Course Introduction

1DV501/1DT901: Introduction to programming

Jonas Lundberg, office B3024

Jonas.Lundberg@lnu.se

The slides are available in Moodle

August 28, 2022

Course Information

- 1DV501 (or 1DT901): Introduction to Programming (7.5 credits)
- A beginners course in Python programming
- **Course Manager: Tobias Andersson-Gidlund (`tobias.andersson@lnu.se`)**
- Main Teacher: Jonas Lundberg (`jonas.lundberg@lnu.se`)
- Involved teachers: Ola Flygt, Tobias Ohlsson, Tobias Andersson-Gidlund + 10 teaching assistants (TAs).
- Literature: Fundamentals of Python Programming,
  Richard L. Halterman, Draft March 29, 2019
  Available as pdf in Moodle.
- Moodle – a web based course information system
  - News
  - Lecture slides
  - Reading instructions
  - Assignments
  - Assignment submission system
  - Chat forum (Slack)
- Moodle is accessed from `http://mymoodle.lnu.se`
  (Moodle requires a Lnu student account)
- **Important:** Mail Tobias your Name + Swedish ID number + LnU username if you can't access Moodle.

Agenda (Today ...)

- Course information
- Computer systems
- The Python programming language
- Getting started with Python

Important
Please register yourself using the Lnu online system reachable from MyMoodle. Contact Tobias (tobias.andersson@lnu.se) and provide name, Swedish ID, Lnu username, if you run into problems.

Reading Instructions: 1.1-1.3 in book by R.L. Halterman

Course Setup

Notice: The course is given in 2 versions: In English and in Swedish. Also, we have a group of students in Kalmar $\Rightarrow$ **make sure to follow correct instructions/schedules**

- 13 lectures, 2h/lecture
  - In English: Monday and Thursdays
  - In Swedish: Monday and Wednesday
- Tutoring sessions: 2 meetings each week
- No written exam!
- Pass requires: Pass all assignments + Pass the mini-project + Pass Python Test
- 3 opportunities to pass assignments and Python Test (October, December, January)
- Final grade: Mini-project(40%) + Assignments(60%)
- Assignments 1-3 and the Mini-project are graded using ECTS grades: A-F (A = Brilliant!, ..., E = Sufficient to Pass, F = Fail)

Important: Some computer science courses starting after Christmas requires that you have completed this course.

Växjö, Kalmar, and Distance

- Student groups: Växjö (≈300), Kalmar (≈20), and distance (≈5)
- Växjö students are further divided into a Swedish and English group
- Kalmar/distance are both Swedish groups

English Group

- All lectures and tutoring sessions are given at campus Växjö
- No streaming, no recordings

Swedish Group

- ≈75% of lectures at campus Växjö, ≈25% at campus Kalmar
- Tutoring sessions at both campuses separately.
- Live streaming using Zoom, no recordings
- Distance (Physics) students will have to rely on the streams and online tutoring sessions.

Tutoring Sessions

- We have divided all students into separate tutoring groups
- Each such group has a separate course schedule
  $\Rightarrow$ read the time-plan carefully.
- **Attending the tutoring sessions is very important**
- Assignments 1 and 2 will be graded during the sessions $\Rightarrow$ you must show up
  - Attending meetings $\Rightarrow$ 80% will pass the course in November
  - Work at home $\Rightarrow$ 50% will pass the course in November

Identifying Your Tutoring Group

- First year students belonging to IT- and Math-programs have their own separate tutoring groups.
- Other Växjö or Kalmar students are assigned a tutoring group. See document entitled *Identifying Your Tutoring Group*
- Distance students from the Physics program has no campus sessions, only distance sessions.

Please contact Tobias (tobias.andersson@lnu.se) if you are not assigned a tutoring group.

Tutoring Groups

Each study program (roughly) forms a tutoring session group with an individual time schedule. Certain programs are further split in a Swedish and an English version.

- Computer Engineering, **TGI1D**, Tutor: Jonas Lundberg
- Electrical Engineering, **TGI1E**, Tutor: Tobias Andersson-Gidlund
- Software Engineering, **TGI1V**, Tutor: Tobias Ohlsson
- Master of Science: Software Engineering, **CIDMV**, Tutor: Jonas Lundberg
- Master of Science: Engineering Mathematics, **CTMAT**, Tutor: Jonas Lundberg
- Network Security (English), **NGDNS-en**, Tutor: Ola Flygt
- Network Security (Swedish), **NGDNS-sv**, Tutor: Ola Flygt
- Software Technology (English), **NGDPV-en**, Tutor: Tobias Andersson-Gidlund
- Software Technology (Swedish), **NGDPV-sv**, Tutor: Tobias Andersson-Gidlund
- ... and a few more!

Use "Identify your tutoring group" in Moodle to identify your group.

Moodle contains detailed time schedules for each group.

The Python Test

- A practical programming test
- 3 exercises, 2 hours
- Supposed to be simple for everyone having handled Assignments 1-2
- One example of an old Python test is available in Moodle
- Preliminary date: Friday October 7

- Part of course examination $\Rightarrow$ you must pass the test to pass the course
- You sign up for the test in Moodle. Registration will open up about two weeks before the test.

Assignment Rules

- **The three assignments are individual. Each student present their own set of solutions**
- Each exercise is tagged as (G) or (VG).
  - G: Exercise that you need to present at the tutoring sessions. **Mandatory to pass the course**.
  - VG: Exercises that should be submitted in Moodle. Mandatory for students aiming for grades A or B.
- Pass requires correct solutions on all mandatory (G) exercises.
- **You must present your assignments before/around the deadline**
  - Not presenting Assignment 1 solutions $\Rightarrow$ not active $\Rightarrow$ you will be removed from the course
- Deadline exceptions can be given if you, within a reasonable time *before* the actual deadline, contact your assignment tutor and give a reasonable explanation.
- Exchange ideas, not solutions. **Plagiarism** $\Rightarrow$ students who copy (parts of the) programs from colleagues or elsewhere, or get others (friends, relatives, hired skilled persons) to complete their assignments, fail the assignment automatically.
- **Read Assignment Rules in Moodle carefully!**

Slack and Moodle

The Moodle website for 1DV501/1DT901 is our main source for publishing information ⇒ one-way communication

- Assignments
- Lecture slides
- Pinned news (e.g. news about an upcoming Python test)
- Various information, rules, and installation instructions

Slack is our chat forum ⇒ two-way communication

- **Slack is our main channel of communication in this course**
- Assignment or course related questions/discussions
- Visited daily by teachers/assistants ⇒ expect a fast response
- Students are encouraged to help each other
  (but please don't post assignment solutions)
- Use the function "Reply in thread". Do not start a new thread when replying to a post

Do not ask course related questions via email. Ask them in Slack. Only personal questions in e-mail.

Online tutoring using Slack

We are using Slack for two things:

1. As a chat forum where students, teaching assistants, and teachers can discuss topics related to the course and assignments.
2. As a tool for time scheduled individual online supervision for **distance students**

We use Slack for time scheduled online tutoring for distance students

- The start of a typical online activity starts by us posting a message in Slack that looks like this:

  ```
  Physics distance students: We will be available here at Slack today
  from 10.15 till 12.00 to answer your questions about Assignment 1.
  Enter your name below in the thread and we will call you.
  ```

- You sign up (if you need help) and we call you up (using Slack) one at the time.
- Advantage: You can share your screen and we can provide detailed answers.
- Please keep it in one thread

Questions?

Any questions about the course setup?

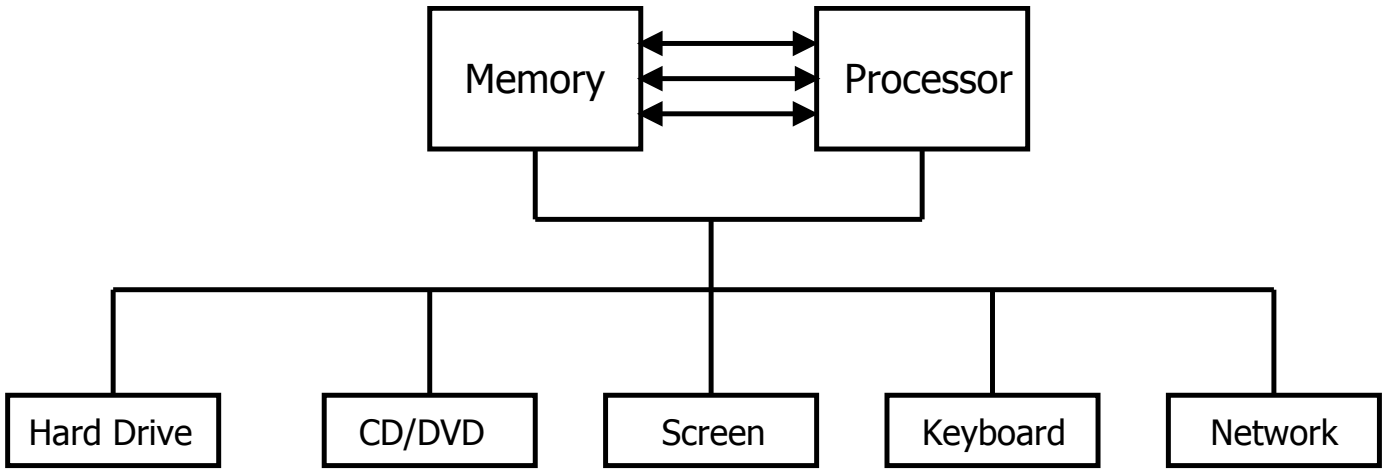
Computer Systems

Most *Computer Systems* consists of

1. **Hardware**
   - Physical parts you can "touch"
   - Processor
   - Primary memory
   - Hard drive
   - Memory card
   - Buses (cables inside computer)
   - Keyboard
   - Screen
2. **Software**
   - Software = programs and their data
   - Operating System
   - Other programs (applications)

- Hardware and software are useless without each other
- This course is about **software development**
- We present a minimum of hardware

Hardware



- **(Micro)Processor** (also *CPU = Central Processing Unit*)
  A chip performing simple computations very fast
- **(Primary) Memory** (also *Main Memory or RAM*)
  Memory where program/data are stored while processed. Small with fast access.
- **Hard drive**
  Permanent storage for programs/data not currently used. Large and slow.
- The hard drive is a **secondary memory** where information can be stored permanently $\Rightarrow$ can survive without electricity
  Other types of secondary memory: SSD, CD, DVD, USB Memory stick, etc

Execution = Running a program

- Execution ⇒ read a program from hard drive to primary memory and start to process its instructions.
- The processor consists of
  - a **arithmetic/logic unit** processing instructions
  - a **register**: very small/fast memory where intermediate data/results are stored
- An executable program = a sequence of instructions in **binary form**

```
C = A + B                                  Binary form
**********                                 ***********
1. Read value of A to R(1)                 1. 01010101 10110010 11000110 01001100
2. Read value of B to R(2)                 2. 00100010 11100111 00101010 10001111
3. Compute R1+R2 and save in R(3)          3. 10010010 11100100 11001001 11010100
4. Assign C value in R(3)                  4. 10101011 00100101 11000011 10010001

Where R(N) = Register number N
```

- Binary form ⇒ instructions are sequences of 0 and 1 that only (almost) a computer can understand.
- Different types of processors have different set of instructions

Digital information

- Digital comes from the word *digit* = 0,1,2,3,4,5,6,7,8,9
- All information on a computer are stored as integers
- In addition to instructions and integers, it also holds for:
  - floats (e.g. decimals like 2.56 or -0.0045)
  - text
  - sound
  - pictures
  - videos
- Each picture pixel is stored as three integers (red,green,blue).
- Text: Each character is stored as an integer.
- Computers store all integers in **binary form**.
  For example, 14 is stored as 1110.

Number Systems – Decimal and Binary

Decimal Numbers

- Ordinary (decimal) numbers have the *base* 10.
- We use the digits 0-9 to describe a number
- $234 = 2 * 100 + 3 * 10 + 4 * 1 = 2 * 10^2 + 3 * 10^1 + 4 * 10^0$
- Each digit (2,3 or 4) has a specific *weight*.
- Decimal $\Rightarrow$ we use weights $..., 10000, 1000, 100, 10, 1$
- $\Rightarrow$ the n:th digit from the end has weight 10^{n-1}

Binary Numbers

- The binary numbers have the *base* 2.
- We use only the digits 0 an 1 to describe a number.
- $1110_2 = 1*2^3+1*2^2+1*2^1+0*2^0 = 1*8+1*4+1*2+0*1 = 14_{10}$
- That is, 1110 in base 2 equals 14 in base 10.
- Binary weights: $\ldots, 16, 8, 4, 2, 1$
- $\Rightarrow$ the n:th digit from the end has weight 2^{n-1}

Binary Numbers

- A binary digit (0 or 1) is called a *bit*
- 8 bits forms a *byte*
- Using a byte, we can represent 256 integers between 0 (0000000) and 255 (11111111)
- In general: Using n bits we can represent 2^n integers between 0 and $2^n - 1$
- Negative numbers: The first bit gives the sign ($1 = +$, $0 = -$), the remaining bits give the size.

Why binary numbers?

- Hardware rules! Much easier, faster, and safer to transport and store information in binary format
- In cables (transport): Voltage on $= 1$, voltage off $= 0$
- In memories (storage): Magnetization up $= 1$, magnetization down $= 0$
- Memory: 1 GB (GigaByte $= 2^{30}$ bytes, more than a billion bytes)
- Hard drives: 1 TB (TeraByte $= 2^{40}$ bytes, more than 1000 billion bytes)
- Furthermore : 1KB $=$ 1 KiloByte $= 2^{10} = 1024$ bytes, 1MB $=$ 1 MegaByte $= 2^{20} = 1048576$ bytes,

Example: Text Encoding

- Each character (alphabet + others) is represented by an integer
- In most cases we use 1 byte/character $\Rightarrow$ 255 possible characters
- There are a few different encoding standards. Most common is ASCII.

```
Hello Sweden!

H=72, e=101, l=108, l=108, o=111, " "=32,

S=83, w=119, e=101, d=100, e=101, n=110, !=33. linebreak=10
```

- Also non-visible characters like space, tab and line break (Enter) have an integer encoding.
- A text file is a long sequence of bytes where each byte represents a character.
- A text editor (like Wordpad) converts between characters and integers whenever a file is read/saved.

Software

The computer software is divided in two parts:

1. **Operating System:** The program that controls the computer
   - Starts the computer
   - Distributes processing power between programs
   - Provides a GUI (Graphical User Interface) for the user
     $\Rightarrow$ We can control the computer by "mouse clicks"
   - A software layer between hardware and other programs.
     $\Rightarrow$ gives other programs controlled access to hardware
     $\Rightarrow$ protects hardware
   - Common OS: Windows, Mac OS, Unix, Linux
2. **Applications:** All other programs
   - Word processing
   - Games
   - Software controlling a car engine
   - MP3 players, anti-virus software, email client, ...
   - Internet Explorer, Visual Studio Code, Excel, ...
   - Solves a specific task by utilizing the hardware accessed through the OS.

Programming Levels

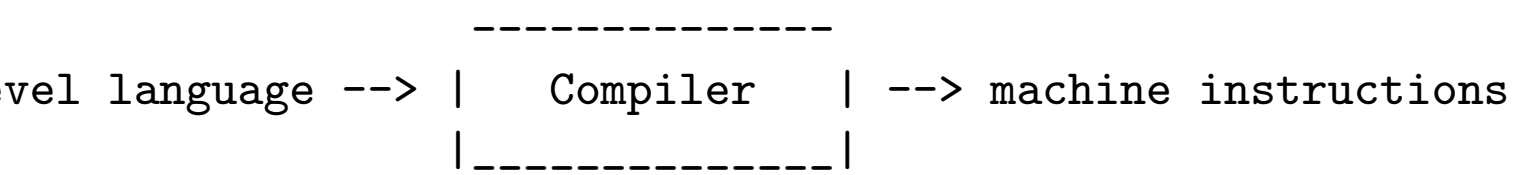
- The processor only understands binary instructions
  (also called *machine code* or *machine instructions*)
- Humans have a hard time understanding binary instructions
- 1950s: Assembler code

```
Assembler (C = A + B)      Binary form
=========                  ==========
1. Load A R1               1. 11100111 00101010 10001111
2. Load B R2               2. 11100111 11001001 11010100
3. Add  R1 R2 R3           3. 10101011 10001111 11010100 00100010
4. Save R3 C               4. 00100101 00100010 10010001
```

- Assembler: Give each binary instruction a name.
- Simple to translate to binary form (one bit assembler $\Rightarrow$ one instruction)
- Assembler is still in use to control hardware

High-level Languages

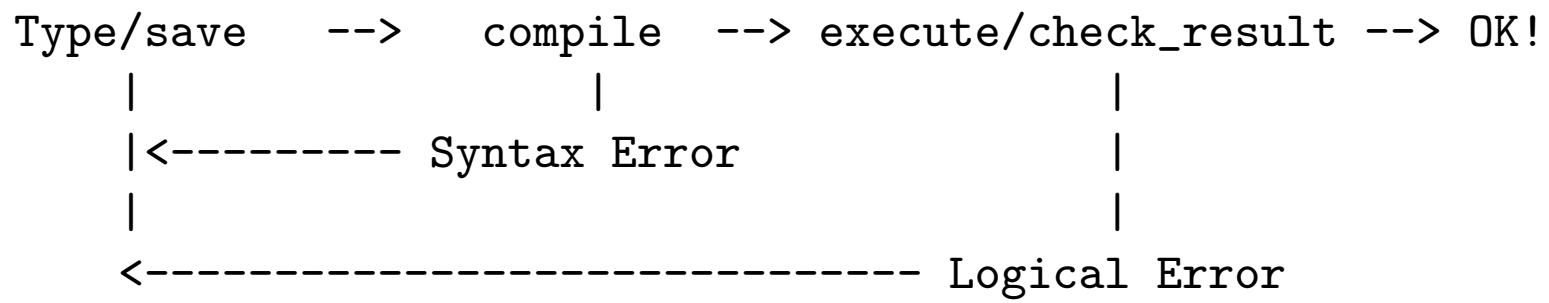
- Assembler is easier than binaries, but still difficult/slow for humans
- The programmer must know everything about the hardware
- 1960s: **High-level languages were introduced where each instruction represents many machine instructions.**
- Common HL languages: C, Ada, Pascal, C++, C#, Java and **Python**
- Converted to machine instructions by a **compiler**
  $\Rightarrow$ a program translating high-level language to machine instructions



- A compiler is specific for: one high-level language, and one type of machine instructions
- The compiler knows about machine instructions $\Rightarrow$ we don't have to.
- The compiler start by checking if the input program is correct
  $\Rightarrow$ `Error: Missing semi-colon on line 234`

High-Level Programming

- A **programming language** has a given **syntax**.
- Syntax are rules about:
  - which words and symbols that can be used in a program
  - how to combine words/symbols into correct **statements**
- A program is in principle a sequence of statements
- Programming (in theory)
  1. Type (edit) the program as ordinary text
  2. Save the program in a file
  3. Compile the program ⇒ generate machine code
  4. Execute the program
- Programming in practice



Why learn how to program?

- Computer ⇒ an extremely powerful problem solving tool
- Programming ⇒ take control over the computer
- Non-programmers ⇒ must rely upon other people's program
- Little programming (less than 1000 lines of code)
  - short scripts for web pages
  - simple device drivers (to control machines)
  - rather complex scientific computations
  - understand what programming is all about
- Much programming (millions of lines of code)
  - develop commercial software, for example
  - games
  - administrative tools program
  - missile system
  - search engine
- **Notice:** Develop robust and user-friendly programs for non-experts require a lot of programming.

No programming language is the best but ...

No language is the best but some languages are in higher demands than others

Ranking	Language	Jobs
1	Java	29.000
2	Javascript	24.000
3	Python	19.000
4	C#	18.000
5	C/C++	17.000
6	PHP	7.000
7	Swift	1.800
8	Go	1.700
9	R	1.500

Jobs: the number of job postings listed on job search site Indeed in 2020

Q: And which language should you learn then?
A: Learn to program in any language, to understand important principles, and to solve problems!

Source: `www.northeastern.edu/graduate/blog/most-popular-programming-languages/`

A 10 minute break!

zzzzzzzzzzzzz ...

The Python Programming Language

- Created by Guido van Rossum and first released in 1991
- Python's design philosophy emphasizes code readability
- Python 2.0 was released in 2000, and Python 3.0 in 2008
- Current version is Python 3.10
- Python is run by the Python Software Foundation (at `python.org`)

- Python is a high-level, interpreted and general-purpose language ...
- ... that focuses on code readability.
- The syntax rules in Python helps the programmers to do coding in fewer steps as compared to Java or C++.
- Python is less difficult to learn compared to Java, C#, ...
  $\Rightarrow$ Python is suitable for a beginners course in programming

- **My opinion** (and many agree)
- Use Python for small programs and prototypes
- Use Java, C#, ... for larger software systems

Java vs Python

Programs in Java and Python that print `Hello` on the screen

Hello.java

```
public class Hello {

    public static void main(String[] args) {
        System.out.println("Hello");
    }
}
```

Each Java program requires a class declaration (`class Hello`) and a main method declaration to work. Printing is done using the `System.out` stream.

hello.py

```
print("Hello")
```

No additional structure is needed. Just insert statements that you would like to execute. In this case `print("Hello")`.

The structure of a (simple) Python program

We use a text editor and writes the following code in a file named `simple.py`

```
# Add two integers and print the result
a = 10
b = 15
sum = a + b  # Compute sum of a and b
print(sum)   # Print result
```

- All Python programs must be saved in a file named "something" `.py`
- `#` $\Rightarrow$ a comment reaching to the end of the line
- `a, b, sum` are variables
- `print(...)` is a library function used to print results to the screen

Just an example, we will start to learn Python in Lecture 2.

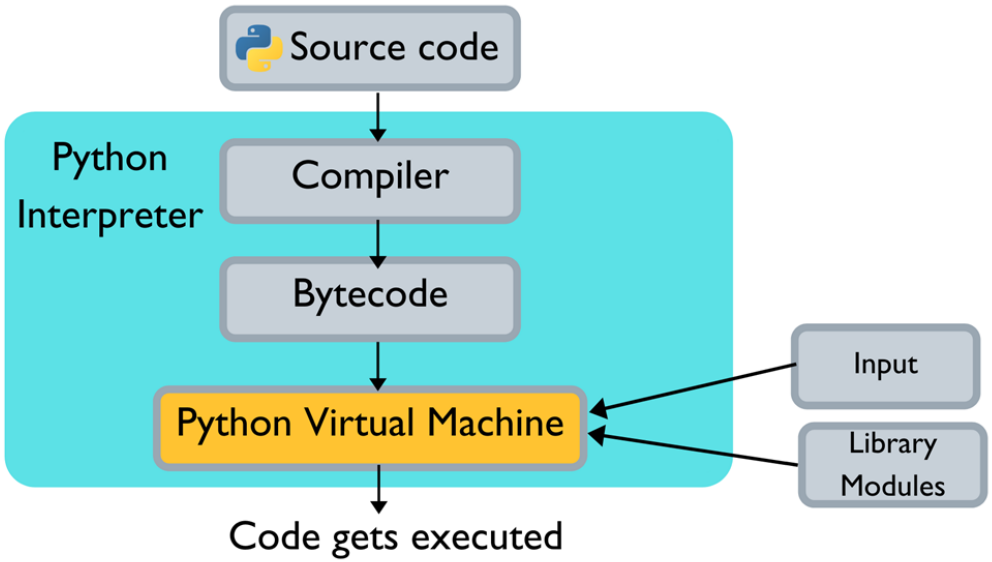
Syntax and Semantics

- The **syntax** rules define how we can combine keywords, identifiers, and other language constructs into a correct program.
- Syntax errors are discovered by the compiler (or VS Code).

```
Find 5 Errors!
==============
% Add two integers and print the result
a = 10
b + 5 = 15
sum = A + b
Print(sum          # Print result
```

- The language **semantics** defines the functionality of all language constructs.
- Example: The semantics of `print("Hello")` is to print `Hello` on the screen.
- **Notice:** A syntactically correct program is not necessary logically correct.
- A computer does exactly what we *tell it* to do,
  not necessary what we *want it* to do.
- The computer (almost) never makes a mistake.
  It is we (programmers) who give it incorrect instructions.

Python Execution



Executing `hello.py` ⇒ Start the Python Interpreter and provide `hello.py` as input.
Inside the interpreter:

1. The compiler (a program) checks if the input program is a correct Python program (and terminates with an error if something is wrong)
2. The compiler translates the source code to an internal format called bytecode
3. The Python Virtual Machine (PVM) takes the bytecode as input and executes the program
4. The PVM knows about library modules and can read input from the keyboard (or from files)

Getting Started - Recommended Approach

1. **Buy a laptop if you don't have one!**
   ($\geq$8GB memory, no Chromebook, no tablet/Ipad, no MS Surface Go)
2. Install Python on your laptop using Anaconda (Distribution)

   `www.anaconda.com/products/distribution`

   Verify that it works by writing `python` in your Terminal/Console prompt.

```
jlnmsi % python
Python 3.8.3 (default, Jul  2 2020, 11:26:31)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()  # terminate python
```

3. Install Visual Studio Code (VSC) on your laptop

   `code.visualstudio.com`

4. Install the Microsoft Python Extension on VSC

   `marketplace.visualstudio.com/items?itemName=ms-python.python`

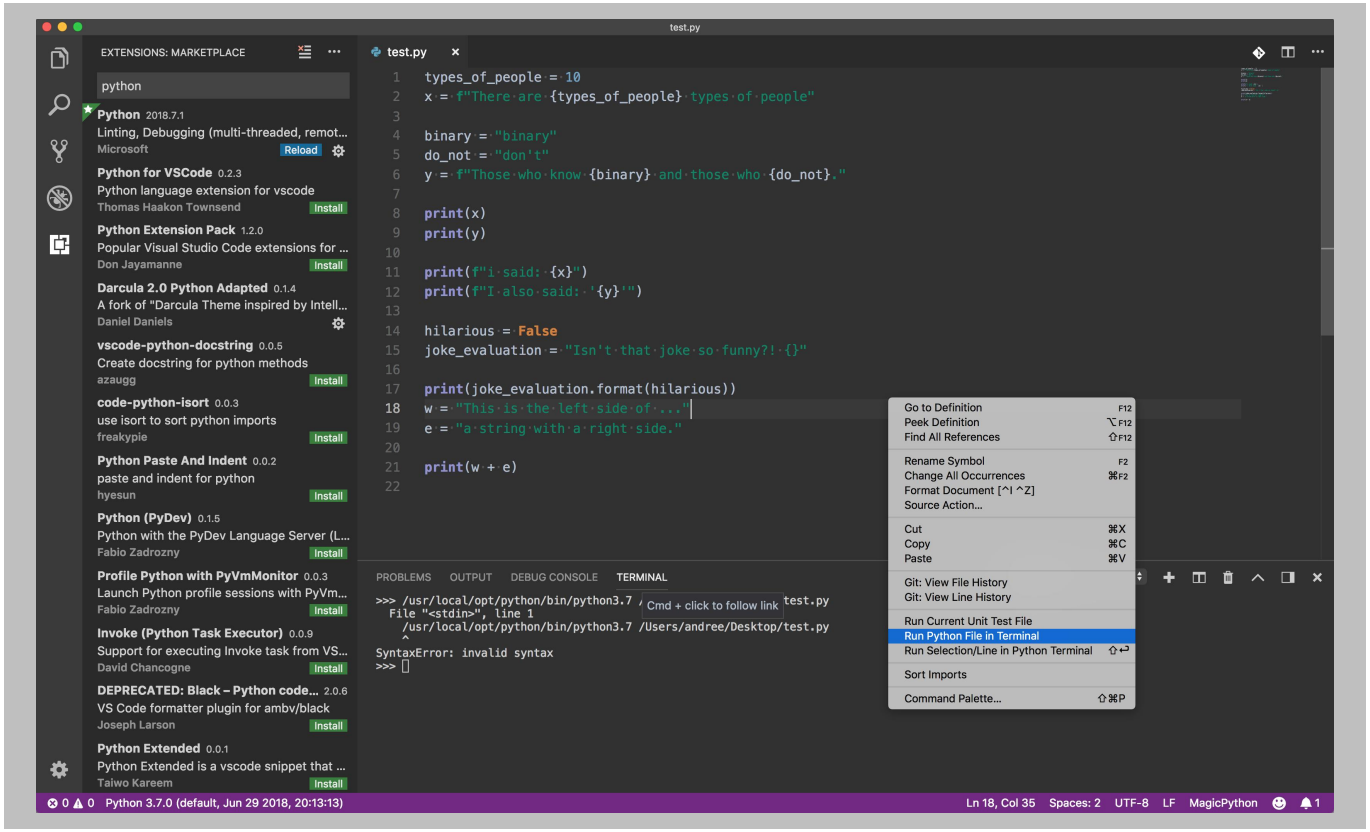
5. Setup a folder structure for your Python programs (See A1 instructions)
6. Create a file `hello.py` and try to run it in VSC

Anaconda

- Anaconda is a Python distribution ⇒ it contains all you need (and much more) to run a Python program ⇒ a Python interpreter and a large number of library modules (and much more)
- Windows users, make sure to use the option "Add Anaconda to my PATH environmental variable" during the installation process.
- Advantage: Easy to download and install
- Advantage: Everything we need in this course
- Disadvantage: Much more than we need in this course
- Verify that it works by writing `python` in your Terminal/Console prompt.

```
jlnmsi % python
Python 3.8.3 (default, Jul  2 2020, 11:26:31)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()  # terminate python
```

Visual Studio Code + Python Extension



- Visual Studio Code is an Integrated Development Environment (IDE) ⇒ a software tool to simplify programming ⇒ provides support for program execution and organizing all program files in a project (and much much more)
- We will use the Python Extension, VSC supports many other programming languages (e.g. Java).

Python Programming - Three approaches

1. Edit a Python file `X.py`, save file, and execute. Use a tool like VSC to simplify the process
2. Write your program in the Terminal/Console window
3. Use Jupyter Notebook $\Rightarrow$ a web-based interactive computational environment for creating Python programs

Notice

- Edit, save, execute (Approach 1 above) is the traditional approach used in most companies and similar to the approach used in other languages
- The Console/Terminal approach only works for very small programs. Used mainly to test and try various commands and modules.
- The Jupyter approach is great for small to medium projects

We will stick to the traditional approach, approach 1 above. It is the default and it is the way to do it in other programming languages.

The textbook by Halterman often uses the Console/Terminal approach to show simple examples. Get used to reading it but we always use the traditional approach in lectures and assignments.

Live Tool Demo

- Check Python in Terminal/Console window
- Simple Python in Terminal/Console window
- Create folders and files in VS Code
- `Hello.py` in VS Code
- Check Python in VS Code
- Change font size and color theme in VS Code

Follow instructions in Assignment 1 for how to setup Python and VS Code.

The Python Help Desk

Students running into problems installing and handling various applications used in this course are advised to contact the **Python Help Desk**.

Two teaching assistants will provide help for the following applications:

- Python (via Anaconda)
- Visual Studio Code
- The Python Extension for Visual Studio Code
- Lints in Visual Studio Code
- Debugging in Visual Studio Code
- Gitlab
- mathplotlib

The first two Python Help Desks will be available at Campus Växjö and Online (Slack) for distance students the following dates.

- Tuesday Aug 30, 12.15-14.00, D1173
- Friday Sep 2, 15.15-17.00, D1173

See time schedule for details and more Help Desks.

Upcoming Activities

- Next Lecture
  - English: Thursday, September 1, at 10.15 (Växjö)
  - Swedish: Wednesday, August 31, at 10.15 (Växjö)
- First tutoring session
  - Each program has their own schedule
  - $\Rightarrow$ Look it up in Moodle
  - Simple Python programs
  - The VSC Development Environment
  - Bring your laptop (if you have one)

- **Before Next Activity**:
  - Try to install Anaconda at home
  - Try to install Visual Studio Code at home
  - Follow instructions (URLs) given in Assignment 1 (in Moodle)

  Also, play around with VS Code. For example, try to change the background color and the font size.