HIGH-LEVEL LANGUAGES (HLL)

- High-level languages
  - easy to read and understand since it is close to human language
  - write in a shorter time
  - easy to debug at the development stage
  - portable language: it can be used on different platforms
  - library: pre-written code, so it saves time because no need for writing again

```python
i = 0
N = int(input('How many numbers do you want to enter?:  '))
n_maximum = int(input('Insert the first number:  '))

while i < N-1:
  n=int(input('Insert a number:  '))

  if n > n_maximum:
    n_maximum = n;

  i += 1
  print('The maximum value is: ', n_maximum)
```

LOW-LEVEL LANGUAGES (LLL)

- Low-level languages
  - a low-level languages is designed for the specific architecture and hardware
  - the programmer needs to understand those architecture and hardware well
  - some instruction work directly on CPU (doesn't need to translate to machine code)
  - relates assembly language and machine code

```
LDA     First
ADD     Second
STO     Sum
```

In order to understand this program the programmer needs to know that

- **LDA** means load the value of the variable into the accumulator
- **ADD** means add the value of another variable to the value stored in the accumulator
- **STO** means replace the value of the variable by the value stored in the accumulator.

```
 GNU nano 3.2                    hello.asm

section  .text
global   _start

_start:

    mov  edx, len
    mov  ecx, msg
    mov  ebx, 1
    mov  eax, 4
    int  0x80

    mov  eax, 1
    int  0x80

section  .data

msg  db  "Hello World!"
len  equ $ - msg
```

HIGH-LEVEL AND LOW-LEVEL LANGUAGES

Language	Advantages	Disadvantages
HLL	Easy to read, write and understand	Programs could be larger
	Quicker to write programs	Programs could take longer to execute
	easier and quicker to debug	
LLL	Program execute faster	Difficult to understand and learn
	Some instructions are understandable by CPUs	Take more time to write a program
	Doesn't take much memory	

ASSEMBLY LANGUAGES

- it is a low-level language
  - to make use of special hardware
  - to make use of special machine-dependent instructions
  - to write code that doesn't take up much space in primary memory
  - to write code that performs a task very quickly.

```
 GNU nano 3.2                    hello.asm

section  .text
global   _start

_start:

    mov  edx, len
    mov  ecx, msg
    mov  ebx, 1
    mov  eax, 4
    int  0x80

    mov  eax, 1
    int  0x80

section  .data

msg  db  "Hello World!"
len  equ $ - msg
```

```
LDA        First
ADD        Second
STO        Sum
```

In order to understand this program the programmer needs to know that

- LDA means load the value of the variable into the accumulator
- ADD means add the value of another variable to the value stored in the accumulator
- STO means replace the value of the variable by the value stored in the accumulator.

TRANSLATORS

- Compilers
  - It translates high-level language to machine code
  - After a program is compiled, executable file is created. Thus, the program can be used again and again without recompilation
  - Advantages
    - With executable file, no need a complier to run the program
    - No source code, user cannot edit or plagiarise
    - Complied program provides faster execution than interpreted program
    - Portable language

TRANSLATORS

- Interpreters
  - It translates statement by statement
  - If there is an error, it stops and shows the error code.
  - No translated program is generated for later used
  - Advantages
    - It is good for developer because it is easy to find errors
    - Portable language
  - Disadvantages
    - Slow execution because it translates code every time to run the program
    - No executable file created

TRANSLATORS

- Assembler
  - It translates Assembly to machine code
  - Assembly work faster than high level language because its code work directly on processor
  - Different type of computer has different assembly language
  - It is machine dependent, it is not portable language
  - The instruction consists two parts, opcode and operand
  - Opcode and operand sometimes are called mnemonic

TRANSLATORS

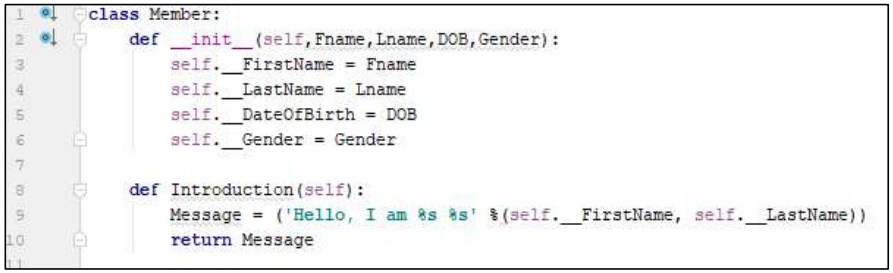
	Assembler	Compiler	Interpreter
Source program written in	assembly language	high-level language	high-level language
Machine dependent	yes	no	no
Object program generated	yes, stored on disk or in main memory	yes, stored on disk or in main memory	no, instructions are executed under the control of the interpreter
Each line of the source program generates	one machine code instruction, one to one translation	many machine code instructions, instruction explosion	many machine code instructions, instruction explosion

TRANSLATORS

Compiler	Interpreter	Assembler
Translates a high-level language program into machine code.	Executes a high-level language program one statement at a time.	Translates a low level assembly language program into machine code.
An executable file of machine code is produced.	No executable file of machine code is produced.	An executable file of machine code is produced.
One high-level language statement can be translated into several machine code instructions.	One high-level language program statement may require several machine code instructions to be executed.	One low-level language statement is usually translated into one machine code instruction.
Compiled programs are run without the compiler.	Interpreted programs cannot be run without the interpreter.	Assembled programs are used without the assembler.
A compiled program is usually distributed for general use.	An interpreter is often used when a program is being developed.	An assembled program is usually distributed for general use.

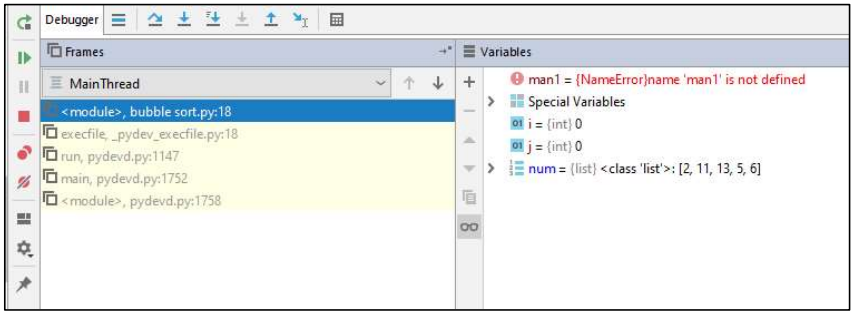
INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

- Integrated development environment (IDE)
  - A software that allow users to write programing code, debug and translate the code
- Source code editor
  - A program that helps programmers write and edit code
  - It has features:
    - Prettyprinting: it colours code/keyword/reserved word
    - Auto-complete: it guesses what to fill in the code and give suggestions
    - Automatic indentation: it indents block of code e.g., for…loop or if…then…else…
    - It highlights syntax errors



INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

- Translators (compilers and interpreters)
  - IDEs also come with translator
  - Previous topic
- A run-time environment with a debugger
  - Single stepping : run one statement at a time
  - Breakpoint : to stop the program running at specific line
  - Variable watch window : to monitor variables during the program running



QUESTION

Statement	Interpreter	Compiler
takes one statement at a time and executes it		
generates an error report at the end of translation of the whole program		
stops the translation process as soon as the first error is encountered		
slow speed of execution of program loops		
translates the entire program in one go		

QUESTION

- State why a compiler or an interpreter is needed when running a high-level program on a computer (1 mark)

- Give one benefit of writing a program in a high-level language.

- Give one benefit of writing a program in a low-level language.

QUESTION

Statement	true (✓)	false (✓)
Assembly language uses mnemonic codes.		
Assembly language programs do not need a translator to be executed.		
Assembly language is a low-level programming language.		
Assembly language is specific to the computer hardware.		
Assembly language is machine code.		
Assembly language is often used to create drivers for hardware.		