If-setning

1. Les inn et tall fra tastaturet. (Husk å bruke casting til riktig datatype.)
   - Hvis tallet er større enn 10, skriv ut "Tallet er større enn 10".
   - Hvis tallet er mindre enn 10, skriv ut "Tallet er mindre enn 10".
   - Hvis tallet er lik 10, skriv ut "Tallet er lik 10".

2. Les inn to tall fra tastaturet og lagre dem i variabler av typen `int`, kall dem `x` og `y`.
   - a) Hvis `x` og `y` er like, skriv ut "x og y er like".
   - b) Hvis `x` er større enn `y`, skriv ut "x er større enn y", ellers skriv ut "x er mindre enn y".

For-løkke

1. Lag en for-løkke som skriver ut tallene fra 0 til 9.
2. Lag en for-løkke som skriver ut tallene fra 5 til 15.
3. Lag en for-løkke som skriver ut alle partallene opp til 20.
4. Lag en for-løkke som skriver ut alle oddetallene opp til 20.
5. Lag en for-løkke som skriver ut tallene fra 30 og ned til 10.
6. Lag en for-løkke som skriver ut alle oddetallene fra 30 til 10.
7. Lag en for-løkke som summerer tallene fra 1 til 10. Skriv ut summen etter hver iterasjon.
8. Lag en for-løkke som summerer alle partallene opp til 20.
9. Lag en for-løkke som summerer alle oddetallene opp til 20.
10. Lag et program som leser inn et tall og skriver ut gangetabellen for dette tallet ved hjelp av en for-løkke.

While-løkke

1. Lag en while-løkke som skriver ut tallene fra 1 til 10.
2. Lag en while-løkke som skriver ut tallene fra 5 til 15.
3. Lag en while-løkke som skriver ut tallene fra 30 til 10.
4. Lag en while-løkke som skriver ut partallene opp til 20.
5. Lag en while-løkke som skriver ut alle kvadrattallene opp til 100.
6. Lag en while-løkke som skriver ut alle kvadrattallene som er mindre enn en verdi som leses inn fra tastaturet.

Doble tall

1. Lag et program som for hver iterasjon dobler tallet som skrives ut på skjermen. Start med tallet 1 og gjør dette i 10 iterasjoner.
   - Resultatet skal være: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512

Fibonaccitall (Vanskelig)

1. Lag et program som skriver ut Fibonacci-tallfølgen på skjermen.
   - Start med 1, 1 og fortsett til en ønsket grense. Fibonacci-tallene er 1, 1, 2, 3, 5, 8, 13, 21, ...

   [Mer informasjon om Fibonacci-tall](https://no.wikipedia.org/wiki/Fibonaccitall)

Listeoperasjoner

1. Legge til og fjerne elementer:
   - `append(element)`: Legger til et element på slutten av listen.
   - `insert(index, element)`: Legger til et element på en bestemt indeks i listen.
   - `extend(iterable)`: Utvider listen ved å legge til elementene i et annet iterable (f.eks. en

annen liste).
  - `remove(element)`: Fjerner den første forekomsten av et element fra listen.
  - `pop([index])`: Fjerner og returnerer et element basert på indeksen. Uten indeks fjerner den siste elementet.
  - `clear()`: Fjerner alle elementene fra listen.
  - `sort()`: Sorterer elementene i listen i stigende rekkefølge (endrer den opprinnelige listen).
  - `sorted()`: Returnerer en ny sortert liste uten å endre den opprinnelige listen.
  - `reverse()`: Reverserer rekkefølgen på elementene i listen (endrer den opprinnelige listen).

2. Søk og indeksering:
  - `index(element)`: Returnerer indeksen til den første forekomsten av et element i listen.
  - `count(element)`: Returnerer antall ganger et element forekommer i listen.

3. Kopiering og kloning:
  - `copy()`: Returnerer en kopi av listen.

4. Listefunksjoner:
  - `len()`: Returnerer antall elementer i listen.
  - `max()`: Returnerer det største elementet i listen.
  - `min()`: Returnerer det minste elementet i listen.
  - `sum()`: Returnerer summen av elementene i listen.

Sorter og Reverser

1. Sorter:
  - Sorter listen `[3, 1, 4, 1, 5, 9, 2, 6, 5]` i stigende rekkefølge.
  - Sorter listen `['apple', 'Banana', 'Cherry']` alfabetisk, uavhengig av store/små bokstaver.
  - Sorter tuplet `(6, 2, 9, 1, 5, 3)` i synkende rekkefølge.
  - Sorter listen `['apple', 'banana', 'cherry']` basert på lengden av hvert ord.
  - Sorter en liste med dictionaries basert på en bestemt nøkkel.

2. Reverser:
  - Reverser listen `[1, 2, 3, 4, 5]`.
  - Konverter strengen `'hello'` til en liste og reverser den ved hjelp av `reversed`.
  - Reverser tuplet `(1, 2, 3, 4, 5)`.
  - Hvordan ville du brukt `reversed` for å reversere nøklene i en dictionary?
  - Reverser rekkefølgen av ordene i strengen `'Python er gøy'` ved hjelp av `reversed`.

Filter

1. Filtrer ut alle oddetallene fra listen `[1, 2, 3, 4, 5, 6]`.
2. Bruk `filter` for å finne alle ord i listen `['apple', 'banana', 'cherry', 'date']` som har mer enn 5 bokstaver.
3. Filtrer ut alle negative tall fra tuplet `(-5, -3, 2, 4, -1, 6)`.
4. Bruk en kombinasjon av `filter` og `map` for å finne kvadratet av tallene i listen `[1, 2, 3, 4, 5]` som er større enn 10.
5. Bruk `filter` for å hente ut alle partall fra en dictionary med tall som nøkler.

Zip

1. Bruk `zip` for å kombinere listen `['a', 'b', 'c']` med listen `[1, 2, 3]` til en liste av tupler.
2. Kombiner to lister med tall, den ene er `[1, 2, 3]` og den andre er `[4, 5, 6]`, til en ny liste

med summer av parrede elementer ved hjelp av `zip` og `map`.
3. Lag en dictionary der nøklene er fra listen `['Anna', 'Bob', 'Charlie']` og verdiene er fra listen `[25, 30, 35]` ved hjelp av `zip`.
4. Kombiner tre lister ved hjelp av `zip`: `['a', 'b', 'c']`, `[1, 2, 3]`, og `['apple', 'banana', 'cherry']`.
5. Hvordan ville du brukt `zip` for å splitte en liste av tupler (f.eks. `[('a', 1), ('b', 2), ('c', 3)]`) i to separate lister?

Map

1. Bruk `map` for å øke alle verdier i listen `[1, 2, 3, 4]` med 1.
2. Konverter alle strengene i listen `['1', '2', '3']` til heltall.
3. Skriv en funksjon som returnerer første bokstav i hvert ord fra listen `['apple', 'banana', 'cherry']` ved hjelp av `map`.
4. Bruk `map` for å lage en ny liste der hvert element er lengden av ordene fra listen `['apple', 'banana', 'cherry']`.
5. Bruk `map` for å lage en ny dictionary fra to lister, der den første listen `['a', 'b', 'c']` er nøkler og den andre `[1, 2, 3]` er verdier.