Velkommen til gruppetime i IN1000 ☺

21. august 2020

Jessie Yue Guan

Planen for i dag

- Bli kjent med hverandre
- Praktisk informasjon
- Faglig gjennomgang

- Selvstendig arbeid
- Jobbe med oblig eller oppgaver
- Rekke opp hånden og få hjelp

Litt om meg :)

- Jessie Yue Guan
- Fact or fiction?
- yuegu@uio.no

Litt om dere :)

- Bli-kjent-opprop
- Hva heter du?
- Hvor gammel er du?
- Hvilken retning går du?
- Hva gjorde du før studiestart?
- En sannhet og en løgn

Emnesiden og semestersiden

- Emnesiden
- https://www.uio.no/studier/emner/matnat/ifi/IN1000/

- Semestersiden
- https://www.uio.no/studier/emner/matnat/ifi/IN1000/h20/index.html

- Sjekk semestersiden ofte! Her kommer det viktig informasjon!

Forkurs og kart over bygget

- Forkurs
- https://uio.instructure.com/courses/22238

- Kart over bygget
- http://magnusli.no/ifirooms/

Studieadministrasjonen

- Studieveiledning
- Permisjon & deltidsstudier
- Utveksling
- Masteropptak
- Tilrettelagt eksamen
- Ikke levert oblig før fristen

- Sitter i 4. etg i bygg B
- Åpen man-fre ca. kl 9-10 og 12-15

Termvakt

- Adgang til bygget
- Bruk av printere
- Nedlastning av programmer
- Førstelinje IT-support

- Sitter i 1. etg ved kantina
- Åpen man-fre kl 8-18

UiO-mailen og videresending

- UiO-mailen
- https://mail.uio.no

- Automatisk videresending til en annen e-postkonto
- https://support.office.com/nb-no/article/bruke-regler-til-%C3%A5-videresende-meldinger-automatisk-45aa9664-4911-4f96-9663-ece42816d746

- Sjekk UiO-mailen ofte! Her kommer det også viktig informasjon!

MineStudier og kalenderoppsett

- MineStudier
- https://minestudier.uio.no/

- Automatiske oppdateringer til en annen kalender
- https://www.uio.no/tjenester/it/utdanning/minestudier/hjelp/abonnere.html

Devilry: Innlevering av obliger

- Devilry
- https://devilry.ifi.uio.no/

- Lever så mange ganger du vil før fristen, retteren ser kun på den siste innleveringen
- Det er bedre å få 3/5 poeng for en ikke-helt-ferdig innlevering enn 0/5 poeng for ingenting!

Trix-oppgaver: Trening i å programmere

- Trix-oppgaver
- https://trix.ifi.uio.no/course/8

- Ikke alle oppgaver har løsningsforslag
- Du kan filtrere oppgavene basert på uke og tema

MatterMost: Stille og svare på spørsmål

- MatterMost
- https://www.uio.no/studier/emner/matnat/ifi/IN1000/h20/praktisk-informasjon/mattermost.html

- IKKE PUBLISER OBLIG-KODE!!!

Gruppetimene

- Felles undervisning også individuelt arbeid
- Si fra hvis dere har noen ønsker
- Ikke obligatorisk oppmøte, men sterkt anbefalt!!!
- Kan ikke bytte gruppe pga. smittevern og smittesporing

Obligatoriske innleveringer

- Obligatoriske innleveringer
- https://www.uio.no/studier/emner/matnat/ifi/IN1000/v20/obliger/

- Oblig 1 til 6
  - Cirka 1 ukes frist
  - Kan ikke utsettes pga. sykdom
  - Kan ikke få nytt forsøk
  - Må få 19 av 29 poeng for å bestå

- Oblig 7 og 8
  - Cirka 2 ukers frist
  - Kan utsettes pga. sykdom
  - Kan muligens få nytt forsøk
  - Må få godkjent på begge

Eksamen

- Eksamen
- https://www.uio.no/studier/emner/matnat/ifi/IN1000/h20/eksamen/index.html

- Eksamenslokalet er ikke på UiO og ligger langt unna sentrum så møt opp i god tid!
- Banen/bussen blir som oftest full, du får kanskje ikke plass, det kan være forsinkelser, osv.

- Dere blir delt opp i tre grupper pga. avstandsregelen og kapasitetsbegrensninger
- Kan skje endringer basert på utviklingen av viruset og antall smittede!!!

Programmer, filer, mapper (generelt)

- Datamaskiner inneholder ulike programmer
- Programmene kan deles inn i ulike kategorier

- Tekstbehandlingsprogrammer/ Tekstredigeringsprogrammer/ Skriveprogrammer
  - For eksempel: NotePad, Word, OneNote, Atom (ish), osv.
- Kommandolinje
  - For eksempel: PowerShell (Windows), Ledetekst (Windows), Terminal (MAC), Shell (Linux)
- Filbehandlingsprogram/ Filorganiseringsprogram
  - For eksempel: File Explorer/Filutforsker (Windows), Finder (MAC), osv.
- Nettlesere
  - For eksempel: Internet Explorer, FireFox, Google Chrome, Safari, Microsoft Edge, Opera, osv.

Programmer, filer, mapper (generelt)

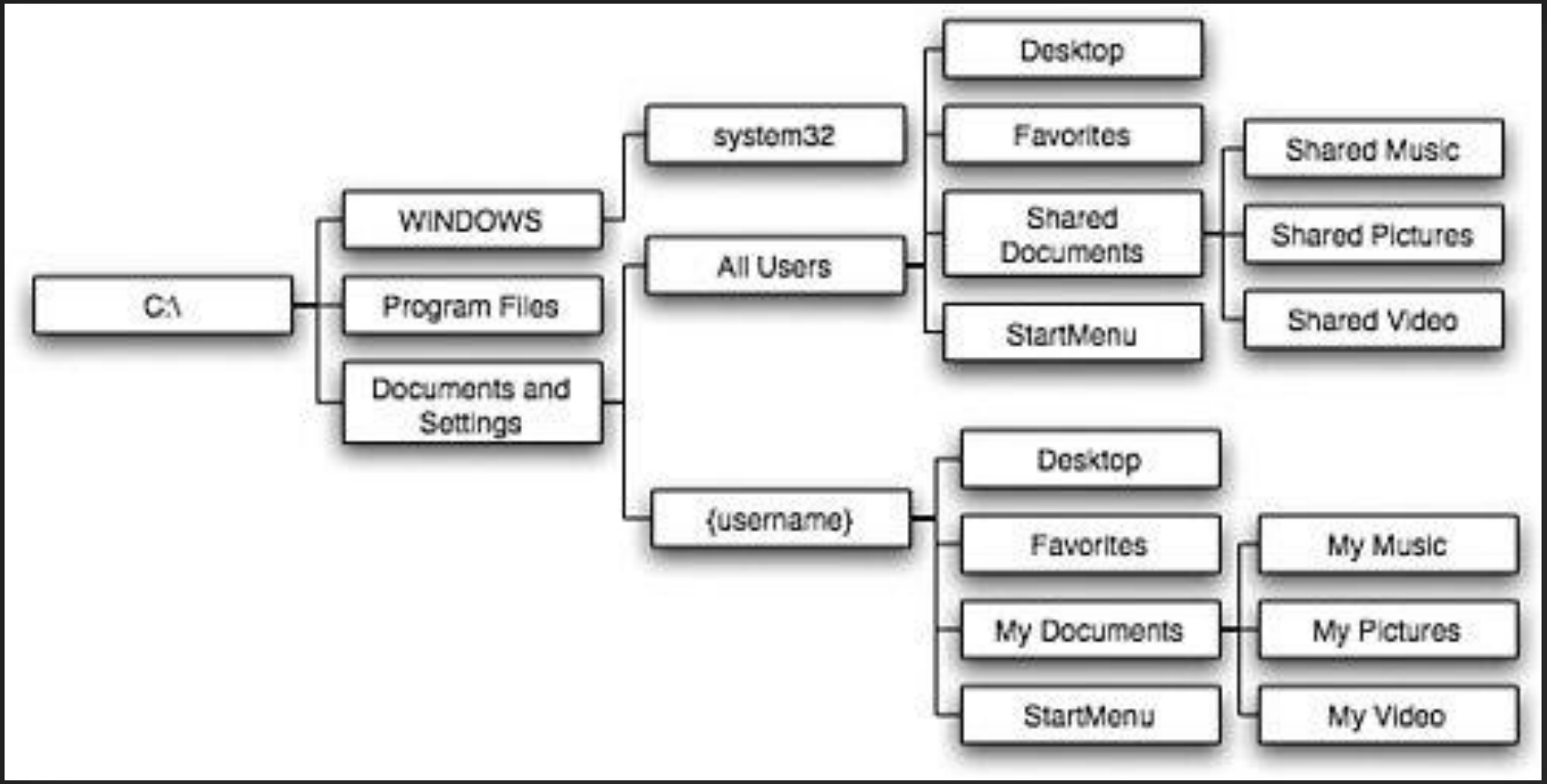
- Programmene består av filer og mapper

- Filer kan sees på som innhold
  - Kan ha forskjellige navn og filtyper
  - Refereres til ved hjelp av filnavn.filtype

- Mapper kan sees på som esker
  - Kan ha mange mapper/esker inni hverandre
  - Har en hierarkisk struktur, se bilde på neste slide

- Noen eksempler på filyper er…
  - Bilder: jpg, png, gif, osv.
  - Lyd: mp3, wav, wma, osv.
  - Video: mp4, svf, mpv, osv.
  - Tekst: txt, docx, py, osv.

Programmer, filer, mapper (generelt)



Programmer, filer, mapper (dette kurset)

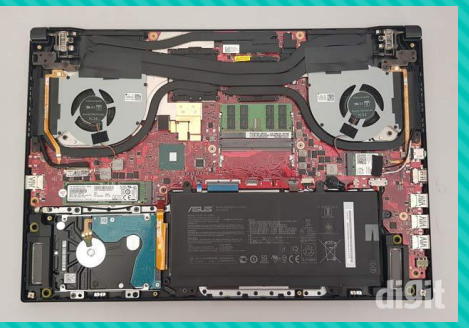
- Python 3.7 består av filer og mapper
- Atom består av filer og mapper
- Dere skal ved hjelp av tekstbehandlingsprogrammet **Atom** og programmeringsspråket **Python** lage deres egne programmer som består av py-filer. Disse programmene skal oppnå et eller flere mål og løse et eller flere problemer og kan kjøres/testes/brukes ved hjelp av en **kommandolinje**.

Terminalen/Kommandolinja

- Det er her du kjører/tester/bruker programmene dine, men du får ikke skrevet dem

- Søk etter programmet…
  - Ledetekst/Command Line på Windows
  - Terminal på Mac
  - Terminal på Linux

Atom



- Det er her du skriver programmene dine, men du får ikke kjørt/testet/brukt dem
- Husk hvilken mappe du befinner deg i og husk å lagre filen hvis du har endret den!!!
- Tip 1: Sørg for at «Show Welcome Guide when opening Atom» IKKE ER avhuket
- Tip 2: Velg «Yes, send my usage data» eller «No, do not send my usage data»
- Tip 3: File -> Settings -> Editor -> Sørg for at «Show Indent Guide» og «Soft Wrap» ER avhuket

Terminalen/Kommandolinja

- Trykk tasten **tab** for å fullføre det du skriver automatisk
- Trykk tasten **pil opp** for å bla igjennom det du har skrevet inn tidligere
- Skriv **python** for å sjekke om Python fungerer
- Skriv **dir** (Windows) eller **ls** (Mac/Linux) og trykk enter for å se filer og mapper
- Skriv **cd *mappenavn*** og trykk enter for å gå til mappen som heter *mappenavn*
- Skriv **cd ..** og trykk enter for å gå opp en mappe
- Skriv **python *filnavn*.py** og trykk enter for å kjøre programmet som heter *filnavn*
- Hold inne **CTRL** mens du trykker **C**: Avbryt programmet

Programmeringstips

- Vær veldig nøye når du skriver kode, datamaskiner kan dessverre ikke lese tankene våres
- Prøv å forstå hver eneste linje med kode, hvorfor ting fungerer, og hvorfor ting ikke fungerer
- Ikke vær redd for å feile, eksperimenter systematisk, prøv alle mulige kombinasjoner
- Fokuser på unntakene, lurespørsmålene, de sjeldne og rare men viktige tilfellene
- Bruk alle ressurser! Pensumboka, forelesninger, gruppetimer, og mye mer

Programmeringstips

- Hvis du ikke skjønner en forklaring fra boka/forelesningen, prøv å søke litt rundt på nettet
- Hvis du har stått fast veldig lenge, ta en pause, så ser du ofte hva som er løsningen
- Finn noen å studere med, det øker motivasjon og du får med deg informasjon
- Jeg forstår konseptene men aner ikke hva jeg skal skrive? Fokuser på praksis/syntaks
- Jeg vet hva jeg skal skrive men aner ikke hvorfor det ikke funker? Fokuser på teori/semantikk

Spørsmål?

- Ikke vær redd for å spørre, det finnes ingen dumme spørsmål! ☺

Variabler

- Noe som kan forandre seg
- Motsatte av en konstant
- En boks med et navn og et innhold
- 4 forskjellige typer i Python

Integers

- Forkortes ofte som "int"
- Heltall
- Kan være positiv eller negativ

- For eksempel:
  - **mitt_tall = -45**
  - **ditt_tall = 8**
  - **i = -9836**
  - **lagerbeholdning = 538**

Floating point

- Forkortes ofte som "float"
- Desimaltall/flyttall
- Kan være positiv eller negativ
- Bruk . og ikke ,

- For eksempel:
  - **mitt_tall = -3.14**
  - **ditt_tall = 7.2349**
  - **f = -239247.2**
  - **rekordtid = 3.9475**

String

- Forkortes ofte som "str"
- Strenger/tekst
- Kan være bokstaver, tall, eller spesialtegn
- Husk å bruke ' ' eller " " eller """ """

- For eksempel:
  - **bokstav = "k"**
  - **tegn = "!"**
  - **setning = "Hei " + brukernavn**
  - **sporsmaal = "Er du 18 år gammel?"**
  - **tekst = "sWo32)"#"**

Boolean

- Forkortes ofte som "bool"
- ENTEN **True** ELLER **False**
- Tenk på det som en lysbryter
- Vanlig å glemme! Viktig å huske!

Spørsmål?

- Ikke vær redd for å spørre, det finnes ingen dumme spørsmål! ☺

Tilordning av verdier

- **x = 4**
  - Vi oppretter en variabel kalt **x** med verdien **4**
- **y = 7**
  - Vi oppretter en variabel kalt **y** med verdien **7**

- **x = y**
  - Vi setter **x** til å være lik **y**, slik at den bytter verdi fra **4** til **7**, **x** er nå lik **7** (!!!)
- **y = x**
  - Vi setter **y** til å være lik **x**, slik at den bytter verdi fra **7** til **7**, **y** er nå lik **7**

- **print(x, y)**
  - Vi printer x og y og får: *7 7*

Input

- **brukernavn = input()**

- **print("Hva er brukernavnet ditt?")**
- **brukernavn = input()**

- **brukernavn = input("Hva er brukernavnet ditt?")**

- Husk å bruke ' ' eller " " eller """ """

- Hva skjer hvis brukeren skriver inn et heltall eller et desimaltall?

Output/Printing

- print("Hei og velkommen!")
- print("Heltallet mitt er ", 2)
- print("Desimaltallet mitt er", 3.14)
- print("Hei", navn, "og velkommen!")
- print("Heltallet ditt er", ditt_heltall")
- print("Desimaltallet ditt er", ditt_desimaltall")

- Husk å bruke ' ' eller " " eller """ """

Spørsmål?

- Ikke vær redd for å spørre, det finnes ingen dumme spørsmål! ☺

Konkatenering av strenger

- Man kan bruke + eller komma for å sette sammen strenger og variabler

streng	+	streng
streng	+	variabel
variabel	+	streng
variabel	+	variabel

streng	,	streng
streng	,	variabel
variabel	,	streng
variabel	,	variabel

- Komma vil føre til at det blir et mellomrom mellom strengene/variablene
- Pluss vil føre til at det ikke blir et mellomrom mellom strengene/variablene
- Hvis du bruker sistnevnte burde du legge til et mellomrom selv for lesbarhet

Aritmetiske operasjoner

- Brukes på integers eller floats
- Resulterer i en integer eller en float

- Består av følgende:
  - Addisjon (**+**)
  - Subtraksjon (**-**)
  - Multiplikasjon (*****)
  - Divisjon (**/**)
  - Potens (******)
  - Heltallsdivisjon (**//**)

Aritmetiske operasjoner - rekkefølge

- Parentesene løses først
  - For eksempel: **4 * (7 + 3)** blir **40**
- Multiplikasjon og divisjon løses etter det
  - For eksempel: **6 + 2 * 5** blir **16**
- Addisjon og subtraksjon løses etter det
  - For eksempel: **20 – 10 / 5** blir **18**

Spørsmål?

- Ikke vær redd for å spørre, det finnes ingen dumme spørsmål! ☺

If, elif, og else

- Hvis, eller hvis, og ellers
- Utgjør til sammen det vi kaller if-setninger

- Brukes når man skal vil kjøre forskjellige kodelinjer/kodeblokker basert på noe
- F.eks. hvis brukeren er en student printes det ut oppgaver, eller hvis brukeren er en lærer printes det ut løsningsforslag

- Kan sammenlignes med et veikryss eller togskinner som går i ulike retninger
- Retningen du velger bestemmer hva du kan gjøre, for eksempel…
  - Hvis du velger å gå i retning av Universitetet i Oslo kan du studere men ikke dra på konsert
  - Hvis du velger å gå i retning av Oslo Spektrum kan du dra på konsert men ikke studere

If, elif, og else

```
antall_katter = 3

if antall_katter < 1:
    print("Liker du ikke katter?")
elif antall_katter < 3:
    print("Du er glad i katter :)")
elif antall_katter < 5:
    print("Du er veldig glad i katter!")
else:
    print("Du har kanskje litt vel mange katter?")
```

- Hva blir printet ut?
- **Du er veldig glad i katter!**
- Hvorfor?
- Fordi 3 < 3 er ikke sant
- Altså 3 er ikke mindre enn 3
- Men 3 <= 3 er sant
- Altså 3 er mindre enn **eller lik** 3

If, elif, og else

- **if alder < 18:**
  - **print(«Du er myndig!»)**
- **print(«Du er ikke myndig!»)**

- **if alder < 18:**
  - **print(«Du er myndig!»)**
- **else:**
  - **print(«Du er ikke myndig!»)**

- Hva er forskjellen mellom disse to programmene?
- I det øverste programmet får brukeren både beskjed om at de er myndig og at de ikke er det.
- Det gir kanskje mening for deg som skriver programmet men ikke for brukeren som ikke nødvendigvis kan programmere.
- Når en if/elif-setning evalueres til sann og innholdet blir utført til vil Python automatisk hoppe over resten av elif/else-setningene

Spørsmål?

- Ikke vær redd for å spørre, det finnes ingen dumme spørsmål! ☺

Relasjonelle operasjoner

- Brukes på integers (heltall) og floats (desimaltall)
- Resulterer i en boolean

- Består av følgende:
  - Er lik (**==**)
  - Er ikke lik (**!=**)
  - Mindre enn (**<**)
  - Mindre enn eller lik (**<=**)
  - Større enn (**>**)
  - Større enn eller lik (**>=**)

Relasjonelle operasjoner - rekkefølge

- Rekkefølgen går fra venstre til høyre
- **x < y < z** skal tolkes som **x < y and y < z**

x < y	y < z	(x < y) and (y < z)
True	True	True
False	True	False
True	False	False
False	False	False

Spørsmål?

- Ikke vær redd for å spørre, det finnes ingen dumme spørsmål! ☺

Logiske operasjoner

- Brukes på booleans
- Resulterer i en boolean

- Består av følgende:
  - negasjon (**not**) som betyr "ikke"
  - konjunksjon (**and**) som betyr "både og"
  - disjunksjon (**or**) som betyr "enten eller"

Logiske operasjoner - not

- La oss si at vi har en boolean variabel som heter **min_bool**
- Hvis vi setter **True** som verdien til **min_bool** vil **not min_bool** bli **False**
- Hvis vi setter **False** som verdien til **min_bool** vil **not min_bool** bli **True**

min_bool	not min_bool
True	False
False	True

Logiske operasjoner - and

- La oss si at vi har en boolean variabel kalt **x** og en boolean variabel kalt **y**
- Hvis *både* **x** *og* **y** har verdien **True**, så vil verdien av **x and y** også bli **True**
- I alle andre tilfeller vil **x and y** bli **False**

x	y	x and y
True	True	True
False	True	False
True	False	False
False	False	False

Logiske operasjoner - or

- La oss si at vi har en boolean variabel kalt **x** og en boolean variabel kalt ***y***
- Hvis *både* **x** *og* ***y*** har verdien **False**, så vil verdien av **x or y** også bli **False**
- I alle andre tilfeller vil **x or y** bli **True**

x	y	x or y
True	True	True
False	True	True
True	False	True
False	False	False

Logiske operasjoner - rekkefølge

- Du skal alltid løse **not** først
  - For eksempel: **not True or True** blir altså **False**
- Deretter skal du løse **and**
  - For eksempel: **True and False or True** blir altså **False**
- Til slutt skal du løse **or**
  - For eksempel: **not False or False** blir altså **True**

Spørsmål?

- Ikke vær redd for å spørre, det finnes ingen dumme spørsmål! ☺

Hva om jeg vil ha " " i strengen?

1. **print(' "En setning som trenger anførselstegn" ')**
2. **print(""" "En setning som trenger anførselstegn" """)**
3. **print(" \"En setning som trenger anførselstegn\" ")**

- Alle alternativer fører til samme output:
- *"En setning som trenger anførselstegn"*

Kommentarer

- **#En kort kommentar på en linje**
- **""" En lang kommentar som inneholder massevis av informasjon og som går over flere linjer og tar litt mer plass """**
- Tenk litt på hvordan man kan vite hva som er en kommentar og hva som er en streng
- Gjør om kodelinjer eller kodeblokker til kommentarer for å teste dem ut uten å slette dem