DIGI - Hovedpoeng

Modul 1: Interaksjon, funksjoner og enkle datatyper

- Datamaskiner bare utfører spesifikke instruksjoner de er programmert med.
- Hver variabel har en spesifikk datatype.
- Vi kan konvertere datatypen til en variabel med for eksempel funksjonene `int()`, `float()`, `str()` når det er mulig.
- Feilmeldinger er ikke farlige men hjelpsomme! Les dem nøye!
- Det kan være lurt å skrive på norsk det vi vil gjøre (f.eks. som kommentar) før vi skriver kode.
- Variabler og funksjoner er veldig hjelpsomme når vi trenger å bruke en verdi eller en kodeblokk flere ganger.
- Rekkefølgen for utførelse av funksjoner følger rekkefølgen de blir kalt, ikke rekkefølgen de blir definert i.
- En funksjon alltid returnerer noe. Men hvis det er ingen "return"-setning, returnerer det `None`.
- En variabel som var opprettet i en funksjon eksisterer ikke utenfor funksjonen. Hvis man vil få tak i en variabel som ble definert i en funksjon utenfor funksjonen, **må** man returnere variabelen i funksjonen.

Modul 2: Betingelser, sekvenser og løkker

- Semantisk og logiske feil er farlige! Vi "liker" kjøretidsfeil (syntaktisk feil) fordi programmet stopper umiddelbart og vi får en hjelpsom feilmelding. Med en logisk feil kjører dessverre programmet uten å stoppe, og vi legger kanskje ikke merke til at resultatet er feil!
- For å unngå store mengder semantiske/logiske feil, samtidig som vi forbedrer lesbarheten av koden vår, kan vi følge denne prosedyren:
    1. **Før vi begynner å skrive kode**, starter vi med å skrive beskrivelsen av hovedoppgavene (oppgaver på høyt nivå) vi ønsker å oppnå i programmet vårt med fulle norske setninger.
    2. Deretter bryter vi ned hovedoppgavene iterativt i flere mindre oppgaver (oppgaver på lavt nivå) til alle oppgavene på lavest nivå tilsvarer (nesten) én linje med kode.
    3. Bruk oppgavene på høyere nivå og oppgavene på lavere nivå som kommentarer i funksjonen.
    4. **Først nå kan du begynne å skrive kode**
- For å raskt oppdage feil i koden
    1. Tenk på flyten av en gitt inndata i koden din. Prøv med forskjellige inndata i tankene dine.
    2. Test koden din med noen gitte inndata (prøv å tenke på typiske inndata man ville bruke med denne koden, men også kanttilfeller som negative tall, 0, tomme lister eller strenger, osv.). Sjekk at resultatet er som forventet.
- Det er viktig å lese dokumentasjonen nøye for funksjonene/metodene vi bruker.
- Ikke vær redd for å søke informasjon på internett. Sørg bare for at du bruker riktige ord for å søke effektivt.
- Man trenger ikke å lære alt om alle de metode/bibliotekene som eksisterer, men i programmering er det viktig å lære å finne informasjon raskt om en gitt funksjon/bibliotek.
- Python har en rekke innebygde funksjoner som gir grunnleggende funksjonalitet og operasjoner
- Det er foranderlige sekvenstyper i Python (`list`) og uforanderlige sekvenstyper (`str`). I tillegg deler alle sekvenstyper felles metoder.
- En spesifikk type kan ha ekstra metoder. For eksempel lister og strenger.

- Husk hvilken type variablene du arbeider med har. Resultatet av en gitt operasjon kan avhenge av variabeltypen, og de tilgjengelige metodene avhenger også av variabeltypen.
- `for`-løkken brukes når vi vet antall ganger vi ønsker å gjenta koden på forhånd eller nå vi vet at vi vil iterere over en hel sekvens.
- `while`-løkken brukes når vi ikke kjenner antallet gjentakelser på forhånd, men vil fortsette å gjenta så lenge en betingelse er sann.

Modul 3: Ordbok, sett og tekstfil

- Alltid test funksjonene dine! Test vanlige tilfellene, kanttilfellene. Det er vanskelig å skrive tester for våre egen funksjoner fordi vi pleier å ikke se potensielle feil i koden vår, men det er hovedmåten å finne feil på.
- En gitt samling av data kan lagres på ulike måter i Python ved hjelp av ulike datastrukturer. Ta deg tid til å velge den beste datatypen for dine spesifikke data og oppgaver.
- Det kan være lurt å konvertere en variabel til en annen type mellom to oppgaver.
- Man bruker en liste (dvs en sekvens som er foranderlig):
    - Når man trenger en ordnet samling av elementer.
    - Når man potensielt ønsker å ha duplikater.
    - Når man vil kunne legge til, fjerne eller endre elementer etter opprettelsen.
- Man bruker en ordbok:
    - Når man har behov for å lagre data i form av nøkkel-verdi-par.
    - For å få en rask tilgang til data basert på unike nøkler.
- Man bruker et sett:
    - Når man trenger en samling av unike elementer uten en bestemt rekkefølge.
    - For å eliminere duplikater fra en annen samling.
    - For å få tilgang til matematiske settoperasjoner, som union, snitt og differanse.