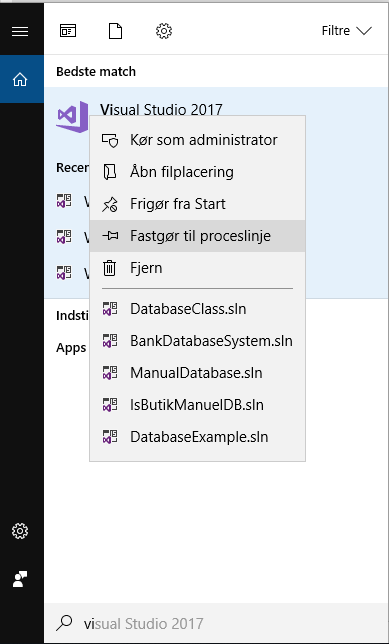


Nu skulle du være klar til at begynde at arbejde med Visual Studio 2019.

# I gang med Visual Studio

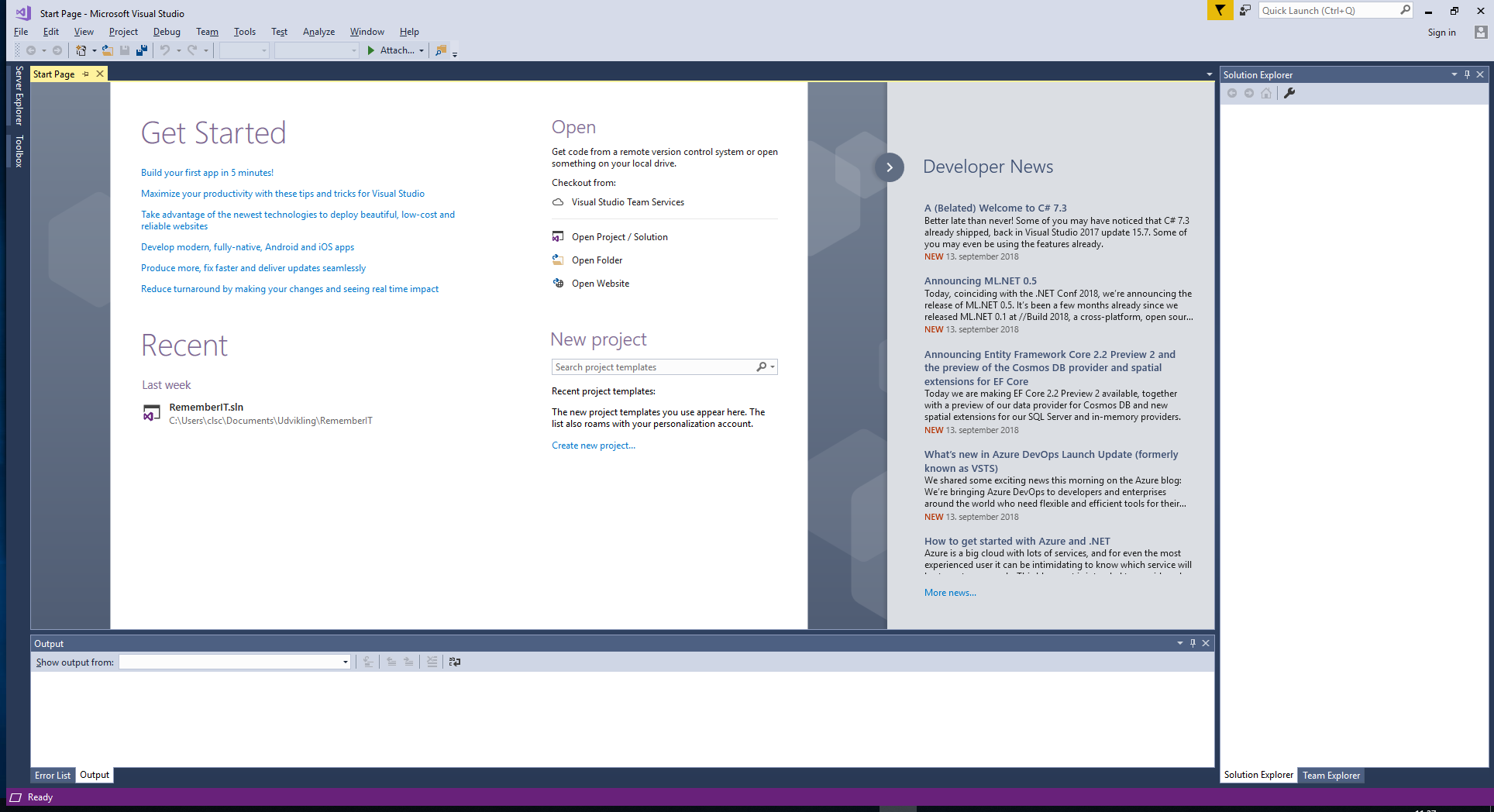
## Tilføj til proceslinje

Vi skal arbejde meget med Visual Studio, så det er rarest hvis det er nemt at starte programmet op. Søg det derfor frem og fastgør det til proceslinjen. Så ligger der altid en knap klar i proceslinjen til at starte Visual Studio op.



## Første gang du starter Visual Studio

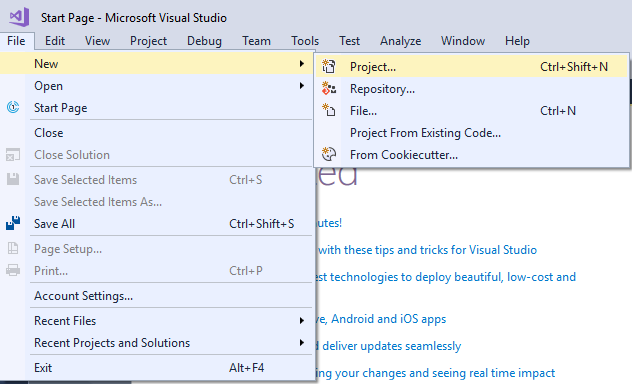
Når Visual Studio startes op er der lidt hjælp til at komme i gang på opstartsbilledet. Øverst i venstre side findes nogle links til vejledninger så man hurtigt kan komme i gang under overskriften **Get Started**. Nederst i venstre side er en oversigt over de seneste projekter man har arbejdet med under overskriften **Recent**. I eksemplet kan man se at jeg har arbejdet med et projekt der hedder RememberIT. I højre side er der genveje til at åbne projekter og foldere og til at oprette nye projekter. Disse funktioner er også tilgængelige i programmets menustruktur.



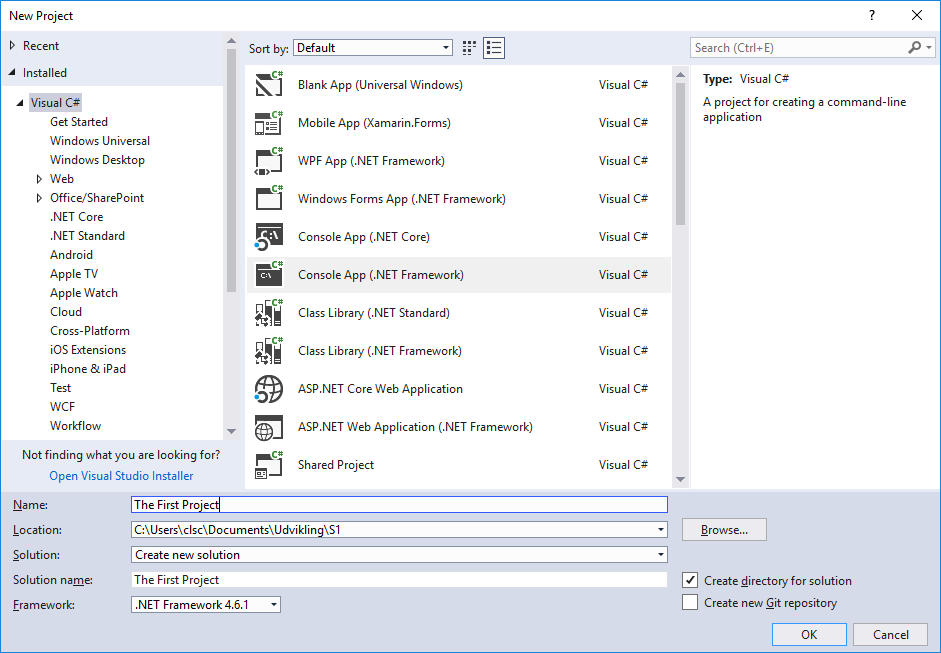
For at lære menustrukturen at kende, vælger vi her i første omgang at finde funktionerne i menuerne i stedet for at benytte genvejene i opstartsbilledet.

# Opret nyt projekt

Vælg i menuen **File/Project…** for at lave et nyt projekt.



Nu åbnes nedenstående dialog hvor man skal tage stilling til hvilken type projekt man vil oprette. I første omgang laver vi det mest simple program, en console App. Framework-versionen vælges fordi det automatisk giver adgang til mange af de biblioteker vi skal arbejde med.



Projektet skal gives et navn, og man skal bestemme hvor det skal ligge. Visual Studio vil som udgangspunkt gerne lægge projekterne i folderen ”…source\repos”. Hvis du vil have dine projekter til at ligge et andet sted (f.eks. et sted under dokumenter), kan du rette denne sti. Man kan også ændre indstillingerne sådan at Visual Studio foreslår en anden sti (se afsnit 2.1 nedenfor).

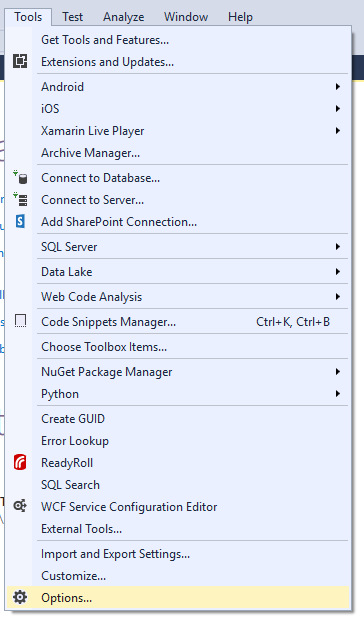
Giv projektet et godt navn, det vil sige et navn der fortæller noget om hvad projektet går ud på, sådan at det er til at finde igen senere. Dette er vores første projekt for lige at lære Visual Studio at kende, så det kan være lidt svært at finde på et godt navn. Så kald det bare dit første projekt. Det vil du så kunne hente ind senere hvis du har brug for at se hvordan det nu var man lige kom i gang med Visual Studio.

Bemærk, når du giver projektet et navn, så skrives automatisk det samme i feltet **Solution name**. Vi skal senere arbejde mere med solutions hvor vi vil have flere projekter i en samlet løsning (solution), men her i første omgang vil vi bare have ét projekt, derfor giver det god mening at solution og projekt har samme navn.

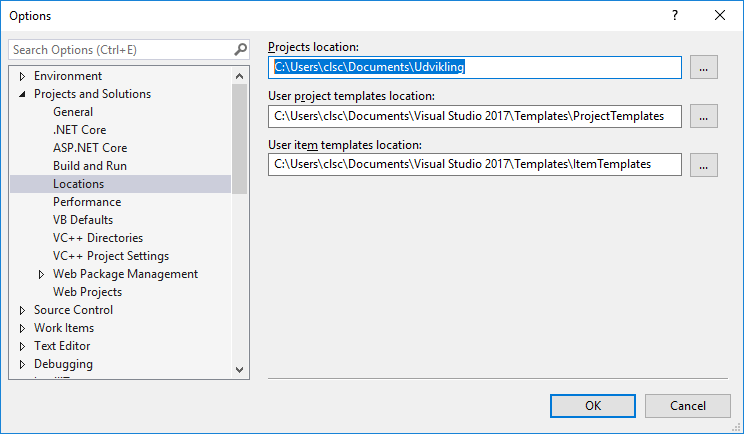
## Indstilling af default folder

I dette afsnit beskrives kort hvordan man ændrer indstillingerne af default folder til projekter. Du kan springe afsnittet over i første omgang hvis du tænker at Visual Studio sikkert er bedst til at vide hvor dine projekter skal gemmes.

Vælg Tools/Options i menuen øverst i Visual Studio:

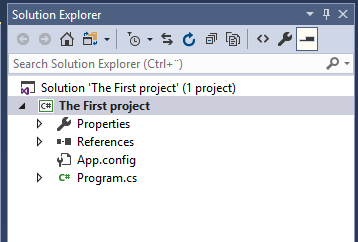


Vælg eller skriv nu den folder hvor du vil gemme dine projekter.



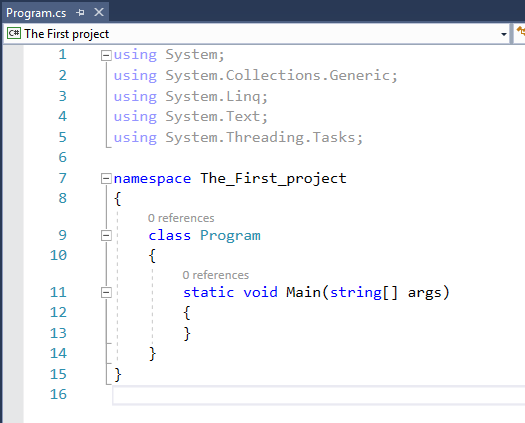
# Find rundt i det nye projekt

Visual Studio har lavet nogle filer til dig automatisk, sådan at det er nemt at komme i gang. I vinduet Solution Explorer kan du finde en oversigt over de dele dit projekt består af. Hvis du ikke kan finde Solution Explorer, så er der anvisninger i afsnit 3.3 nedenfor.



Som du kan se, har vi en solution med et projekt. Her hedder begge The First Project, de hedder sikkert noget andet hos dig. I første omgang springer vi direkte ned i filen Program.cs hvor selve programmet ligger. Visual Studio har oprettet et tomt program der ikke gør noget som helst, vi skal prøve at få det til at lave lidt. Den sidste del af filnavnet ”cs” er en forkortelse af C-Sharp og fortæller altså at filen indeholder C#-kode.

Dobbeltklik på Program.cs. Nu åbnes filen i arbejdsområdet, det store område midt på skærmen hvor du normalt vil kunne redigere dine filer. Med tiden vil du placere forskellige vinduer rundt omkring på skærmen i Visual Studio efterhånden som du finder ud af hvilke vinduer du bruger ofte.



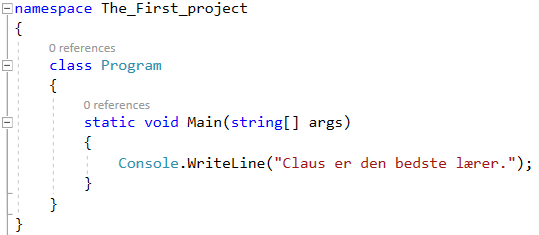
Programmet består af tre elementer der er placeret inden i hinanden. Det gøres tydeligt af indrykningen, men indrykningerne har kun kosmetisk betydning. Det er de krøllede parenteser der fortæller hvilke dele af et program der er defineret inde i andre dele. Hver gang en del begynder laves en venstre-krølle-parentes. Delen sluttes så af med en højre-krølle-parentes.

Vi kan altså læse det automatisk genererede program som følger. Programmet består af et name space der hedder det samme som projektet. Dette name space indeholder en class der hedder Program. class Program indeholder en metode (kaldes også en procedure eller en funktion) der hedder Main. Vi kommer senere til lære hvordan man kan se det er en metode, men det vil vi ikke gå i detaljer med nu.

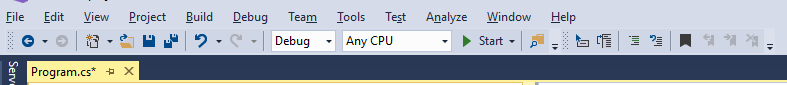
Det eneste du skal vide er, at metoden Main automatisk bliver udført når man starter programmet, så hvis vi vil have programmet til at gøre noget, skal vi skrive noget C#-kode her. Som man kan se er Main-metoden tom fra starten (der står ikke noget mellem ’{’ og ’}’).

## Hello world og ‘;’

Vi skal du lave vores første program, det man ofte kalder ”hello world”. Med et simpelt program vil vi teste om vi har fået lavet et program der kan køre. Programmet nedenfor udskriver en måske ikke helt sandfærdig tekst. Du kan skrive din egen tekst som du foretrækker.



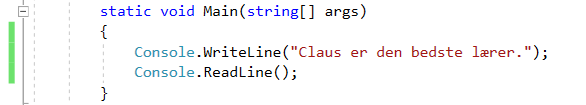
Når du har indtastet koden, skal du prøve at køre programmet. Til det kan du anvende den grønne pil øverst i Visual Studio:



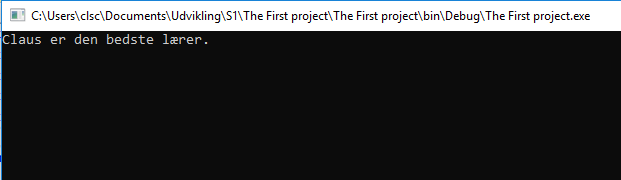
Hov, hvad skete der? Ingen ting?

Hvis du så godt efter, så kom din tekst ganske sikkert frem i et sort konsolvindue, men straks efter lukkede vinduet igen. Det var fordi programmet var færdigt (det nåede den sidste ’}’). For at vi kan nå at se hvad der sker, bliver vi nødt til at bede programmet vente lidt når det er færdigt.

Indtast følgende:



Linjen Console.ReadLine(); er en instruktion til programmet om at tage imod input som brugeren indtaster. Det betyder at når programmet når til dette sted, så venter det på input fra brugeren og derfor slutter programmet først af når brugeren har givet sit input og trykket Enter. Prøv nu at køre programmet.



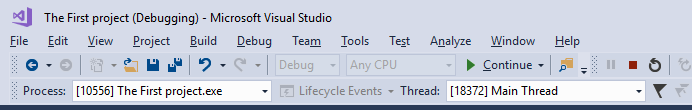
Når du kører programmet skulle du gerne få din besked op i et konsol-vindue. Tryk Enter for at afslutte programmet.

Måske har du bemærket at de to linjer kode vi har skrevet afsluttes med et ’;’. Dette tegn bruges til at adskille to instruktioner. Som ved indrykninger bruges linjeskift kun kosmetisk, det er tegnet ’;’ der adskiller instruktioner. Det betyder at man i princippet kan lave hele sit program på én linje hvis man har en teksteditor der understøtter meget lange linjer. I praksis benyttes linjeskift og indrykninger i stor stil for at gøre programmet læsevenligt. Det bliver hurtigt klart at det er meget vigtigt. Men husk altid ’;’.

## Afbryd programmet

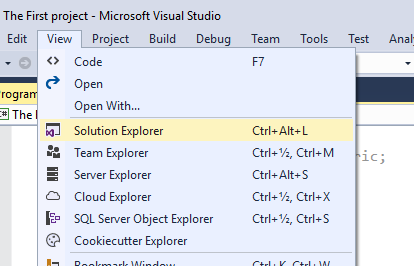
Nogle gange har man brug for at stoppe programmet midt i det hele i stedet for at lukke det pænt ned, eller man har lavet et program der aldrig standser.

Man kan afbryde program inde fra Visual Studio ved at trykke på den røde stop-firkant i toppen af skærmen:



## Find Solution Explorer (igen)

Hvis du ikke har noget Solution Explorer vindue åbent (eller du er kommet til at lukke det), kan du åbne det ved at finde det i **View**-menuen:

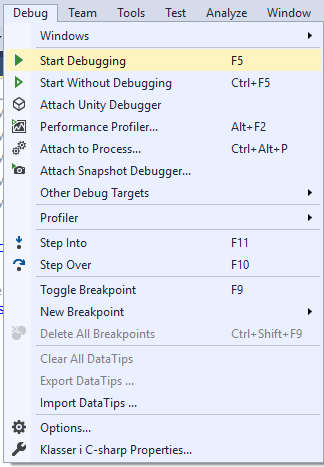


# Introduktion til debug

Når du trykker på start-pilen eller taster F5 som vist i afsnit 3.1, starter programmet som nævnt op. Men det gør mere end det. Det starter også det der hedder en Debugger op som gør at man kan følge med i hvad der sker i programmet mens det kører. Det er en meget stor fordel når man forsøger at finde ud af hvorfor en eller anden del af programmet ikke virker som det skal. Debuggeren kan også bruges som værktøj til at lære at forstå noget kode andre har skrevet. Når man følger med debuggeren gennem programmet får man en forståelse af hvordan de enkelte dele af programmet spiller sammen.

I dette afsnit kommer vi ind på nogle begreber der ikke er blevet præsenteret endnu men som bruges meget i forbindelse med debugging. Det er taget med her sådan at al information om debugging er samlet i ét afsnit. I første omgang betyder det ikke så meget om du forstår alt hvad der står, du kan altid komme tilbage

Debugging betyder ”at fjerne fejl (bugs)” – en bug betyder egentlig bille men bruges i computerverdenen om alle slags fejl. Læs evt. mere her: <https://en.wikipedia.org/wiki/Debugging>.Prøv at folde Debug-menuen ned øverst i Visual Studio:



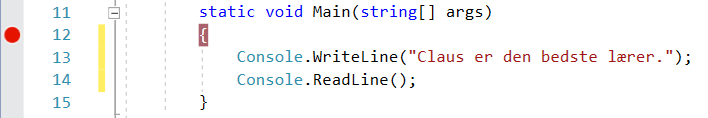
Som det fremgår har debuggeren mange funktioner. Vi skal i dette afsnit se på de mest vigtige funktioner.

* Den markerede menu-indgang Start Debugging - F5 er den funktion man aktiverer når man trykker på Start-knappen som nævnt i afsnit 3.1. Så den kender du allerede.
* Så er der også en indgang der hedder Start Without Debugging. Den er der ikke nogen grund til at dvæle meget ved. Så længe du er i gang med at lave programmet er det altid en fordel at køre med debugging, og hvis programmet skal bruges rigtigt, laver man en release-version som slet ikke kan debugges. Programmet vil køre hurtigere hvis du starter det uden debugging, men endnu hurtigere hvis du laver en release-version af programmet.
* Ellers er det hovedsagelig de tre punkter nævnt nedenfor som vi kommer til at beskæftige os med:
  + Step Into F11
  + Step Over F10
  + Toggle Breakpoint F9

## Breakpoints

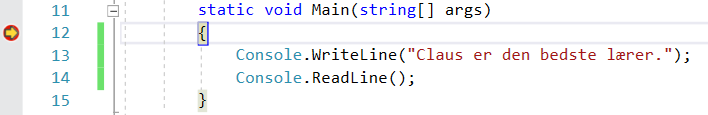
Vi starter med den nederste først. At sætte et breakpoint er ofte det første man gør når man gerne vil debugge sit program. Et breakpoint er en markering af et sted i koden hvor man gerne vil have programmet til at stoppe så man kan følge hvad programmet gør skridt for skridt derfra. Ofte sætter man breakpointet lige før det sted i programmet man tror der er problemer med.

Nu har vi ikke lavet et ret langt program her, så vi sætter bare et breakpoint lige i starten af metoden Main(). Man sætter et breakpoint ved at stille cursoren i den linje hvor man gerne vil have programmet til at stoppe op og så vælger man Toggle Breakpoint i Debug-menuen eller man trykker F9. Breakpointet markeres med en rød ”bullet” i margin.



Kør nu programmet ved at trykke på Start-knappen eller F5.

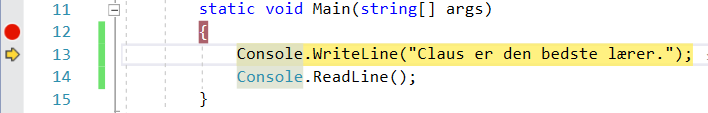
Programmet starter nu op, men ser så pludselig ud til at fortryde og fokus går tilbage til Visual Studio. Det der er sket er at programmet er startet op, men er stoppet da det kom til det sted hvor vi havde sat et breakpoint. Det markeres med en gul pil i margin. Den gule pil viser hvor i programmet vi er lige nu.



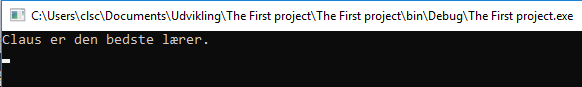
## At steppe sig gennem programmet

Som vist ovenfor giver debuggeren os to handlinger vi kan udføre, Step Into og Step Over. Vælg Step Over i menuen eller tryk F10.

Nu flytter pilen til næste linje. Pilen vil altid pege på *den næste instruktion der bliver udført*, så vi står altså i realiteten *lige før* den instruktion der er fremhævet med gult.



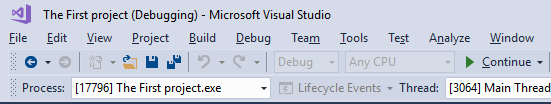
Først når vi trykker F10 igen bliver instruktionen udført og vi kan se udskriften i konsolvinduet:



Den anden handling vi kan udføre med debuggeren – Step Into – kan bruges til at følge programafviklingen ind i andre metoder når instruktionen i den linje debuggeren står i er et metodekald. Console.WriteLine(…) og Console.ReadLine() er faktisk begge to metodekald, men selv om du forsøger at steppe dig ind i de metoder, så stepper den over i stedet. Det skyldes at de to metoder ikke er oversat med debug-information. Hvis de ikke er oversat med debug-information kan de ikke debugges. Sådan er det med en alle de .Net-metoder vi kalder, dem får vi ikke lov til at komme med ind i.

Da vi ikke har arbejdet med begrebet metoder, giver det ikke meget mening at gå mere i dybden med Step Into på nuværende tidspunkt.

Ved at trykke F5 eller trykke på den grønne pil der nu har fået teksten Continue, beder man debuggeren om at lade programmet køre videre, sådan at vi nu ikke længere går gennem programmet skridt for skridt (single stepper).



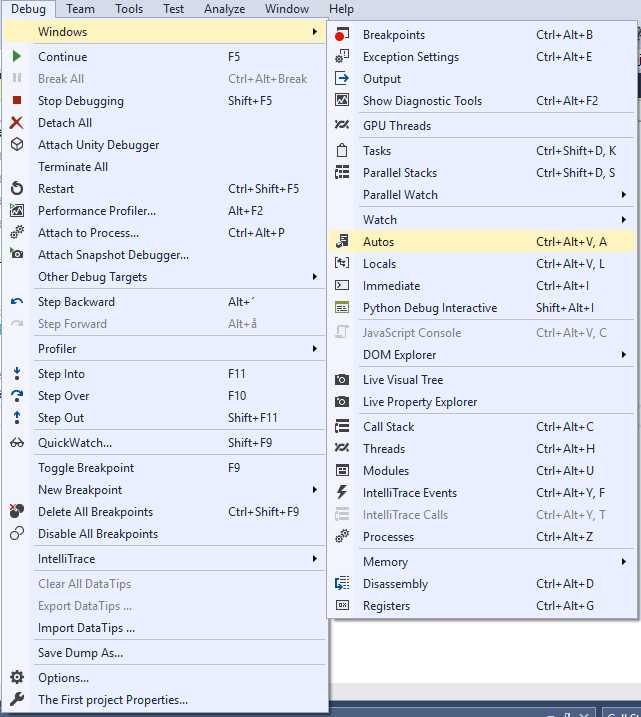
## Vigtige debug-vinduer

Indtil nu er det et meget simpelt program vi har lavet, derfor er det også begrænset hvor meget information vi kan få debuggeren til at vise os – programmet holder nærmest ikke noget runtime-information. Alligevel præsenteres i dette afsnit kort nogle af de muligheder vi har for at finde oplysninger om det kørende program.

### Vinduet Autos

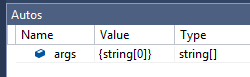
Dette vindue vises sandsynligvis automatisk den første gang du starter debuggeren og den stopper i et breakpoint. Dette vindue er nok det informationsvindue du kommer til at benytte allermest i forhold til debugging. I Autos-vinduet forsøger Visual Studio at vise dig den information den gætter på du er interesseret i. Det vil ofte være lokale variable eller variable der bruges ”i nærheden” af det sted i koden hvor vi står. Vi har ikke oprettet nogen variabel, så vinduet er rimelig tomt lige nu, men ikke helt.

Hvis ikke vinduet vises, så få det vist sådan:



Nedenstående er indholdet af Autos-vinduet når debuggeren afbryder udførelsen i det breakpoint vi satte ovenfor i afsnit 4.1 (du kom forbi det sted i koden i afsnit 4.2, så stop programmet med den røde stopknap og start det forfra).

Når programmet standser i breakpointet kan man se følgende i Autos-vinduet.



Vinduet fortæller at der findes en tilgængelig variabel der hedder args.

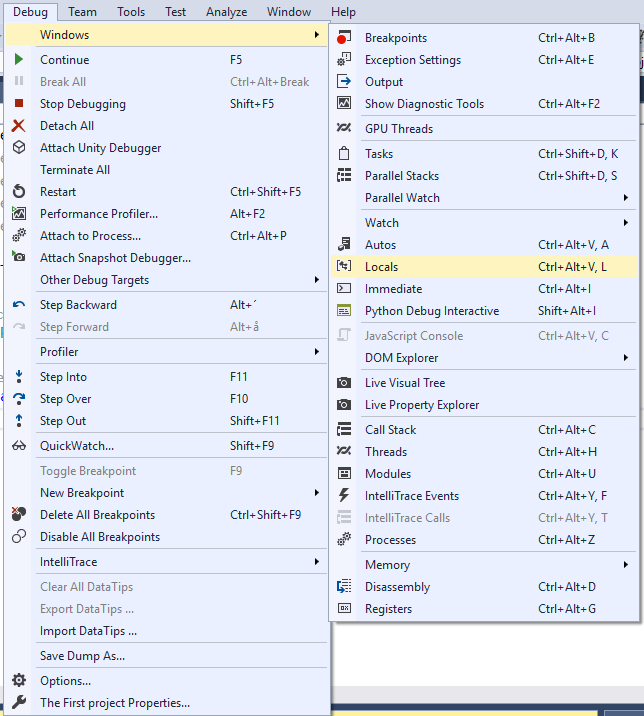
args er et array af string.

Værdien af variablen args får man ikke meget ud af at læse hvis man ikke på forhånd ved hvad det betyder, der står kort sagt bare at args ikke har nogen elementer, så der er ikke nogen værdi.

Her kom nogle ord som variabler, string og array som vi ikke har set endnu, det hele skal nok give mere mening når vi har noget konkret at debugge.

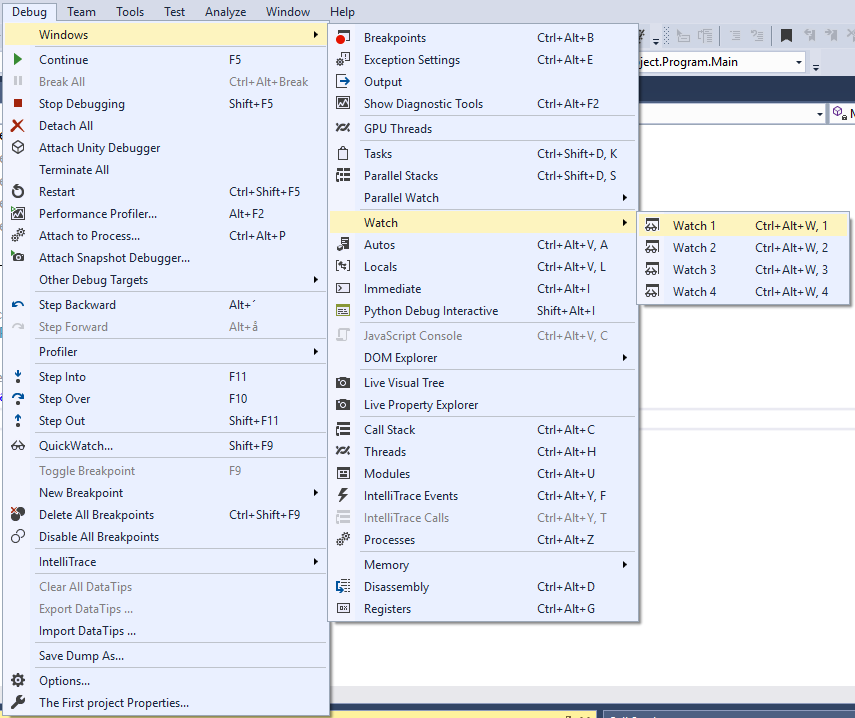
### Vinduet Locals

Vinduet Locals viser de lokale variabler der er erklæret i den aktuelle method. Find vinduet sådan:



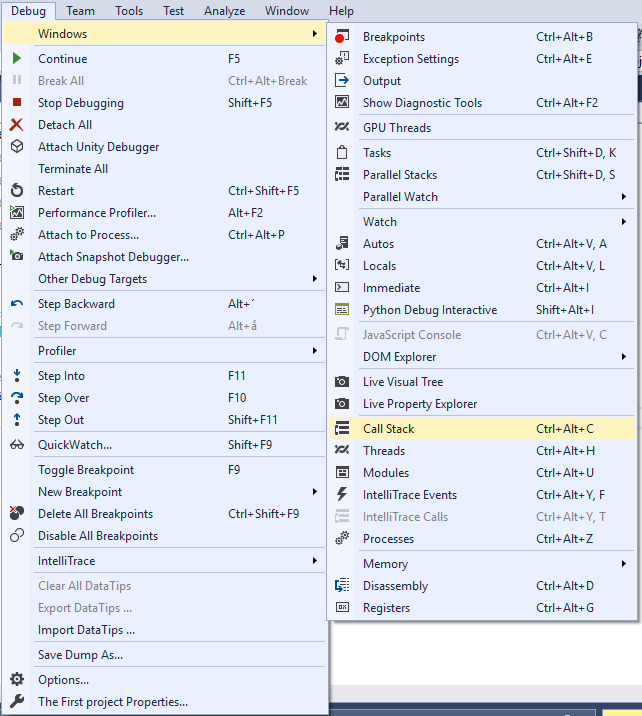
### Watch-vinduerne

I Watch-vinduerne kan man selv bestemme hvilke variabler man vil holde øje med.

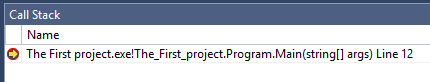


### Vinduet Call Stack

Når programmerne bliver store oplever man at methods kalder methods som kalder methods som kalder … Så kan det være rart at kunne navigere rundt i hele kaldsstakken så man også kan se hvordan vi er kommet til det sted i koden hvor vi står og ved at klikke i kaldsstakken kan vi se værdierne af de forskellige variabler i alle methods på kaldsstakken.



Som nævnt er der ikke meget at se i kaldsstakken med dette simple eksempel, men den er meget brugbar når programmerne begynder at blive mere komplekse.



1. Dialogen er baseret på en eksisterende installation, så den kan se lidt anderledes ud hos dig. [↑](#footnote-ref-1)