FAKULTAS ILMU KOMPUTER
UNIVERSITAS INDONESIA

Topic 0:
Introduction to Programming and Computer System

CSGE601020 - Dasar-Dasar Pemrograman 1

Perkenalan

Raja Oktovin P. Damanik, M.Sc.
(rajaoktovin@cs.ui.ac.id)

Latar Belakang Pendidikan:

- S1: Fasilkom UI (2010)
- S2: Institute of Logic, Language, and Computation (ILLC) Universiteit van Amsterdam (2016)

Hobi:

Badminton, Renang, Kulineran, Nonton, Nyanyi

Dasar-Dasar Pemrograman 1

- 4 SKS
  - 200 menit kegiatan tatap muka per minggu
  - 240 menit kegiatan akademik terstruktur
  - 240 menit kegiatan belajar mandiri
- Jadwal Kelas
  - Kelas A:
    - Senin 10.00-11.40
    - Rabu 10.00-11.40
    - Jumat 14.00-15.40
  - Kelas F:
    - Senin 13.00-14.40
    - Rabu 13.00-14.40
    - Jumat 14.00-15.40

Peraturan Kuliah

- **Kehadiran**
  - Minimum 75% kehadiran peserta kuliah dari kehadiran dosen supaya dapat mengikuti ujian akhir.
  - Pada kasus COVID, ada keringanan.
- **Tidak dapat mengikuti ujian (mid & final)**
  - Ada aturan yang membatasi siapa yang berhak ujian susulan
  - Surat sakit dari rumah sakit
- **Terlambat menyerahkan tugas**
  - Tugas terlambat tidak akan diterima (plan ahead)
  - Pada kasus sangat khusus, dibicarakan dengan dosen
- **Tidak dapat mengikuti laboratorium**
  - Menghubungi asisten untuk menjelaskan penyebabnya

Penilaian Capaian

Lab (10x)	10%
Tugas Pemrograman (4x)	20%
Kuis	15%
UTS	25%
UAS	35%

Nilai minimal kelulusan di UI adalah C.

Nilai	Min	Max
A	85	100
A-	80	85
B+	75	80
B	70	75
B-	65	70
C+	60	65
C	55	60
C-	50	55
D	40	50
E	0	40

Sifat-sifat Tugas

Aktivitas kuliah seperti penugasan dirancang dengan tujuannya masing-masing secara spesifik. Penilaian capaian mahasiswa dibuat serelevan dan memiliki variasi yang cukup untuk mengembangkan kemampuan Anda di berbagai aspek.

Jangan segan untuk konfirmasi ke dosen sifat penugasannya apakah individu atau justru dikerjakan secara berkelompok; apakah boleh diskusi atau tidak.

Ada aktivitas yang mensyaratkan mahasiswa harus mengerjakannya berkelompok.

Ada aktivitas yang mensyaratkan mahasiswa harus mengerjakanya secara individu (kumpulkan hasil karya sendiri).

Pada beberapa tugas individu, ada yang boleh diskusi dan ada yang tidak boleh diskusi.

Mengapa ada tugas individu? Mengapa ada tugas kelompok?

DDP 1: Kecurangan dan Plagiarisme

- Belajar bersama dan diskusi secara umum disarankan dan merupakan hal yang baik. Namun, semua lab dan tugas DDP 1 bersifat individu (kumpulkan karya Anda masing-masing).
- **Anda harus melalui proses belajar tersebut sendiri hingga menghasilkan sebuah program yang diminta!**
- Ada ruang aktivitas lain di mana kita buat program bersama-sama.
- Lebih baik susah sekarang daripada susah nanti. Jangan curang!

- Contoh curang untuk kuliah DDP 1:
  - Menyalin program teman
  - Meminta program teman yang sudah jadi, dipahami, dikode ulang dengan perubahan
  - Mengirim programnya ke teman yang kesulitan karena sudah "teman"
  - Memprogram bersama-sama (keroyokan)
  - Mencari program di internet dan menggunakannya
- Biasanya mahasiswa curang karena *running out of time* atau menyerah sehingga mengambil jalan pintas.
- Diskusi dan meminta bantuan boleh. Pada kasus terburuk, boleh kontak dosen atau asdos!
- Kecurangan bisa mengakibatkan Anda langsung mendapat **nilai E.**
- **Jangan segan konfirmasi dosen apa yang boleh dan tidak di setiap aktivitas perkuliahan.**

Acknowledgement

This slide is an adapted version of ‘Introduction to Programming’ slides used in DDP1 Course (2020/2021) by **Hafizh Rafizal Adnan, M.Kom.**

Several materials are reused from ‘Komputer dan Programming’ slides used in Dasar-Dasar Pemrograman 1 dengan Python (CSGE601020/4 SKS) Course (https://ocw.ui.ac.id/course/view.php?id=142) by **Fariz Darari, Ph.D.**

Some of the design assets used in these slides were provided by ManyPixels under an nonexclusive, worldwide copyright license to download, copy, modify, distribute, perform, and use the assets provided from ManyPixels for free, including for commercial purposes, without permission from or attributing the creator or ManyPixels.

Some additional contents, illustrations and visual design elements are provided by **Lintang Matahari Hasani, M.Kom.** *(lintang.matahari01[at]cs.ui.ac.id)*

In this session, you will learn ...

- Computer: Definition and Architecture
- An Overview of Programming
- Computational Thinking
- How Python works: Syntax and some simple examples

Triggering Question 1

What is a computer?

What is a computer?

A computer is a machine that:

1. **stores data** (numbers, words, pictures),
2. **interacts with devices** (the monitor screen, sound system, printer), and
3. **executes programs**.

- ➔ A computer program is a finite sequence of instructions and decisions that the computer carries out to achieve a task.
- ➔ A computer executes instructions very rapidly.
- ➔ A computer is a general purpose machine.

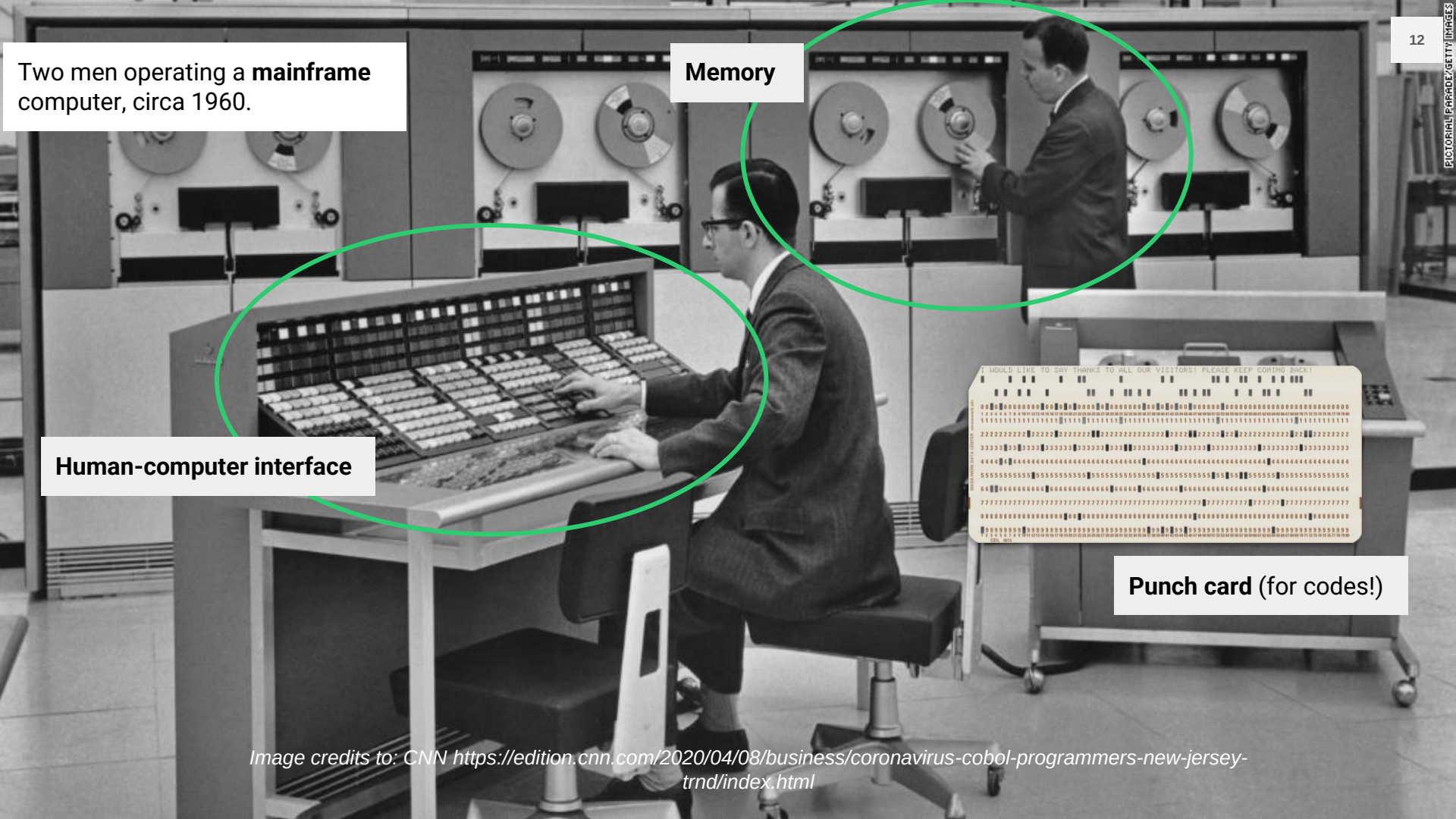


Image credits to: CNN https://edition.cnn.com/2020/04/08/business/coronavirus-cobol-programmers-new-jersey-trnd/index.html

Coding in 1960s

Punched card from a Fortran program: `Z(1) = Y + W(1)`

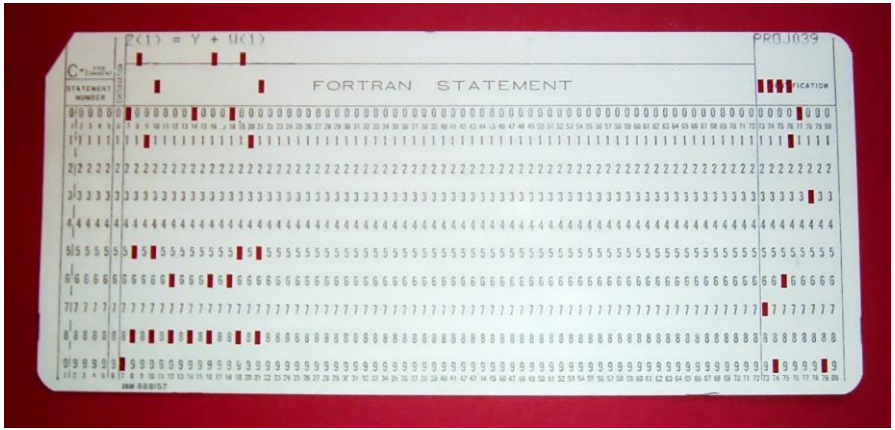


Image credits to: Arnold Reinhold

Punched card from a Fortran program: `println("Hello,world.");`

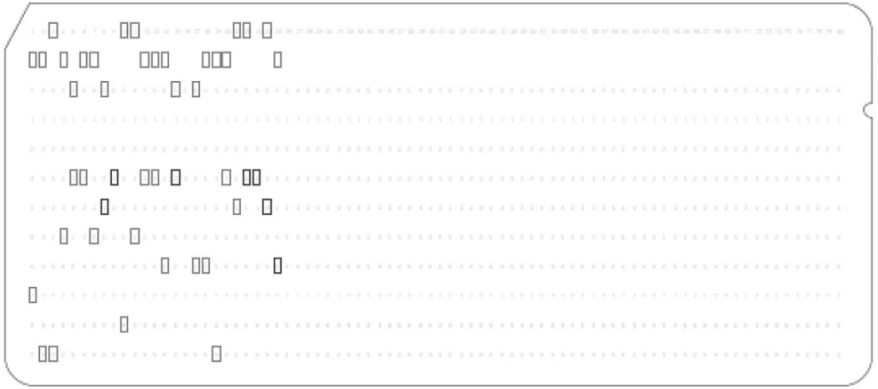
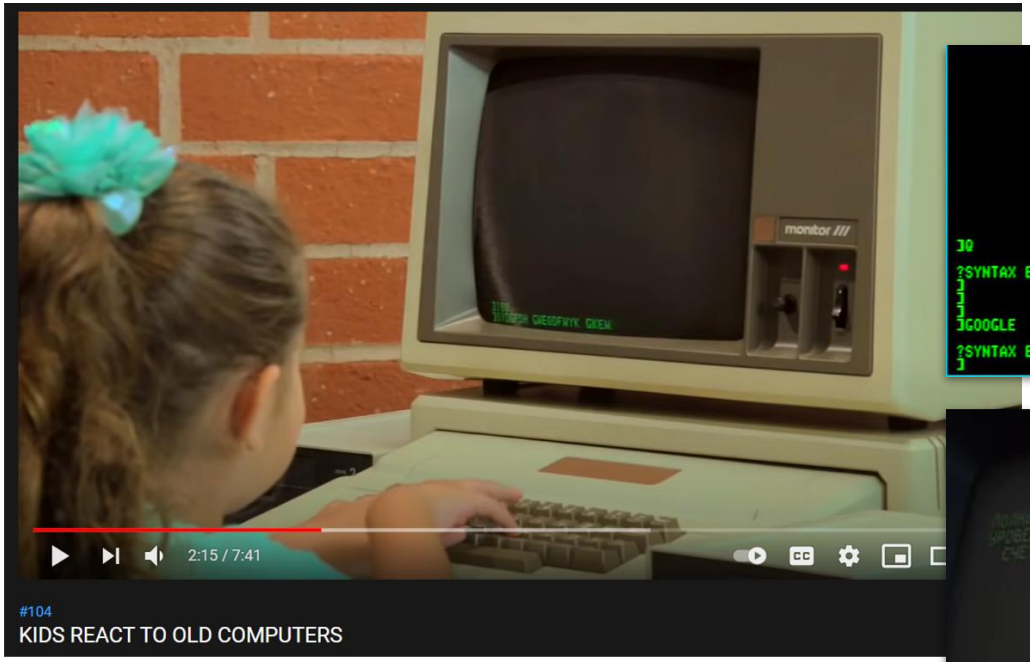
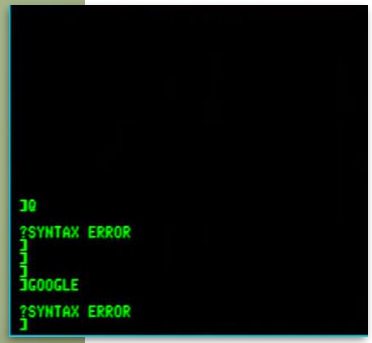
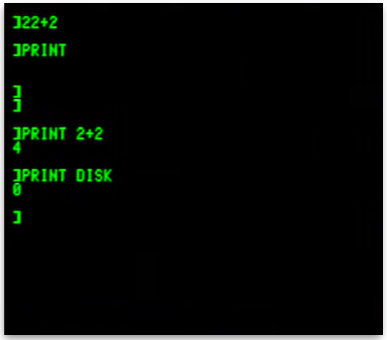


Image credits to:
https://www.jeffreythompson.org/blog/2015/02/20/punch-card-encoding/



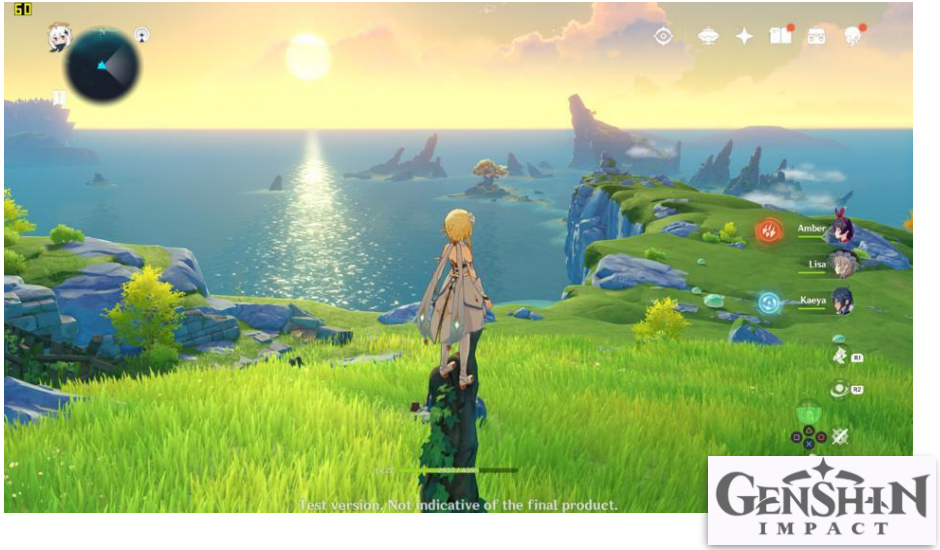


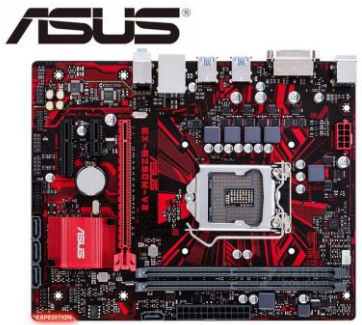


Kids react to an OLD PC (personal computer) from 1970s
https://www.youtube.com/watch?v=PF7EpEnglgk

The original Tetris game running on
Диалоговый вычислительный комплекс (DVK-2) personal computer
https://www.youtube.com/watch?v=O0gAgQQHFcQ

Hardware vs Software





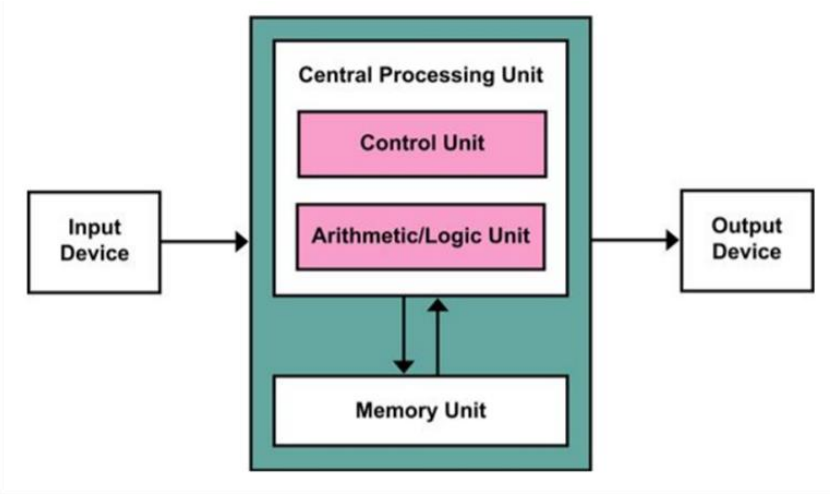
Spesifikasi komputer – Contoh dunia nyata (cont.)

ASUS VivoBook
F541UJ

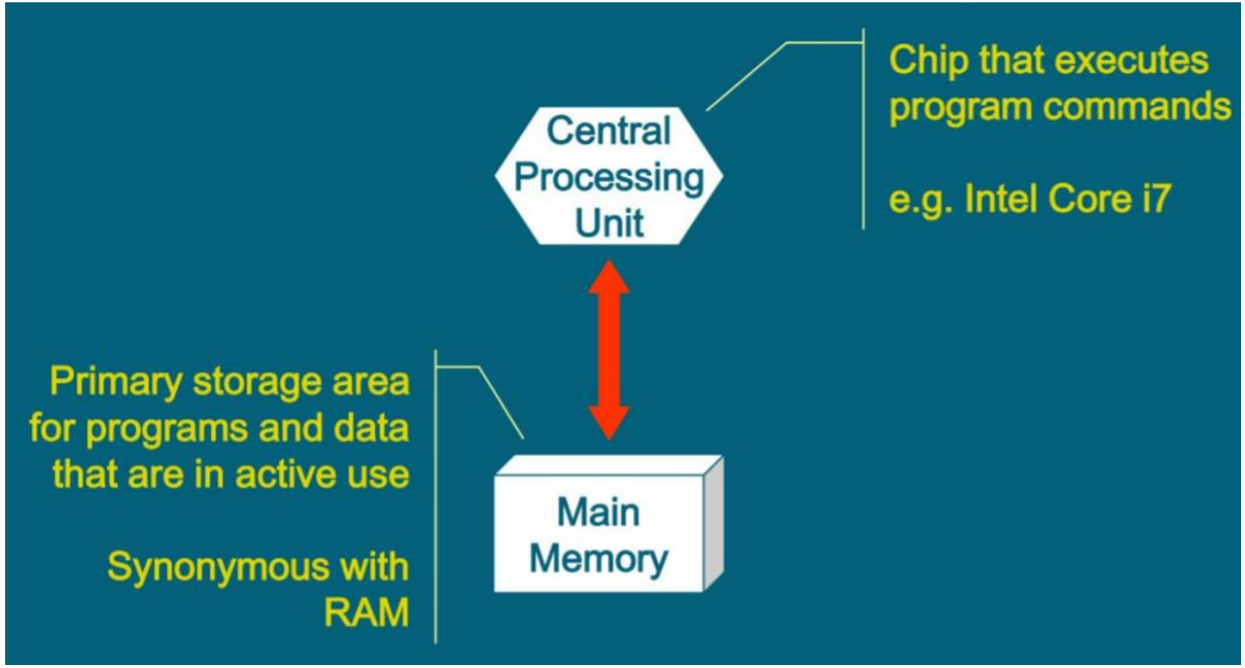
Main specs

Processor	Screen	Hard drive	RAM	Operating system	Graphics card
Intel Core i5-7xxx i5-7200U 2,5GHz	15.6" 1366x768	1000GB HDD	12GB DDR4-SDRAM	Windows 10 Home	NVIDIA GeForce 920M

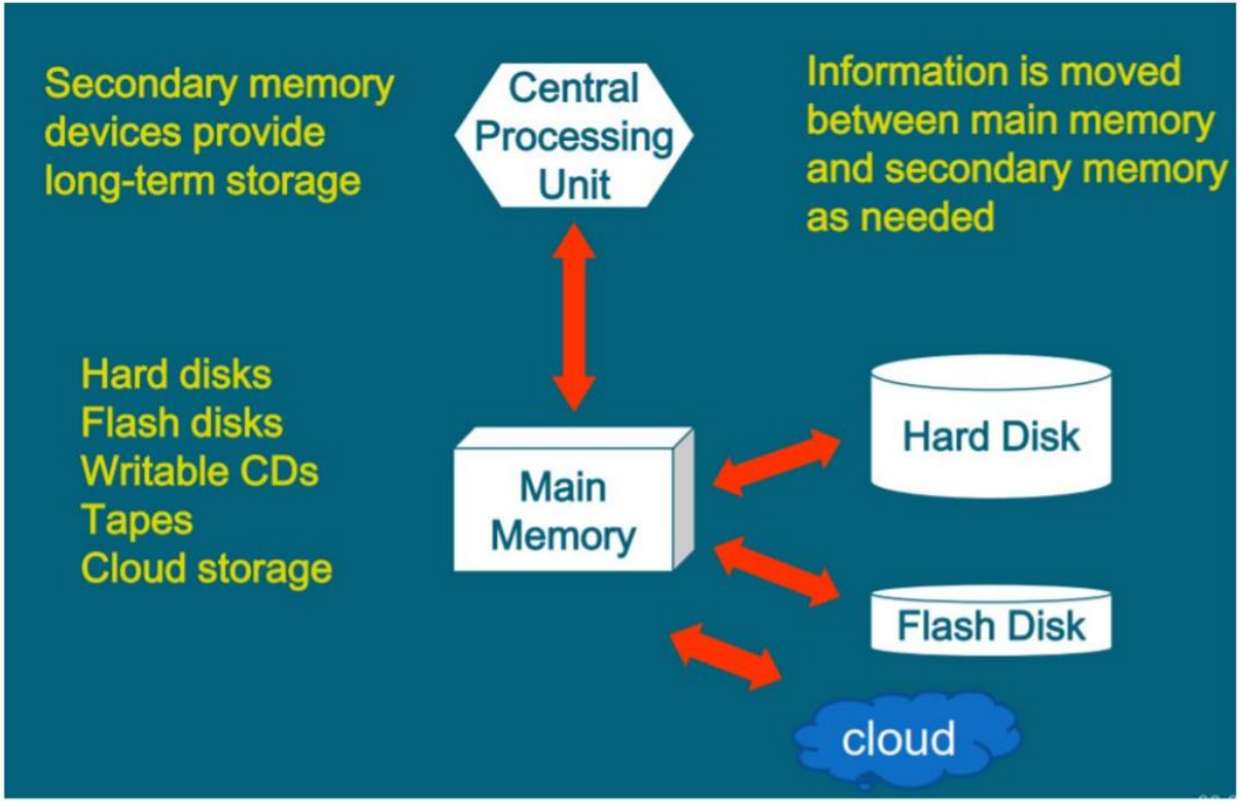
Computer Architecture (Von Neumann, 1945)



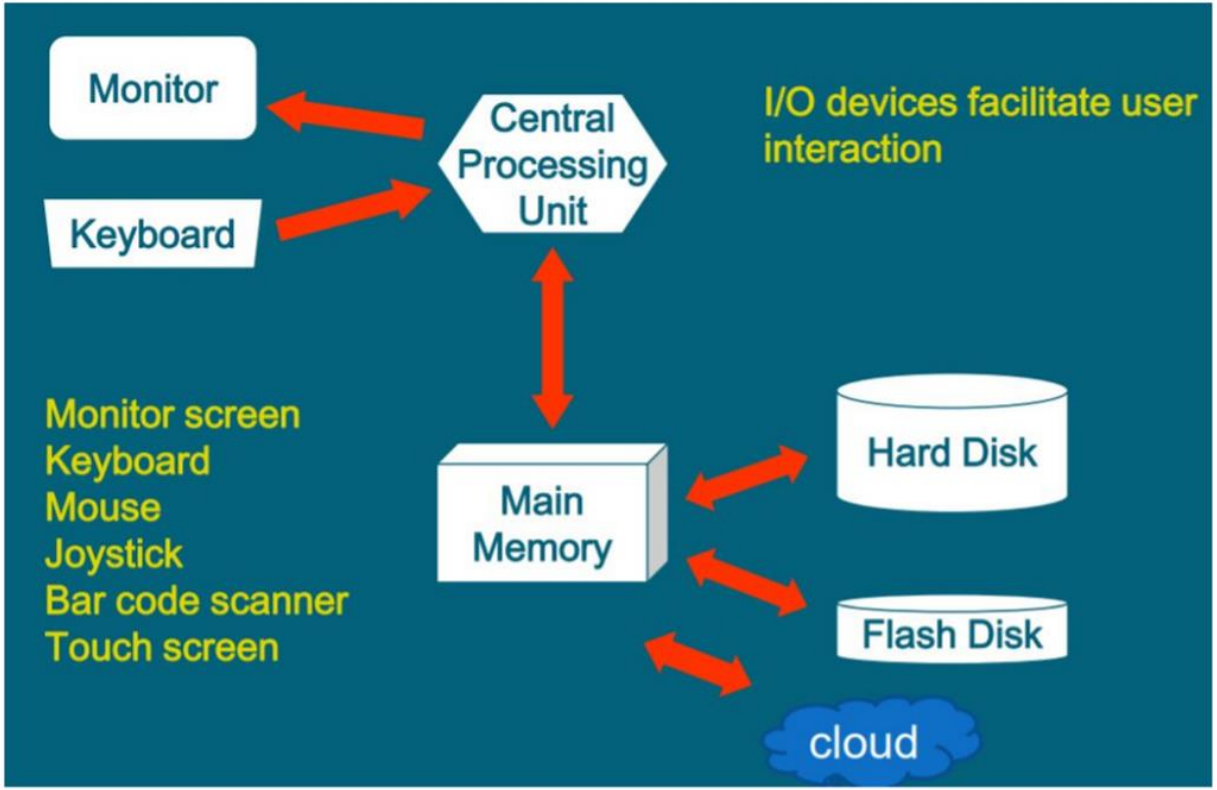
CPU dan Main Memory



Secondary Memory Device



Input / Output Devices



Debug

- ➔ A program can have errors (known as bugs).
- ➔ **Debugging**: the process of finding errors in a program and correct them.

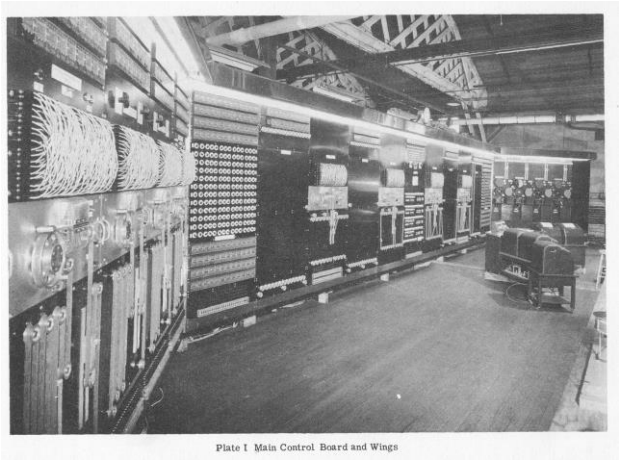
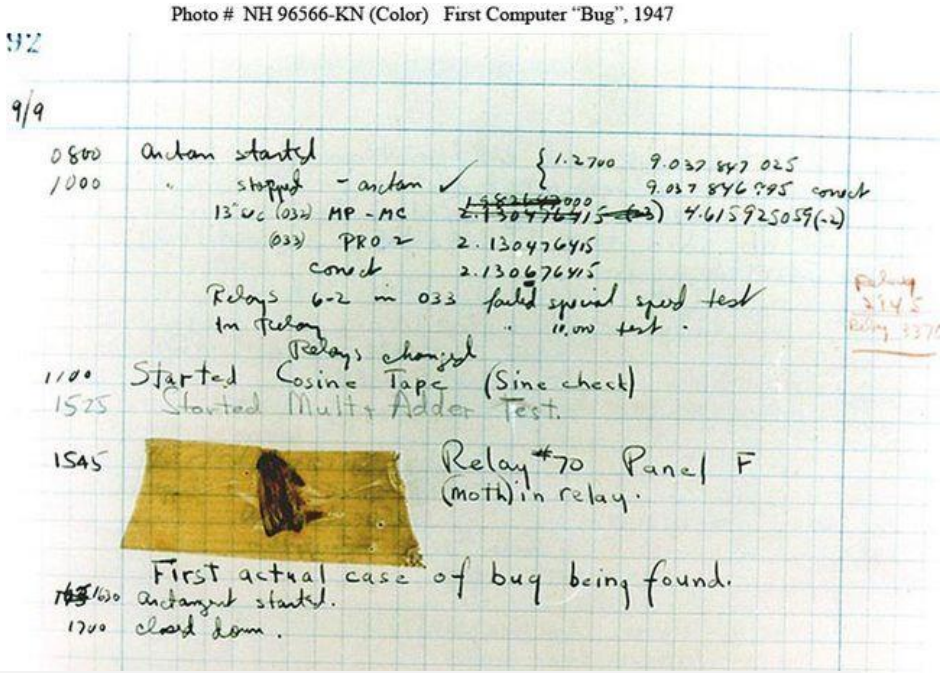


Plate I Main Control Board and Wings

https://imgur.com/gallery/Eg32Y



The very first documented computer bug from Harvard's Mark II Computer

https://www.nationalgeographic.org/thisday/sep9/worlds-first-computer-bug/

What is computer science?

Computer Science is the **study of computers and computational systems**.

Unlike electrical and computer engineers, computer scientists **deal mostly with software and software systems**; this includes their theory, design, development, and application.

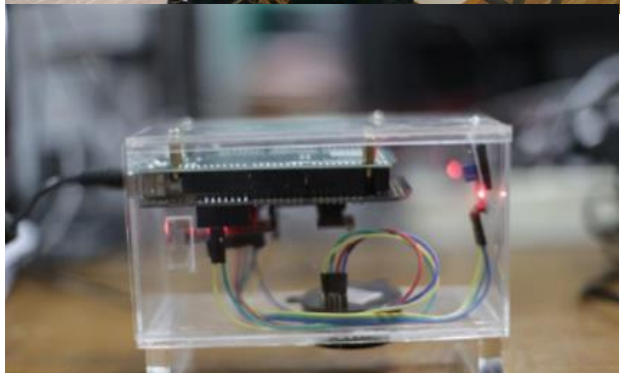


Image credits to: kuliahdimana.id, testingtime.com, Habibie et al. (2016)

Some CS Domains ...

Artificial intelligence	Computer systems and networks
Database systems	Human-computer interaction
Computer vision	Numerical analysis
Software engineering	Programming languages
Information security	Bioinformatics

etc ...

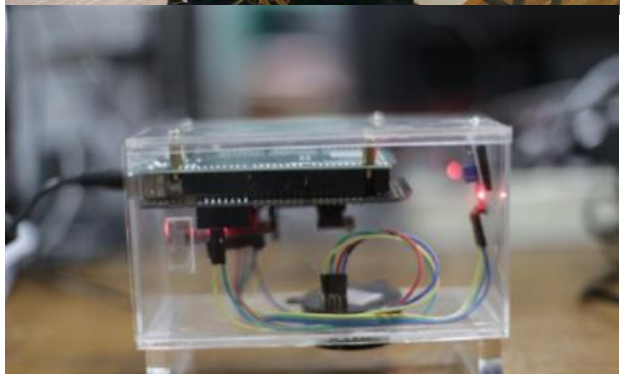


Image credits to: kuliahdimana.id, testingtime.com, Habibie et al. (2016)

What is a program?

A program is **a collection of instructions** for a computer to do certain tasks (**problem solving**, etc.)

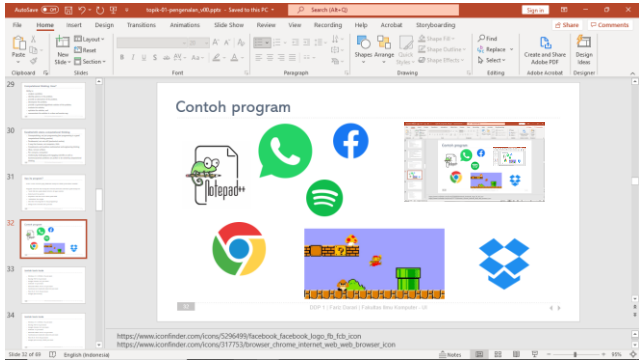
A computer program consists of a finite sequence of basic operations

- ➔ Put a red dot onto a certain screen position
- ➔ Send letter P to the printer
- ➔ Add up to numbers
- ➔ Get a number from a certain location in memory
- ➔ If this value is negative, stop the program
- ➔ Repeat this instruction a thousand times

An example:

```
# This program says hello and asks for my name.

print('Hello, world!')
print('What is your name?')    # ask for their name
myName = input()
print('It is good to meet you, ' + myName)
print('The length of your name is:')
print(len(myName))
```

Contoh program



What is programming?

Computer programming is a way of giving computers **instructions** about what they should do next.

These instructions are known as **code**, and computer programmers write code to solve problems or perform a task.

An example:

```
# This program says hello and asks for my name.

print('Hello, world!')
print('What is your name?')    # ask for their name
myName = input()
print('It is good to meet you, ' + myName)
print('The length of your name is:')
print(len(myName))
```

Programming languages

Programs are expressed in **unambiguous**, **precise** way using programming languages. (Natural language has ambiguity and imprecision.)

- ➔ Every program has a structure in precise form, called its syntax and has precise meaning called its semantics.
- ➔ Process of writing an algorithm in a programming language often called coding.

An example:

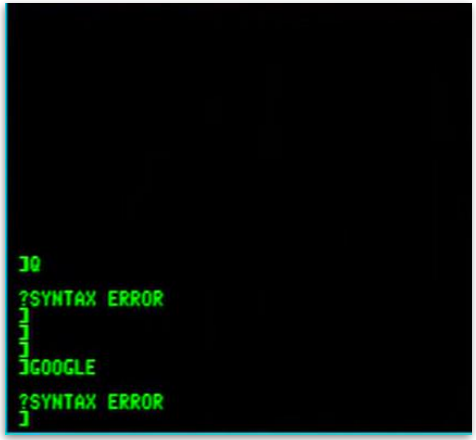
```
# This program says hello and asks for my name.

print('Hello, world!')
print('What is your name?')    # ask for their name
myName = input()
print('It is good to meet you, ' + myName)
print('The length of your name is:')
print(len(myName))
```

Syntax and Semantics

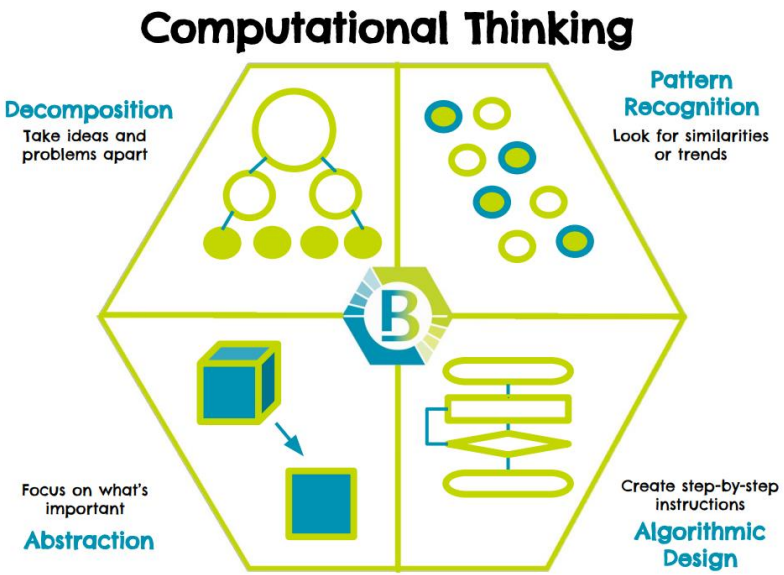
The **syntax** rules of a language **define how we can put together symbols, keywords, and identifiers** to make a valid program.

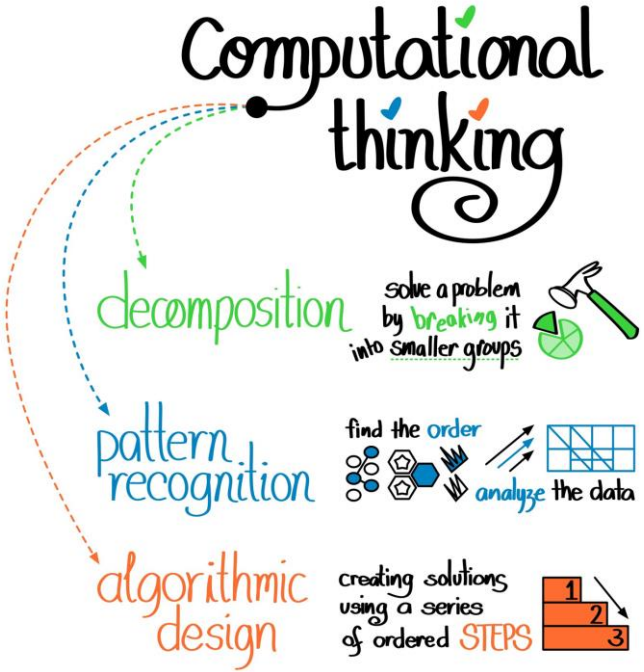
The **semantics** of a program statement **define what it means** (its purpose or role in a program).

- ➔ A program that is syntactically correct is not necessarily logically (semantically) correct.
- ➔ A program will always do what we tell it to do, not what we meant to tell it to do.



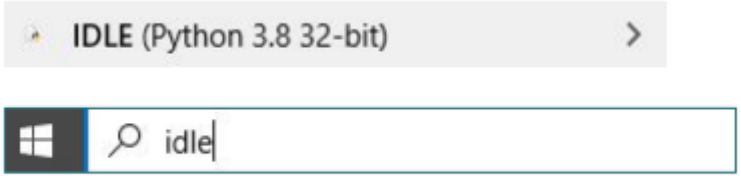
Computational Thinking





https://medium.com/pythonforkids/lesson-1-computational-thinking-f02fa49ac82b
https://www.wcpss.net/domain/17003

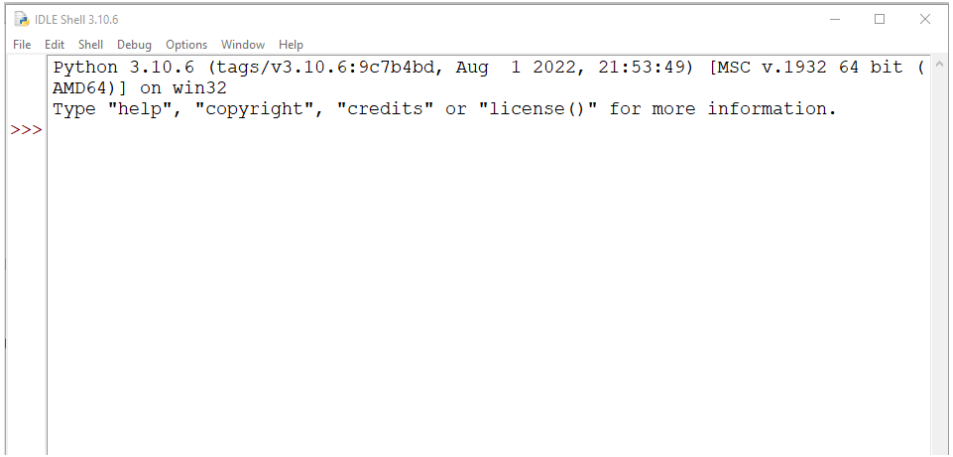
Python Editor



To do programming, we need editor (where we type our program commands). You can even use Notepad. ☺

There are many editors available, but in this course we will simply use two:

- IDLE (included already when you install Python)
- Visual Studio Code.



Installation can be found in SCELE page.

We will begin with using IDLE.
Type IDLE in your Windows Explorer search bar and a window with title "IDLE Shell 3.x.x" will open.

Using IDLE

The “>>>” is a Python prompt indicating that Python is ready for us to give it a command.

These commands are called statements. Putting a Python statement in the Python prompt allows us to do programming interactively; Python interprets the statement one at a time.

Turtle

Let us now do some programming with turtle!

Imagine a robotic turtle on coordinate (0,0).

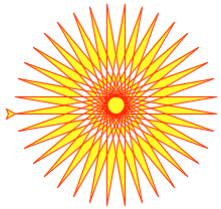
Writing `import turtle` allows us to instruct the turtle using command such as

```
turtle.forward(90)
turtle.left(10)
```

https://docs.python.org/3/library/turtle.html

Turtle star

Turtle can draw intricate shapes using programs that repeat simple moves.



```
from turtle import *
color('red', 'yellow')
begin_fill()
while True:
    forward(200)
    left(170)
    if abs(pos()) < 1:
        break
end_fill()
done()
```

Turtle methods

Let us now do some programming with turtle!

We will use methods that is provided in turtle's library.

Turtle methods

Turtle motion

Move and draw

- forward() | fd()
- backward() | bk() | back()
- right() | rt()
- left() | lt()
- goto() | setpos() | setposition(
- setx()
- sety()
- setheading() | seth()
- home()
- circle()
- dot()
- stamp()
- clearstamp()
- clearstamps()
- undo()
- speed()

Tell Turtle's state

- position() | pos()
- towards()
- xcor()
- ycor()
- heading()
- distance()

Pen control

Drawing state

- pendown() | pd() | down()
- penup() | pu() | up()
- pensize() | width()
- pen()
- isdown()

Color control

- color()
- pencolor()
- fillcolor()

Filling

- filling()
- begin_fill()
- end_fill()

More drawing control

- reset()
- clear()
- write()

Turtle state

Visibility

- showturtle() | st()
- hideturtle() | ht()
- isvisible()

Appearance

- shape()
- resizemode()
- shapesize() | turtlesize()
- shearfactor()
- settiltangle()
- tiltangle()
- tilt()
- shapetransform()
- get_shapepoly()

Using events

- onclick()
- onrelease()
- ondrag()

Special Turtle methods

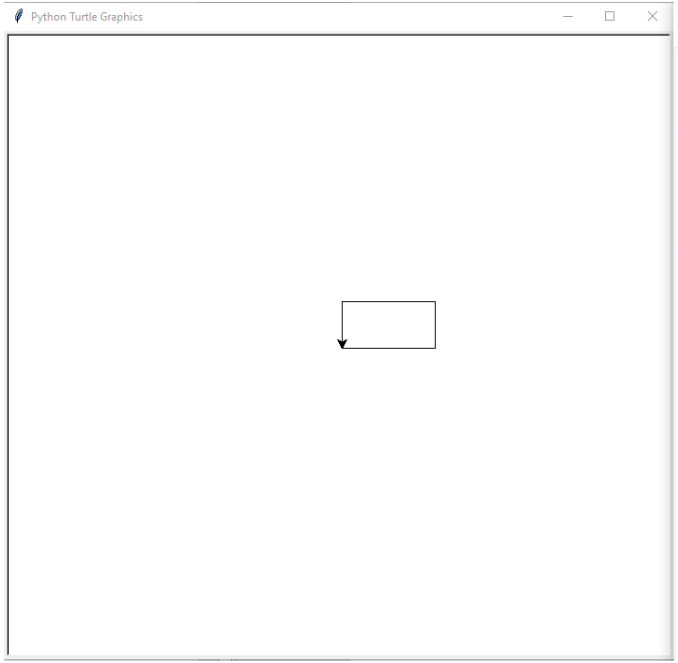
- begin_poly()
- end_poly()
- get_poly()
- clone()
- getturtle() | getpen()
- getscreen()
- setundobuffer()
- undobufferentries()

Turtle Example

Drawing a rectangle ☺

Turtle Example

Drawing a rectangle ☺



IDLE Shell 3.10.6

File Edit Shell Debug Options Window Help

```
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:53:49) [MSC v.1932 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import turtle
>>> turtle.getscreen()
<turtle._Screen object at 0x00000216C490C040>
>>> turtle.forward(100)
>>> turtle.left(90)
>>> turtle.forward(50)
>>> turtle.left(90)
>>> turtle.forward(100)
>>> turtle.left(90)
>>> turtle.forward(50)
>>>
```

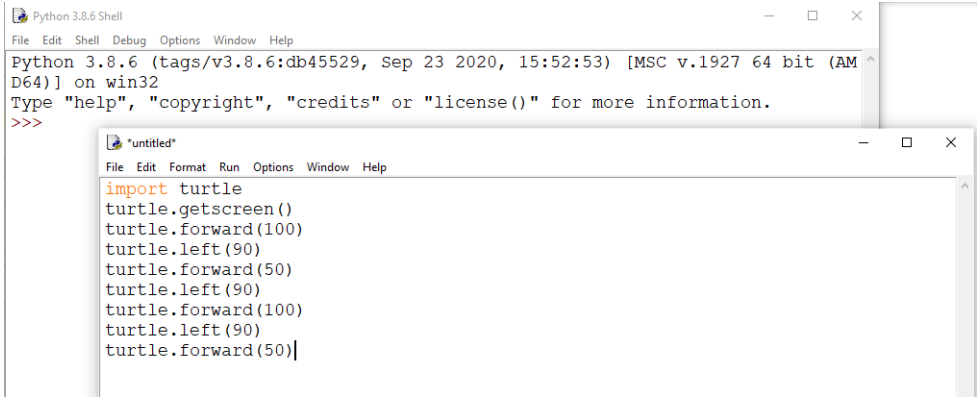
Creating a Python module

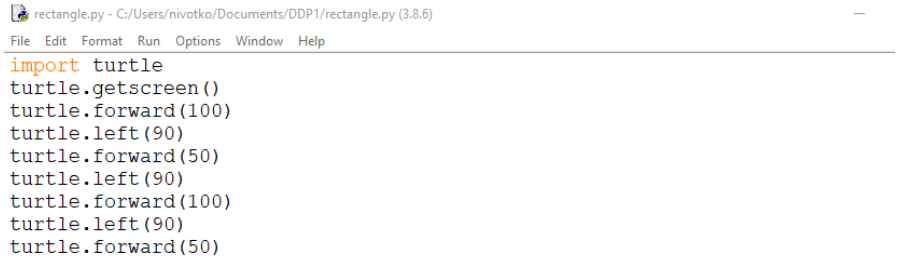
The commands we put into the Python shell to draw the rectangle using turtle can be written in one file called Python module / Python program.

On IDLE, click **File** > **New File**. A new window will occur and you can type the commands there.

The difference is this is now a program that is not immediately executed but rather collected and saved first as a file called Python module. Later, we can ask the computer to execute the commands (as if they are put in the shell one by one) in this module by just running the module once.

We can save the module by click **File > Save As** and write the name of your module. We choose here rectangle.py. The extension of a Python module is .py.





Creating a Python module

On the rectangle.py window, we can execute the commands by running the module. Click **Run > Run Module** or simply press the shortcut **F5**.

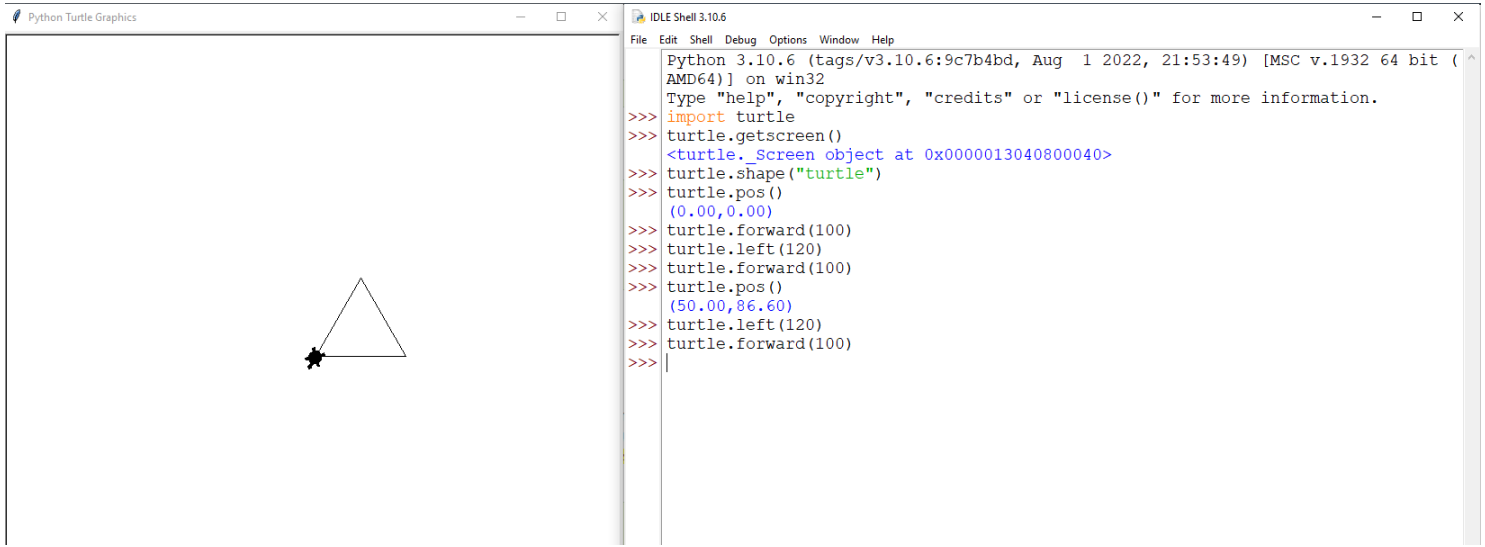
It will produce the same result as if you type the commands in the module one by one in the Python.

Turtle Example

Drawing an equilateral triangle ☺

Turtle Example

Drawing an equilateral triangle ☺



Turtle Example

We can again save the commands in a Python module. We name the module **triangle.py**.

triangle.py - C:/Users/nivotko/Documents/DDP1/triangle.py (3.8.6)

File Edit Format Run Options Window Help

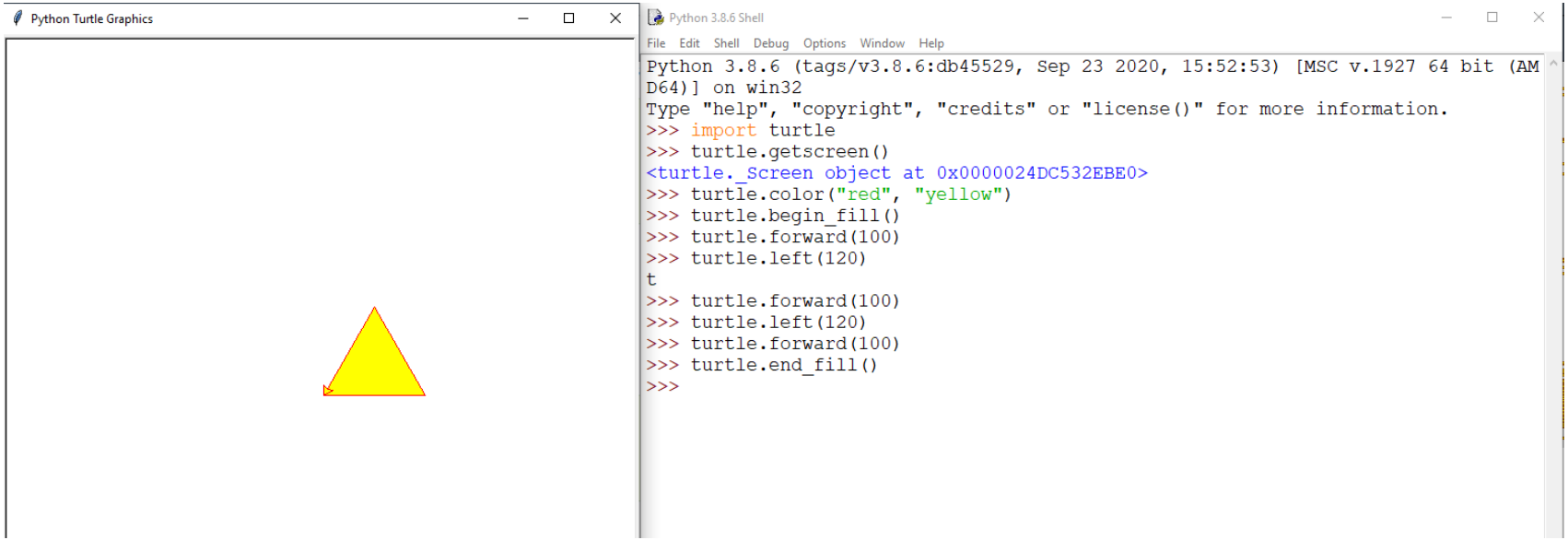
```
import turtle
turtle.getscreen()
turtle.shape("turtle")
turtle.pos()
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.pos()
turtle.left(120)
turtle.forward(100)
```

Turtle Example

Drawing an equilateral triangle ☺ with style

Turtle Example

Drawing an equilateral triangle ☺ with style



Turtle Example

Drawing an equilateral triangle ☺ with style

trianglestyle.py - C:/Users/nivotko/Documents/DDP1/trianglestyle.py (3.8.6)

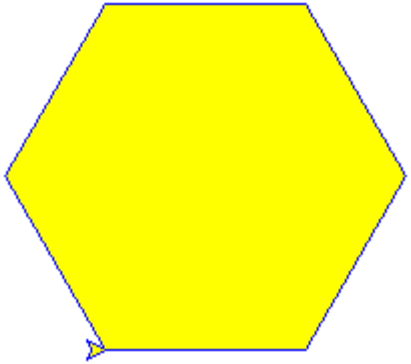
File Edit Format Run Options Window Help

```
import turtle
turtle.getscreen()
turtle.color("red", "yellow")
turtle.begin_fill()
turtle.pos()
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.pos()
turtle.left(120)
turtle.forward(100)
turtle.end_fill()
```

Exercise 1

Create a Python module .py that asks the turtle to draw a regular hexagon (all with the same side lengths and same angles) with blue border and yellow filling.

Question. What is the size of an angle of a regular hexagon?



Turtle Example

The following code creates 2 hexagons.

Try to understand what the following methods do in the turtle library documentation:

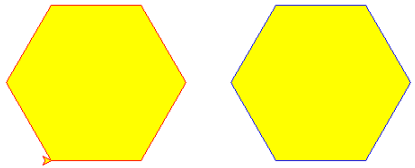
- .penup(),
- .pendown(),
- .goto()

twohexagons.py - C:/Users/nivotko/Documents/DDP1/twohe

File Edit Format Run Options Window Help

```
import turtle
turtle.getscreen()
turtle.color("blue", "yellow")
turtle.begin_fill()
turtle.forward(100)
turtle.left(60)
turtle.forward(100)
turtle.left(60)
turtle.forward(100)
turtle.left(60)
turtle.forward(100)
turtle.left(60)
turtle.forward(100)
turtle.left(60)
turtle.forward(100)
turtle.left(60)
turtle.end_fill()

turtle.penup()
turtle.goto(-250,0)
turtle.pendown()

turtle.color("red", "yellow")
turtle.begin_fill()
turtle.forward(100)
turtle.left(60)
turtle.forward(100)
turtle.left(60)
turtle.forward(100)
turtle.left(60)
turtle.forward(100)
turtle.left(60)
turtle.forward(100)
turtle.left(60)
turtle.forward(100)
turtle.left(60)
turtle.end_fill()
```

Python Turtle Graphics



Comment

When writing a program, you can add **comment**. In Python, comment started with #.

It is usually written on its own line or after a statement on the same line.

When executed comments will be ignored! You can write human message in the comment, Python does not care what it says.

Usually, comment is to explain your code.

star.py - C:/Users/nivotko/Documents/DDP1/star.py (3.8.6)

File Edit Format Run Options Window Help

```
import turtle
turtle.getscreen()
turtle.left(72) #why  72?
turtle.forward(100)
turtle.right(144) #why 144?
turtle.forward(100)
turtle.right(144)
turtle.forward(100)
turtle.right(144)
turtle.forward(100)
turtle.right(144)
turtle.forward(100)
turtle.right(144)
#it is a star!
```

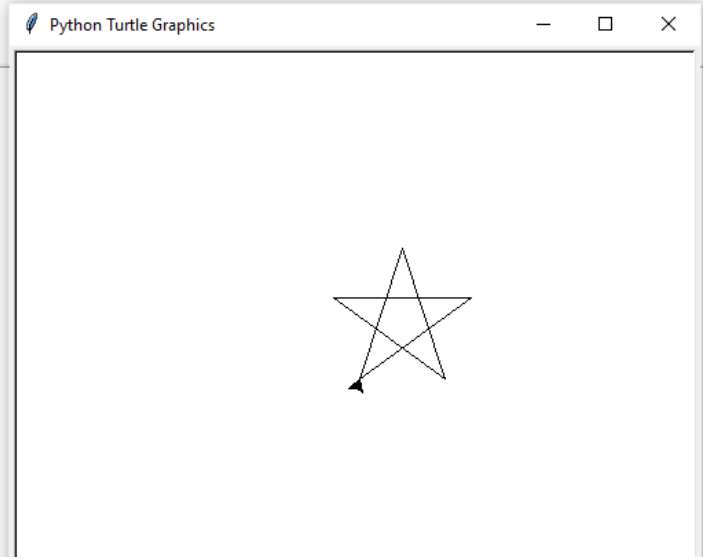
Comment

Python will ignore the comments when star.py module is executed.

star.py - C:/Users/nivotko/Documents/DDP1/star.py (3.8.6)

File Edit Format Run Options Window Help

```
import turtle
turtle.getscreen()
turtle.left(72) #why  72?
turtle.forward(100)
turtle.right(144) #why 144?
turtle.forward(100)
turtle.right(144)
turtle.forward(100)
turtle.right(144)
turtle.forward(100)
turtle.right(144)
turtle.forward(100)
turtle.right(144)
#it is a star!
```



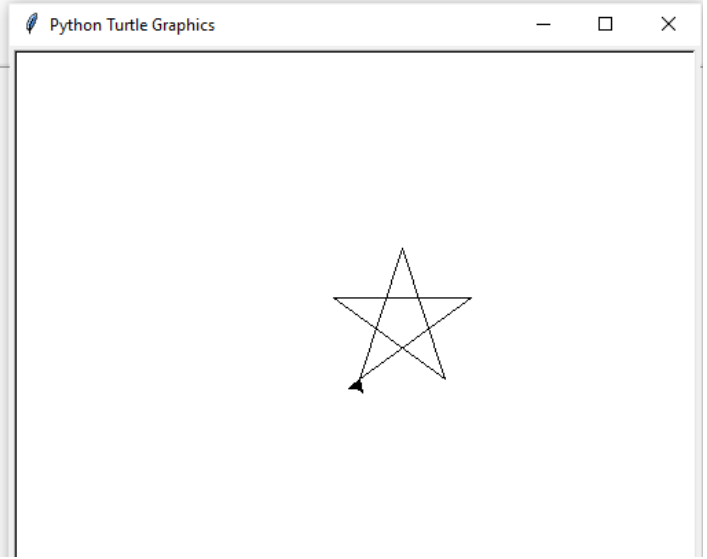
Comment

Python will ignore the comments when star.py module is executed.

star.py - C:/Users/nivotko/Documents/DDP1/star.py (3.8.6)

File Edit Format Run Options Window Help

```
import turtle
turtle.getscreen()
turtle.left(72) #why  72?
turtle.forward(100)
turtle.right(144) #why 144?
turtle.forward(100)
turtle.right(144)
turtle.forward(100)
turtle.right(144)
turtle.forward(100)
turtle.right(144)
turtle.forward(100)
turtle.right(144)
#it is a star!
```



Exercise 2

Consider the following triangle.py code again but with some of the statement commented.
Without running the module, predict what the turtle will draw!

[Imagine you are the turtle ☺; ignore the comments!]

triangle.py - C:\Users\nivotko\Documents\DDP1\triangle.py (3.8.6)

File Edit Format Run Options Window Help

```
import turtle
turtle.getscreen()
turtle.pos()
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
#turtle.left(120)
turtle.forward(100)
```

Exercise 3

Write a Python module that draws David star but composed of blue triangle and red triangle.

Q&A Session