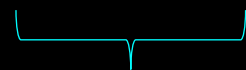


010101100110010101101100011010110110111101101101011011010110010101101110

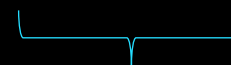
01010110011001010110110001101011011011101101101011011010110010101101110

01010110011001010110110001101011011011101101101011011010110010101101110



V

01010110011001010110110001101011011011101101101011011010110010101101110



v e l k o m m e n

010101100110010101101100011010110110111101101101011011010110010101101110

v e l k o m m e n

Binary ↕	Oct ↕	Dec ↕	Hex	Glyph		
				1963 ↕	1965 ↕	1967 ↕
010 0000	040	32	20	space (no visible glyph)		
010 0001	041	33	21	!		
010 0010	042	34	22	"		
010 0011	043	35	23	#		
010 0100	044	36	24	\$		
010 0101	045	37	25	%		
010 0110	046	38	26	&		
010 0111	047	39	27	'		
010 1000	050	40	28	(		
010 1001	051	41	29	)		
010 1010	052	42	2A	*		
010 1011	053	43	2B	+		
010 1100	054	44	2C	,		
010 1101	055	45	2D	-		
010 1110	056	46	2E	.		
010 1111	057	47	2F	/		

011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	:
011 1100	074	60	3C	^
011 1101	075	61	3D	
011 1110	076	62	3E	∇
011 1111	077	63	3F	?
100 0000	100	64	40	@ , @

100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R

101 0101
101 0110
101 0111

101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\ ~ \
101 1101	135	93	5D	]
101 1110	136	94	5E	↑ ^
101 1111	137	95	5F	† _
110 0000	140	96	60	@ ' .
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e

110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o
111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{

010101100110010101101100011010110110111101101101011011010110010101101110

v e l k o m m e n

Binary ↕	Oct ↕	Dec ↕	Hex	Glyph		
				1963 ↕	1965 ↕	1967 ↕
010 0000	040	32	20	space (no visible glyph)		
010 0001	041	33	21	!		
010 0010	042	34	22	"		
010 0011	043	35	23	#		
010 0100	044	36	24	\$		
010 0101	045	37	25	%		
010 0110	046	38	26	&		
010 0111	047	39	27	'		
010 1000	050	40	28	(		
010 1001	051	41	29	)		
010 1010	052	42	2A	*		
010 1011	053	43	2B	+		
010 1100	054	44	2C	,		
010 1101	055	45	2D	-		
010 1110	056	46	2E	.		
010 1111	057	47	2F	/		

011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;
011 1100	074	60	3C	< ^
011 1101	075	61	3D	=
011 1110	076	62	3E	> v
011 1111	077	63	3F	? ~
100 0000	100	64	40	@ ' @

100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R

101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\ ~ \
101 1101	135	93	5D	]
101 1110	136	94	5E	↑ ^
101 1111	137	95	5F	↓ _
110 0000	140	96	60	@ ' .
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e

110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o
111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{

UTF-8

UTF-8 is a variable-length character encoding for Unicode.

0101011001100101011011000110101101101111011011010110110010101101110

v e l k o m m e n

011101000110100101101100

t i l

110 1000	150	104	68		h
110 1001	151	105	69		i
110 1010	152	106	6A		j
110 1011	153	107	6B		k
110 1100	154	108	6C		l
110 1101	155	109	6D		m
110 1110	156	110	6E		n
110 1111	157	111	6F		o
111 0000	160	112	70		p
111 0001	161	113	71		q
111 0010	162	114	72		r
111 0011	163	115	73		s
111 0100	164	116	74		t
111 0101	165	117	75		u
111 0110	166	118	76		v

010101100110010101101100011010110110111101101101011011010110010101101110

v e l k o m m e n

011101000110100101101100

t i l

010010010100111001000110001100010011000000110000

I N F 1 0 0

100 0101	105	69	45	E	011 0000	060	48	30	0
100 0110	106	70	46	F	011 0001	061	49	31	1
100 0111	107	71	47	G	011 0010	062	50	32	2
100 1000	110	72	48	H	011 0011	063	51	33	3
100 1001	111	73	49	I	011 0100	064	52	34	4
100 1010	112	74	4A	J	011 0101	065	53	35	5
100 1011	113	75	4B	K	011 0110	066	54	36	6
100 1100	114	76	4C	L	011 0111	067	55	37	7
100 1101	115	77	4D	M	011 1000	070	56	38	8
100 1110	116	78	4E	N	011 1001	071	57	39	9
					011 1010	072	58	3A	:
					011 1011	073	59	3B	;
					011 1100	074	60	3C	<
					011 1101	075	61	3D	-

INNØRING I PROGRAMMERING

INF100

HØST 2025

Torstein Strømme

Hva er data?

Praktisk informasjon om emnet

Hva er programmering?

HVA ER DATA?



W-Observatory_phys_ocean_2016 (1).tab — Edited

License: CC-BY-NC-SA (Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC-BY-4.0) (URI: <https://creativecommons.org/licenses/by/4.0/>))

Status: Curation Level: 1 (URI: https://wiki.pangaea.de/wiki/Curation_levels) * Processing Level: PANGAEA data processing level_3

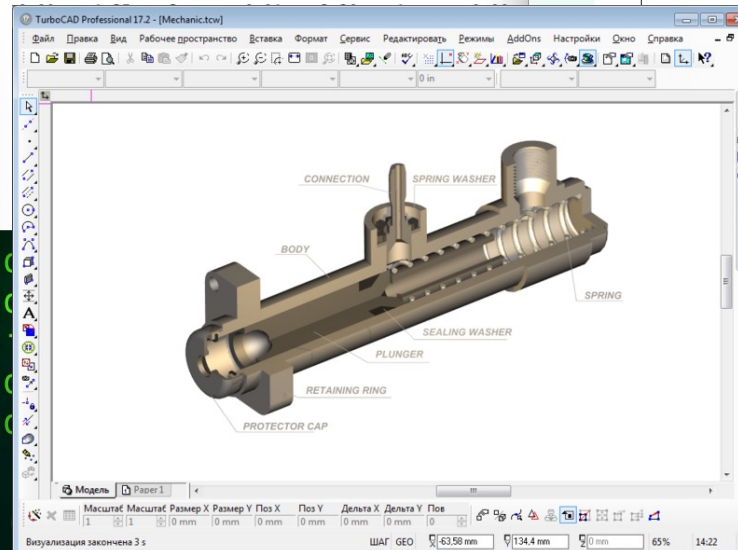
Size: 102861 data points

Date/Time Chl a [µg/l] (Value to be treated with...) NOBS [#] (Chlorophyll a; number of observations) Chl a conf [±] (90% confidence value (+/-) in...) 02 sat [%] (Temperature and salinity compensation) NOBS [#] (Oxygen saturation; number of observations) 02 sat conf [±] (90% confidence value (+/-) in...) Sal (PSU) NOBS [#] (Salinity; number of observations) Sal conf [±] (90% confidence value (+/-) in...) Temp [°C] (Temperature; number of observations) Temp conf [±] (90% confidence value (+/-) in...) Turbidity [FTU] (Value have to be treated with...) NOBS [#] (Turbidity; number of observations) Turbidity conf [±] (90% confidence value (+/-) in...)

Date/Time	Chl a [µg/l]	02 sat [%]	Sal (PSU)	Temp [°C]	Turbidity [FTU]
2016-01-01T00:00	0.00	1.34	2	0.01	2.08
2016-01-01T01:00	0.00	1.34	2	0.01	2.08
2016-01-01T02:00	0.00	1.34	2	0.01	2.08
2016-01-01T03:00	0.00	1.34	2	0.01	2.08
2016-01-01T04:00	0.00	1.34	2	0.01	2.08
2016-01-01T05:00	0.00	1.34	2	0.01	2.08
2016-01-01T06:00	0.00	1.34	2	0.01	2.08
2016-01-01T07:00	0.00	1.34	2	0.01	2.08
2016-01-01T08:00	0.00	1.34	2	0.01	2.08
2016-01-01T09:00	0.00	1.34	2	0.01	2.08
2016-01-01T10:00	0.00	1.34	2	0.01	2.08
2016-01-01T11:00	0.00	1.34	2	0.01	2.08
2016-01-01T12:00	0.00	1.34	2	0.01	2.08

single_loc...

```
{
  "geometry": {
    "type": "Point",
    "coordinates": [
      -2.6076,
      61.8833
    ]
  },
  "id": "kzHHg",
  "type": "Feature"
}
```



Konsumprisindeksen – SSB

ssb.no/priser-og-prisindekser/kon...

New Chrome available

Tabell 1 Konsumprisindeks (2015=100)

Konsumprisindeks (2015=100)			
	Indeks	Månedsendring (prosent)	12-måneders endring (prosent)
	Juni 2025	Mai 2025 - Juni 2025	Juni 2024 - Juni 2025
Indeks	137,8	0,2	3,0
alkoholfrie	139,9	1,4	4,7
ikke drikkevarer og	131,4	0,4	3,5
tøy	103,5	-2,5	-4,0
brensel	140,5	-0,8	4,2

Data med lite struktur

1001010110110001101011011011110110110101101100101011011100101011001

????????Y

010101100110010101101100011010110110111101101101011011010110010101101110

Velkommen

En hvilken som helst sekvens av 0'ere og 1'ere

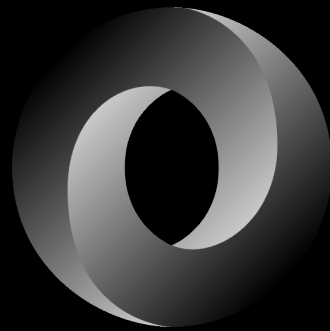
En sekvens med 0'ere og 1'ere begrenset til å kunne tolkes med UTF-8

Data med mye struktur

Arild er åtte år og 122cm høy.
Den andre gutten heter Emil. Han
er 6 år og er 1 m og 16 cm høy.
Olav er 62 cm lang og 4 måneder
gammel.

```
[  
  {  
    "name": "Arild",  
    "age": 8,  
    "height": 122  
  },  
  {  
    "name": "Emil",  
    "age": 6,  
    "height": 116  
  },  
  {  
    "name": "Olav",  
    "age": 0.33,  
    "height": 62  
  }  
]
```

JSON



DATATYPE

int

```
[
  {
    "name": "Arild",
    "age": 8,
    "height": 122
  },
  {
    "name": "Emil",
    "age": 6,
    "height": 116
  },
  {
    "name": "Olav",
    "age": 0.33,
    "height": 62
  }
]
```

DATATYPE

int

float

```
[
  {
    "name": "Arild",
    "age": 8,
    "height": 122
  },
  {
    "name": "Emil",
    "age": 6,
    "height": 116
  },
  {
    "name": "Olav",
    "age": 0.33,
    "height": 62
  }
]
```

DATATYPE

int

float

str

```
[
  {
    "name": "Arild",
    "age": 8,
    "height": 122
  },
  {
    "name": "Emil",
    "age": 6,
    "height": 116
  },
  {
    "name": "Olav",
    "age": 0.33,
    "height": 62
  }
]
```

DATATYPE

int

float

str

dict

```
[  
  {  
    "name": "Arild",  
    "age": 8,  
    "height": 122  
  },  
  {  
    "name": "Emil",  
    "age": 6,  
    "height": 116  
  },  
  {  
    "name": "Olav",  
    "age": 0.33,  
    "height": 62  
  }  
]
```

DATATYPE

int

float

str

dict

list

```
[  
  {  
    "name": "Arild",  
    "age": 8,  
    "height": 122  
  },  
  {  
    "name": "Emil",  
    "age": 6,  
    "height": 116  
  },  
  {  
    "name": "Olav",  
    "age": 0.33,  
    "height": 62  
  }  
]
```

DATATYPE

int 42

float 1.5

str "....."

dict { ...: ..., ...: ... }

list [..., ..., ...]

```
[
  {
    "name": "Arild",
    "age": 8,
    "height": 122
  },
  {
    "name": "Emil",
    "age": 6,
    "height": 116
  },
  {
    "name": "Olav",
    "age": 0.33,
    "height": 62
  }
]
```

Oppgave: hvor mange objekter (inkludert nøstede objekter) er det av hver type i denne filen?

int 42

float 1.5

str "....."

dict { ...: ..., ...: ... }

list [..., ..., ...]

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 42,
      "type": "Feature",
      "properties": {
        "Name": "Gruppetimer"
      },
      "geometry": {
        "coordinates": [
          5.32938628135696,
          60.38151104034819
        ],
        "type": "Point"
      }
    }
  ]
}
```

Oppgave: hvor mange objekter (inkludert nøstede objekter) er det av hver type i denne filen?

int

float

str

dict

list

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 42,
      "type": "Feature",
      "properties": {
        "Name": "Gruppetimer"
      },
      "geometry": {
        "coordinates": [
          5.32938628135696,
          60.38151104034819
        ],
        "type": "Point"
      }
    }
  ]
}
```

Oppgave: hvor mange objekter (inkludert nøstede objekter) er det av hver type i denne filen?

int

float

str

dict

list

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 42,
      "type": "Feature",
      "properties": {
        "Name": "Gruppetimer"
      },
      "geometry": {
        "coordinates": [
          5.32938628135696,
          60.38151104034819
        ],
        "type": "Point"
      }
    }
  ]
}
```

Oppgave: hvor mange objekter (inkludert nøstede objekter) er det av hver type i denne filen?

int

float

str

dict

list

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 42,
      "type": "Feature",
      "properties": {
        "Name": "Gruppetimer"
      },
      "geometry": {
        "coordinates": [
          5.32938628135696,
          60.38151104034819
        ],
        "type": "Point"
      }
    }
  ]
}
```

Oppgave: hvor mange objekter (inkludert nøstede objekter) er det av hver type i denne filen?

int

float

str

dict

list

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 42,
      "type": "Feature",
      "properties": {
        "Name": "Gruppetimer"
      },
      "geometry": {
        "coordinates": [
          5.32938628135696,
          60.38151104034819
        ],
        "type": "Point"
      }
    }
  ]
}
```

Oppgave: hvor mange objekter (inkludert nøstede objekter) er det av hver type i denne filen?

int

float

str

dict

list

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 42,
      "type": "Feature",
      "properties": {
        "Name": "Gruppetimer"
      },
      "geometry": {
        "coordinates": [
          5.32938628135696,
          60.38151104034819
        ],
        "type": "Point"
      }
    }
  ]
}
```

Oppgave: hvor mange objekter (inkludert nøstede objekter) er det av hver type i denne filen?

int

float

str

dict

list

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 42,
      "type": "Feature",
      "properties": {
        "Name": "Gruppetimer"
      },
      "geometry": {
        "coordinates": [
          5.32938628135696,
          60.38151104034819
        ],
        "type": "Point"
      }
    }
  ]
}
```

Oppgave: hvor mange objekter (inkludert nøstede objekter) er det av hver type i denne filen?

int

float

str

dict

list

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 42,
      "type": "Feature",
      "properties": {
        "Name": "Gruppetimer"
      },
      "geometry": {
        "coordinates": [
          5.32938628135696,
          60.38151104034819
        ],
        "type": "Point"
      }
    }
  ]
}
```

Oppgave: hvor mange objekter (inkludert nøstede objekter) er det av hver type i denne filen?

int

float

str

dict

list

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 42,
      "type": "Feature",
      "properties": {
        "Name": "Gruppetimer"
      },
      "geometry": {
        "coordinates": [
          5.32938628135696,
          60.38151104034819
        ],
        "type": "Point"
      }
    }
  ]
}
```

Oppgave: hvor mange objekter (inkludert nøstede objekter) er det av hver type i denne filen?

int

float

str

dict

list

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 42,
      "type": "Feature",
      "properties": {
        "Name": "Gruppetimer"
      },
      "geometry": {
        "coordinates": [
          5.32938628135696,
          60.38151104034819
        ],
        "type": "Point"
      }
    }
  ]
}
```

Oppgave: hvor mange objekter (inkludert nøstede objekter) er det av hver type i denne filen?

int

float

str

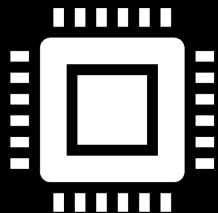
dict

list

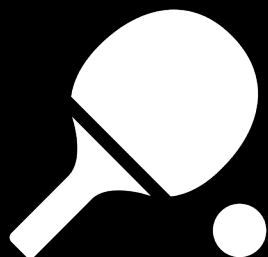
```
{
  "type": "FeatureCollection",
  "features": [
    {
      "id": 42,
      "type": "Feature",
      "properties": {
        "Name": "Gruppetimer"
      },
      "geometry": {
        "coordinates": [
          5.32938628135696,
          60.38151104034819
        ],
        "type": "Point"
      }
    }
  ]
}
```

PRAKTISK INFORMASJON

HVEM ER VI?



cvkdvg



Torstein Strømme



Administrator for INF100

- Tilrettelegging
- Emneregistrering
- Studieprogram



Jenny Thunes



Lab-admin

- Utsettelse
- Feil med retting



Wim Van den Broeck





Hilde Jordal



Crystal Chang Din



Aleksandr «Sasha» Popov

GRUPPELEDERE

Alexander, Amanda, Andreas, Are, Arshia, Åsmund,
Aurora, Eivind, Eivind, Emil, Emilie, Eva, Felix, Hevar,
Idiris, Izaak, Jo, Jonas, Jonas, Karl-Fredrik, Kristofers,
Ludvik, Marthe, Maya, Nickolay, Oskar, Runar, Sara,
Sheldon, Sigurd, Sondre, Stine, Stine og Thale



EN UKE I INF100

mandag 10:00
Lab publiseres

	Man	Tirs	Ons	Tors	Fre
8-10					
10-12	kickstart				
12-14					
14-16		forelesning			

I løpet av uken

- gruppetime
- drop-in gruppetime
- egenstudie

EN UKE I INF100

	Man	Tirs	Ons	Tors	Fre
8-10					
10-12	kickstart				
12-14					
14-16		forelesning			



Drop-in gruppetime

SEMESTEROVERSIKT TENTATIV

Ukenummer

34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Innleveringer

quiz1												quiz2		eksa..
	lab1			lab3		lab5		lab7						
		lab2		lab4		lab6		lab8						

SEMESTEROVERSIKT TENTATIV

Ukenummer

34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Innleveringer

quiz1										quiz2				eksa..
	lab1		lab3		lab5		lab7							
		lab2		lab4		lab6	lab8							

SEMESTEROVERSIKT TENTATIV

Ukenummer

34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Innleveringer

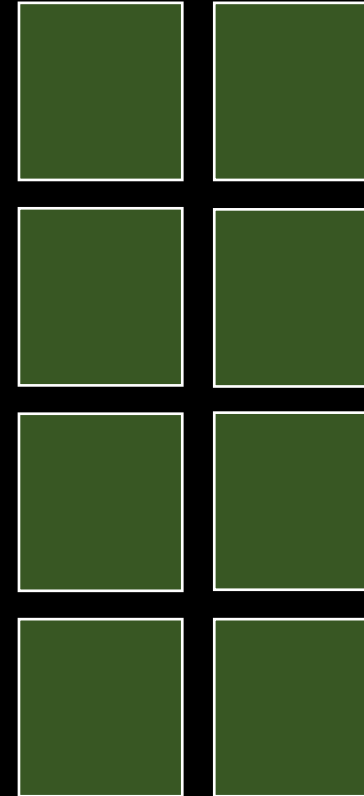
quiz1											quiz2			eksa..
	lab1		lab3		lab5		lab7							
		lab2		lab4		lab6	lab8							

Gruppetimen

f.-test	akt1		akt2		akt3		eksam.-forbr.
		pres1		pres2		pres3	

80% eksamen

8 x 2.5% lab

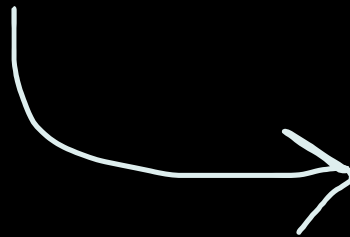


Karakter

ARBEIDSKRAV

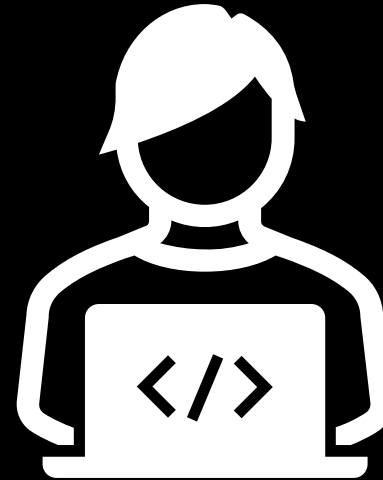
- Gjennomfør 2/2 quizer
- Gjennomfør 3/3 presentasjoner med god innsats
- Få minst 50% poeng på labene

Ved uforskyldte forhold som sykdom, heimevernsøvelse etc., søk om utsettelse eller fritak til Wim.

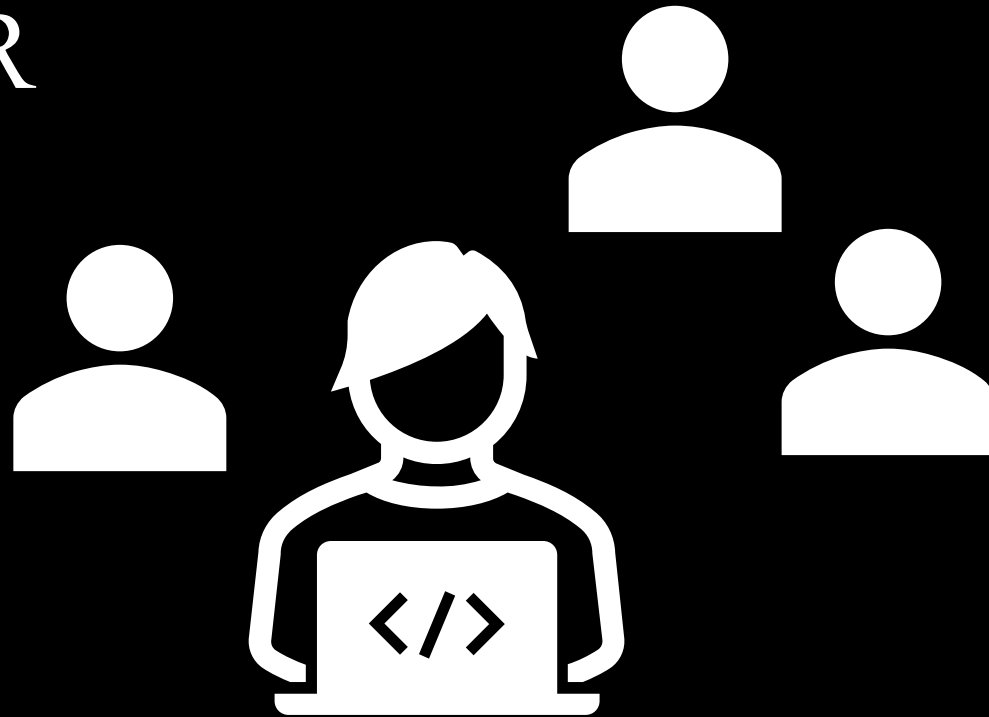


LAB

- 25 poeng per lab
 - Beregnet til ca 8-10 arbeidstimer
 - Frist: 2 uker
-
- Noen av labene inkluderer en oppgave basert på gruppeaktivitet. For å få disse poengene, må du møte i gruppen din



PRESENTASJONER



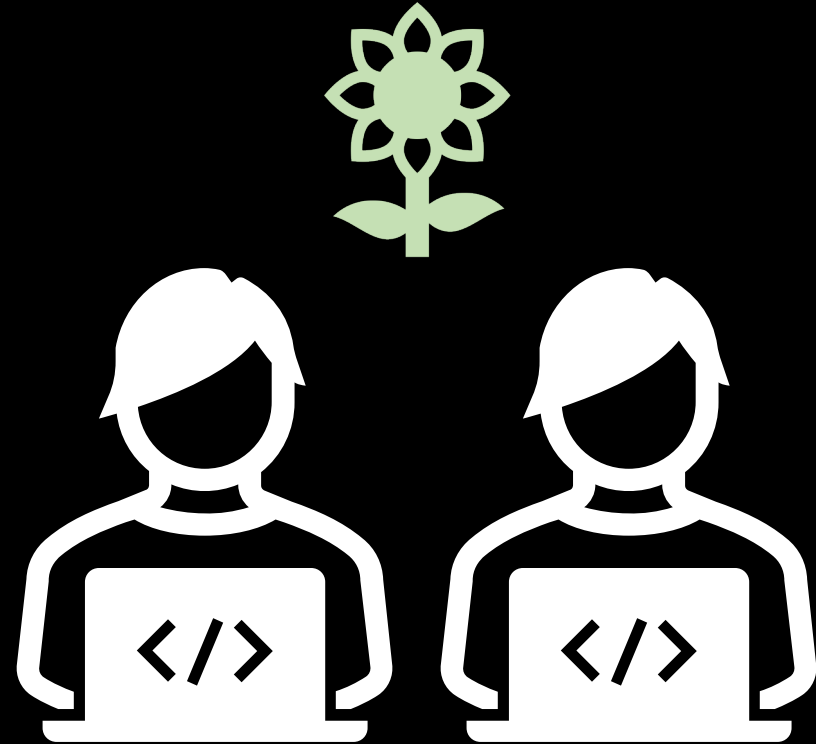
- Tre ganger i semesteret
- Publikum:
din gruppeleder + ca 3 medstudenter
- Varighet:
10 minutter presentasjon + spørsmål og diskusjon (x4)

VIKTIG!

MELD DEG INN I EN GRUPPE

SAMARBEID

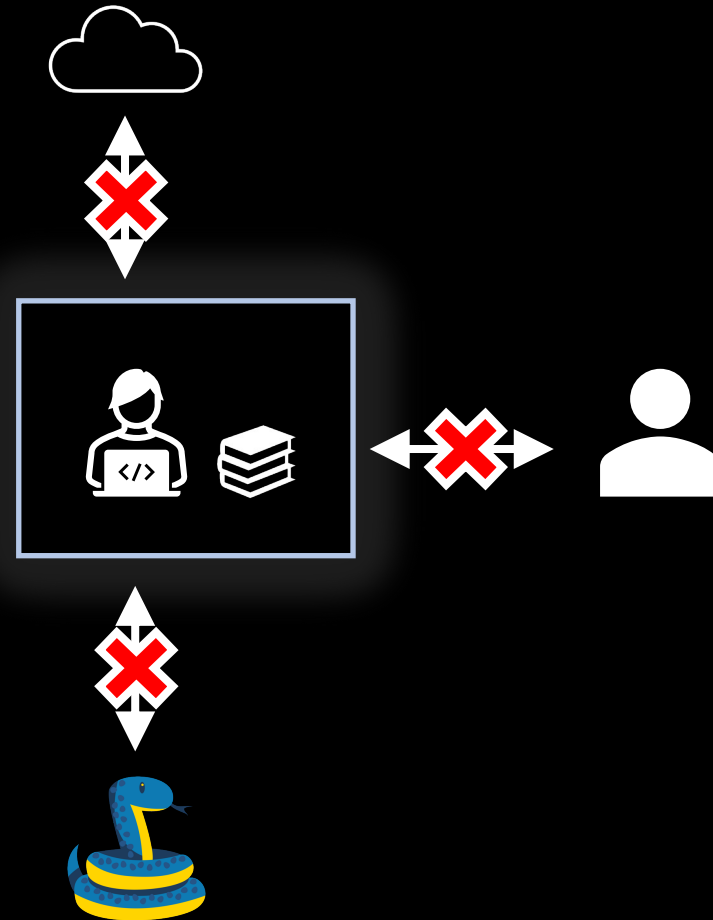
- Samarbeid er **bra**
- Innleveringer er **individuelle**
 - du må skrive og forstå alt selv
- **Viktig å sitere**
 - samarbeid
 - hjelp man mottar
 - kilder man benytter (inkludert KI)



husk **fullt navn** når du siterer

EKSAMEN

- Lukket digital eksamen
- Opp til 6 sider egne notater
- Viktig å sitere kilder



PLATTFORMER



mitt.uib.no

- Kunngjøringer
- Oppgaveinnlevering
- Strømming av forelesning
- Opptak av forelesning



inf100.ii.uib.no

- Kursnotater
- Oppgavetekster



CodeGrade

- Automatisk retting av lab'er
- Manuell feedback på lab'er
- Tilgang via mitt.uib.no



Discord

- Spørsmål og svar
- Tips og triks

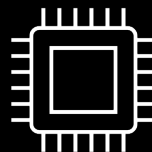
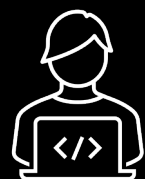
NASJONAL FORKUNNSKAPSTEST

- I gruppetimene denne uken
 - Denne uken: velg hvilken gruppetime du selv ønsker
- For alle studenter i emnet
 - Hvis du ikke har anledning til å delta: send begrunnelse til Wim

SNARVEIPRØVE

- For deg som allerede kan programmere (men helt frivillig)
- Automatisk rettet
- Over 70% →
 - Alle presentasjoner regnes som bestått
 - Alle gruppeaktiviteter i alle lab'er regnes som bestått
 - Poeng for labX beregnes som `max(labX_score, lab8_score)`
- Tirsdag 2. september kl 16:15 – 17:15
 - Fysisk oppmøte i RFB Aud 1 like etter forelesning
 - Tillatte hjelpemidler: enkel kalkulator, penn og papir
 - Ta med: student-id og egen laptop. Sjekk at Safe Exam Browser virker.

HVA ER PROGRAMMERING?



```
# Input
```

```
hourly_wage = 200
```

```
daily_hours = 7.5
```

```
days_worked = 20
```

```
# Calculation
```

```
daily_earnings = hourly_wage * daily_hours
```

```
total_earnings = daily_earnings * days_worked
```

```
# Output
```

```
print(f'Total earnings: {total_earnings}')
```

→ # Input

```
hourly_wage = 200  
daily_hours = 7.5  
days_worked = 20
```

Calculation

```
daily_earnings = hourly_wage * daily_hours  
total_earnings = daily_earnings * days_worked
```

Output

```
print(f'Total earnings: {total_earnings}')
```

Variabler	Verdier

→ # Input

```
hourly_wage = 200  
daily_hours = 7.5  
days_worked = 20
```

Calculation

```
daily_earnings = hourly_wage * daily_hours  
total_earnings = daily_earnings * days_worked
```

Output

```
print(f'Total earnings: {total_earnings}')
```

Variabler	Verdier
	200

→ # Input

```
hourly_wage = 200  
daily_hours = 7.5  
days_worked = 20
```

Calculation

```
daily_earnings = hourly_wage * daily_hours  
total_earnings = daily_earnings * days_worked
```

Output

```
print(f'Total earnings: {total_earnings}')
```

Variabler	Verdier
hourly_wage	200

Input

hourly_wage = 200

daily_hours = 7.5

days_worked = 20

Calculation

daily_earnings = hourly_wage * daily_hours

total_earnings = daily_earnings * days_worked

Output

print(f'Total earnings: {total_earnings}')

Variabler	Verdier
hourly_wage	200

→ # Input

hourly_wage = 200

daily_hours = 7.5

days_worked = 20

Calculation

daily_earnings = hourly_wage * daily_hours

total_earnings = daily_earnings * days_worked

Output

print(f'Total earnings: {total_earnings}')

Variable	Verdier
hourly_wage	200
	7.5

→ # Input

hourly_wage = 200

daily_hours = 7.5

days_worked = 20

Calculation

daily_earnings = hourly_wage * daily_hours

total_earnings = daily_earnings * days_worked

Output

print(f'Total earnings: {total_earnings}')

Variable	Verdier
hourly_wage	200
daily_hours	7.5

`# Input`

`hourly_wage = 200`

`daily_hours = 7.5`

→ `days_worked = 20`

`# Calculation`

`daily_earnings = hourly_wage * daily_hours`

`total_earnings = daily_earnings * days_worked`

`# Output`

`print(f'Total earnings: {total_earnings}')`

Variable	Verdier
hourly_wage	200
daily_hours	7.5

→ # Input

hourly_wage = 200

daily_hours = 7.5

days_worked = 20

Calculation

daily_earnings = hourly_wage * daily_hours

total_earnings = daily_earnings * days_worked

Output

print(f'Total earnings: {total_earnings}')

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
	20

Input

hourly_wage = 200

daily_hours = 7.5

→ days_worked = 20

Calculation

daily_earnings = hourly_wage * daily_hours

total_earnings = daily_earnings * days_worked

Output

print(f'Total earnings: {total_earnings}')

Variable	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20

```
# Input
```

```
hourly_wage = 200
```

```
daily_hours = 7.5
```

```
days_worked = 20
```

```
# Calculation
```

```
→ daily_earnings = hourly_wage * daily_hours
```

```
total_earnings = daily_earnings * days_worked
```

```
# Output
```

```
print(f'Total earnings: {total_earnings}')
```

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20

Input

hourly_wage = 200

daily_hours = 7.5

days_worked = 20

Calculation

→ daily_earnings = hourly_wage * daily_hours

total_earnings = daily_earnings * days_worked

Output

print(f'Total earnings: {total_earnings}')

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20

hourly_wage * daily_hours

Input

hourly_wage = 200

daily_hours = 7.5

days_worked = 20

Calculation

→ daily_earnings = hourly_wage * daily_hours

total_earnings = daily_earnings * days_worked

Output

print(f'Total earnings: {total_earnings}')

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20

200 * daily_hours

Input

hourly_wage = 200

daily_hours = 7.5

days_worked = 20

Calculation

→ daily_earnings = hourly_wage * daily_hours

total_earnings = daily_earnings * days_worked

Output

print(f'Total earnings: {total_earnings}')

Variable	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20

200 * 7.5

Input

hourly_wage = 200

daily_hours = 7.5

days_worked = 20

Calculation

→ daily_earnings = hourly_wage * daily_hours

total_earnings = daily_earnings * days_worked

Output

print(f'Total earnings:')

1500.0

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20

```
# Input
```

```
hourly_wage = 200
```

```
daily_hours = 7.5
```

```
days_worked = 20
```

```
# Calculation
```

```
→ daily_earnings = hourly_wage * daily_hours
```

```
total_earnings = daily_earnings * days_worked
```

```
# Output
```

```
print(f'Total earnings: {total_earnings}')
```

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
	1500.0

```
# Input
```

```
hourly_wage = 200
```

```
daily_hours = 7.5
```

```
days_worked = 20
```

```
# Calculation
```

```
→ daily_earnings = hourly_wage * daily_hours
```

```
total_earnings = daily_earnings * days_worked
```

```
# Output
```

```
print(f'Total earnings: {total_earnings}')
```

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
daily_earnings	1500.0

```
# Input
```

```
hourly_wage = 200
```

```
daily_hours = 7.5
```

```
days_worked = 20
```

```
# Calculation
```

```
daily_earnings = hourly_wage * daily_hours
```

```
→ total_earnings = daily_earnings * days_worked
```

```
# Output
```

```
print(f'Total earnings: {total_earnings}')
```

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
daily_earnings	1500.0

Input

```
hourly_wage = 200  
daily_hours = 7.5  
days_worked = 20
```

Calculation

```
daily_earnings = hourly_wage * daily_hours  
→ total_earnings = daily_earnings * days_worked
```

Output

```
print(f'Total earnings: {total_earnings}')  
# Output: Total earnings: 1500.0
```

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
daily_earnings	1500.0

daily_earnings * days_worked

Input

```
hourly_wage = 200  
daily_hours = 7.5  
days_worked = 20
```

Calculation

→

```
daily_earnings = hourly_wage * daily_hours  
total_earnings = daily_earnings * days_worked
```

Output

```
print(f'Total earnings: {total_earnings}')
```

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
daily_earnings	1500.0

1500.0 * days_worked

Input

```
hourly_wage = 200  
daily_hours = 7.5  
days_worked = 20
```

Calculation

```
daily_earnings = hourly_wage * daily_hours  
→ total_earnings = daily_earnings * days_worked
```

Output

```
print(f'Total earnings: {total_earnings}')  
# Output: Total earnings: 1500.0
```

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
daily_earnings	1500.0

1500.0 * 20

Input

```
hourly_wage = 200  
daily_hours = 7.5  
days_worked = 20
```

Calculation

```
daily_earnings = hourly_wage * daily_hours  
→ total_earnings = daily_earnings * days_worked
```

Output

```
print(f'Total earnings: {total_earnings}')  
Total earnings: 30000.0
```

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
daily_earnings	1500.0

Input

```
hourly_wage = 200  
daily_hours = 7.5  
days_worked = 20
```

Calculation

→

```
daily_earnings = hourly_wage * daily_hours  
total_earnings = daily_earnings * days_worked
```

Output

```
print(f'Total earnings: {total_earnings}')
```

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
daily_earnings	1500.0
	30000.0

Input

```
hourly_wage = 200  
daily_hours = 7.5  
days_worked = 20
```

Calculation

→

```
daily_earnings = hourly_wage * daily_hours  
total_earnings = daily_earnings * days_worked
```

Output

```
print(f'Total earnings: {total_earnings}')
```

Variable	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
daily_earnings	1500.0
total_earnings	30000.0

```
# Input
```

```
hourly_wage = 200
```

```
daily_hours = 7.5
```

```
days_worked = 20
```

```
# Calculation
```

```
daily_earnings = hourly_wage * daily_hours
```

```
total_earnings = daily_earnings * days_worked
```

```
# Output
```

```
→ print(f'Total earnings: {total_earnings}')
```

Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
daily_earnings	1500.0
total_earnings	30000.0

Input

```
hourly_wage = 200  
daily_hours = 7.5  
days_worked = 20
```

Calculation

```
daily_earnings = hourly_wage * daily_hours  
total_earnings = daily_earnings * days_worked
```

Output

→

```
print(f'Total earnings: {total_earnings}')
```

Variable	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
daily_earnings	1500.0
total_earnings	30000.0



```
# Input
```

```
hourly_wage = 200
```

```
daily_hours = 7.5
```

```
days_worked = 20
```

```
# Calculation
```

```
daily_earnings = hourly_wage * daily_hours
```

```
total_earnings = daily_earnings * days_worked
```

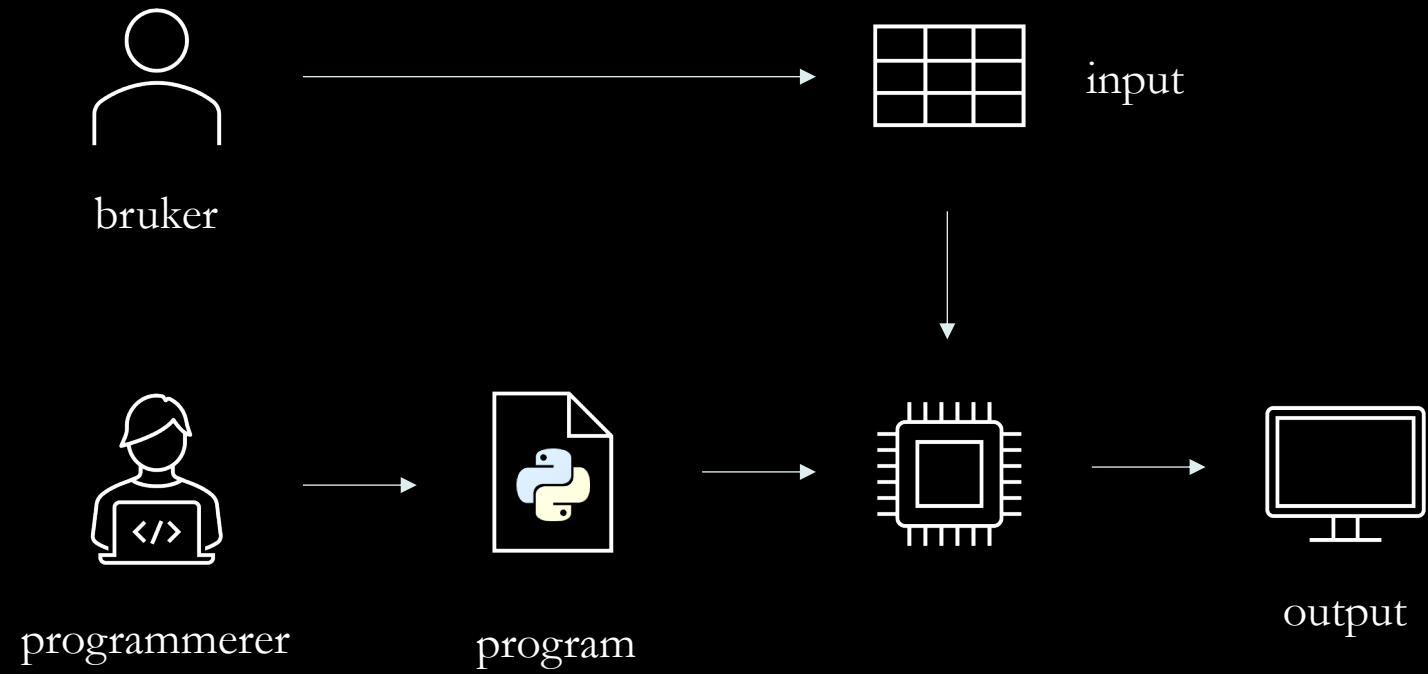
```
# Output
```

```
print(f'Total earnings: {total_earnings}')
```



Variabler	Verdier
hourly_wage	200
daily_hours	7.5
days_worked	20
daily_earnings	1500.0
total_earnings	30000.0





```
hourly_wage = 200  
daily_hours = 7.5  
days_worked = 20  
  
daily_earnings = hourly_wage * daily_hours  
total_earnings = daily_earnings * days_worked  
  
print(f'Total earnings: {total_earnings}')
```



```
print('Total earnings: 30000.0')
```

INPUT

Ulike måter å få input

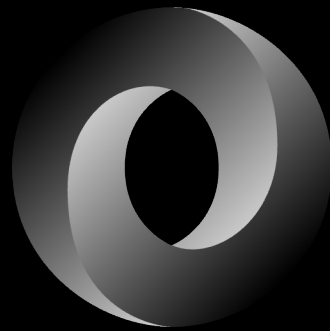
- Brukeren kan endre på verdier i selve programmet 
- Programmet er interaktivt 
- Programmet leser input fra en fil eller fra internett 
- Argumenter på kommandolinjen når programmet starter 

INPUT

Ulike måter å få input

- Brukeren kan endre på verdier i selve programmet 
- Programmet er interaktivt 
- Programmet leser input fra en fil eller fra internett 
- Argumenter på kommandolinjen når programmet starter 

JSON



```
{  
  "hourly_wage": 200,  
  "daily_hours": 7.5,  
  "days_worked": 20  
}
```

total_earnings.py

```
from pathlib import Path
import json

file_contents = (Path("data.json")
                  .read_text(encoding="utf-8"))
data = json.loads(file_contents)

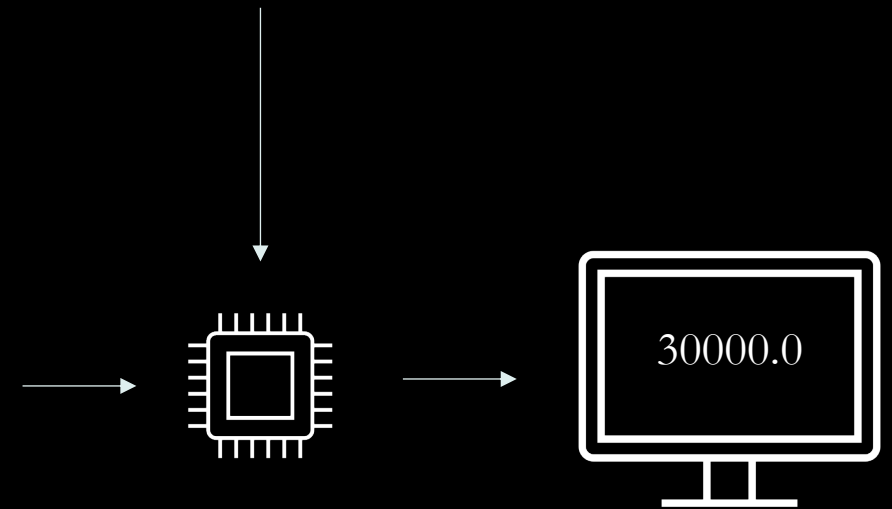
hourly_wage = data["hourly_wage"]
daily_hours = data["daily_hours"]
days_worked = data["days_worked"]

daily_earnings = hourly_wage * daily_hours
total_earnings = daily_earnings * days_worked

print(f"Total earnings: {total_earnings}")
```

data.json

```
{
  "hourly_wage": 200,
  "daily_hours": 7.5,
  "days_worked": 20
}
```



VIKTIG!

MELD DEG INN I EN GRUPPE