NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SC1003/SC5001/CE1103/CZ1103
Introduction to Computational Thinking and Programming:
Course Briefing

Who Am I?

Dr. Li Fang
College of Computing and Data Science (CCDS)

Office: N4-2B-42;
Phone: 6790-6106
Email: asfli@ntu.edu.sg
Consultation:
Appointment via email/Teams

Naming convention for the email subject: course ID-**

e.g. SC1003-****

slido

Please download and install the Slido app on all computers you use

Have you ever learned programming?

ⓘ Start presenting to display the poll results on this slide.

slido

Please download and install the Slido app on all computers you use

Have you ever learned computation thinking?

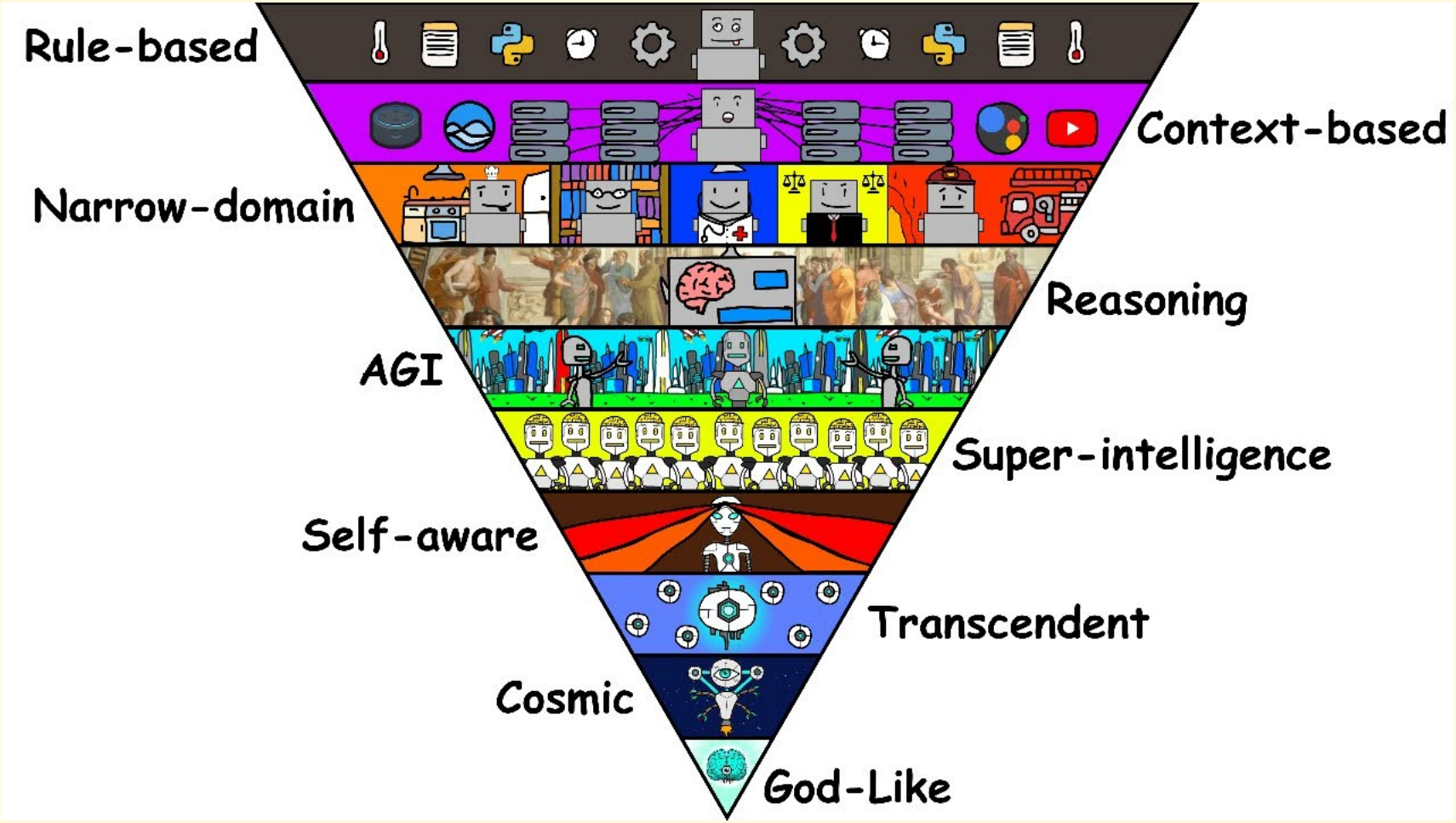
ⓘ Start presenting to display the poll results on this slide.

How important is computational thinking as a skill?

- Computational thinking is an essential skill in today's technology-driven world. It involves solving problems, designing systems, and understanding human behavior by drawing on concepts fundamental to computer science.
- Computational thinking is not just for computer scientists; it is a fundamental skill that enhances problem-solving, critical thinking, and the ability to tackle complex issues across various domains. Its importance in education, the workplace, and daily life makes it an essential skill for everyone.

Education: Prepare the students for the future

The 10
Stages of AI



https://www.youtube.com/watch?v=uQn3oi0SMbo

Key Aspects of Computational Thinking

- **Problem-Solving**:
  - **Decomposition**: Breaking down complex problems into manageable parts.
  - **Pattern Recognition**: Identifying similarities within problems to apply known solutions.
  - **Abstraction**: Focusing on the important information only, and ignoring irrelevant detail.
  - **Algorithms**: Developing step-by-step solutions to problems or rules to follow to solve problems.

slido

Please download and install the Slido app on all computers you use

Have you ever used Generative AI systems?

ⓘ Start presenting to display the poll results on this slide.

Since ChatGPT can code well, why do we still need to learn programming?

Even though AI like ChatGPT can code well, there are several important reasons why learning programming remains valuable:

1. Understanding Fundamentals

- **Core Concepts**: Understanding basic programming concepts such as algorithms, data structures, and computational thinking is essential. These are foundational skills that apply across various programming languages and technologies.
- **Problem Solving**: Programming teaches problem-solving skills, which are critical in many fields, not just software development.

2. Customization and Control

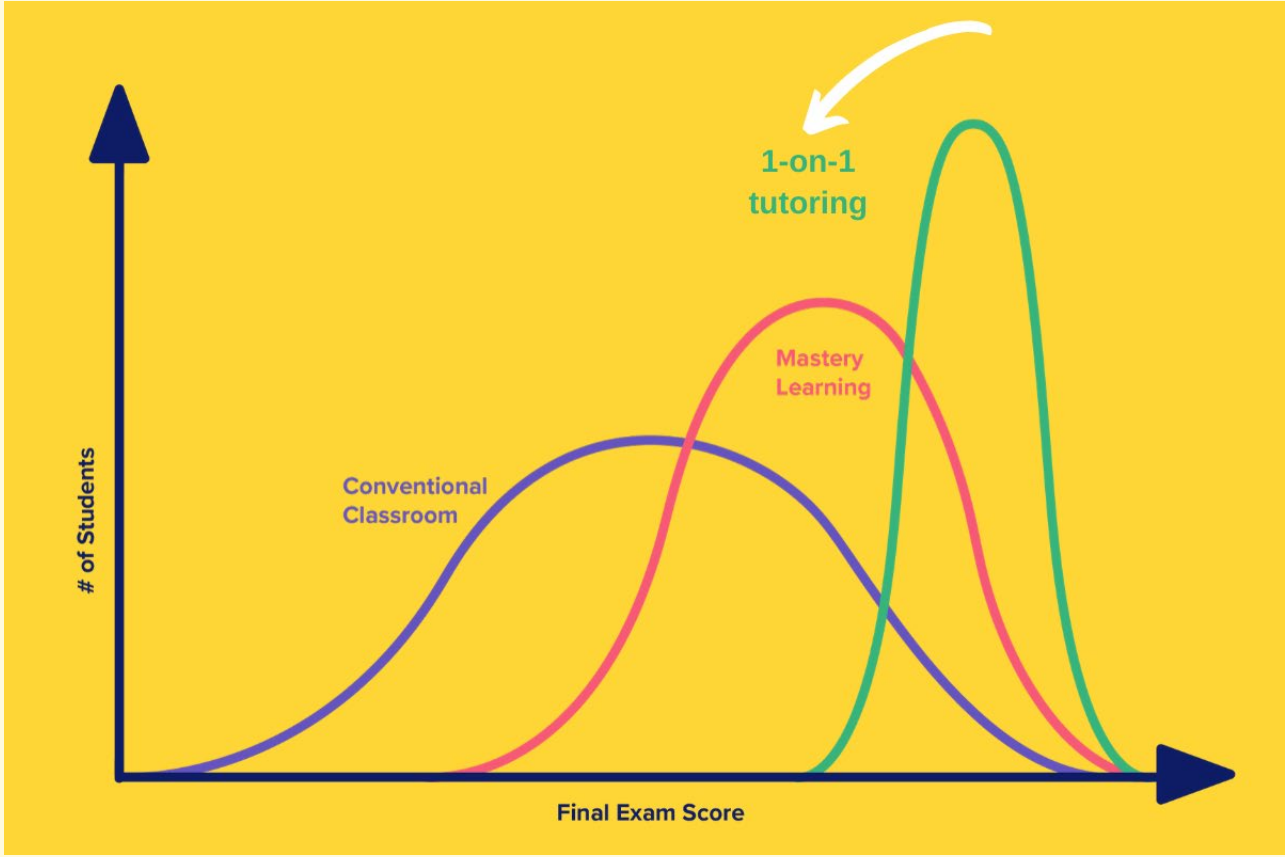
- **Tailored Solutions**: AI-generated code may not always fit specific needs or contexts. Knowing how to program allows you to customize and control your code to meet exact requirements.
- **Debugging and Optimization**: Understanding programming helps in identifying and fixing bugs, as well as optimizing code for better performance.

3. Creativity and Innovation

1-on-1 tutoring produces student performance 2 standard deviations (2σ) above the conventional classroom

The graph compares student achievement under:

- **Conventional instruction** (1 teacher to 30 students)
- **Mastery learning** (structured, feedback-oriented learning)
- **Tutorial instruction** (1-on-1 tutoring)



"The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring"Educational Researcher, Vol. 13, No. 6, pp. 4–16 (1984) Benjamin Bloom.

https://corp.tutorocean.com/tutoroceans-solution-to-blooms-2-sigma-problem/

Recommended Course Schedule

YEAR 1 SEMESTER 1				
Course Code	**Course Title**	**Type**	**AU**	**Pre-requisite**
SC1003	Introduction to Computational Thinking & Programming	F-Core	3	Nil
SC1004	Linear Algebra for Computing	Core	4	SC1003 (co-requisite)
SC1005	Digital Logic	Core	3	Nil
MH1812	Discrete Mathematics	Core	3	Nil
CC0003	Ethics & Civics in a Multicultural World	C-Core	2	Nil
CC0005	Healthy Living & Wellbeing	C-Core	3	Nil
HW0001	Introduction to Academic Communication	-	-	For student failed QET
HW0002	Calculus	-	-	For student not done Calculus
			18	

YEAR 1 SEMESTER 2				
Course Code	**Course Title**	**Type**	**AU**	**Pre-requisite**
SC1006	Computer Organisation & Architecture	Core	3	SC1005 (co-requisite)
SC1007	Data Structures & Algorithms	Core	3	SC1003
SC1008	C and C++ Programming	Core	3	SC1003
SC2000	Probability & Statistics for Computing	Core	3	Nil
SC2002	Object Oriented Design & Programming	Core	3	SC1003 OR SC1007
CC0001	Inquiry & Communication in an Interdisciplinary World	C-Core	2	Nil
CC0002	Navigating the Digital World	C-Core	2	Nil
			19	
		Year 1 Total AU	**37**	

YEAR 2 SEMESTER 1				
Course Code	**Course Title**	**Type**	**AU**	**Pre-requisite**
SC2001	Algorithm Design & Analysis	Core	3	SC1007 & MH1812
SC2005	Operating Systems	Core	3	SC1006 & SC1007
SC2203	Automata. Computability and Complexity	Core	3	SC2001 (co-requisite)
SC2006	Software Engineering	Core	3	SC2002 (co-requisite)
SC2008	Computer Network	Core	3	SC2000 & SC1004
CC0007	Science & Technology for Humanity	C-Core	3	Nil
			18	

YEAR 2 SEMESTER 2				
Course Code	**Course Title**	**Type**	**AU**	**Pre-requisite**
SC2207	Introduction to Databases	Core	3	SC2001 (co-requisite)
CC0006	Sustainability: Society, Economy & Environment	C-Core	3	Nil
ML0004	Career & Innovative Enterprise for the Future World	C-Core	2	Nil
SC3xxx/SC4xxx	Major Prescribe Elective (Choose within SC3xxx/SC4xxx)	MPE	3	Year 3 Standing
SC3xxx/SC4xxx	Major Prescribe Elective (Choose within SC3xxx/SC4xxx)	MPE	3	Year 3 Standing
	Broadening and Deepening Elective	BDE	3	Refer to Class Schedule
			17	
	Year 2 Total AU		**35**	

Lecturers

Li Fang
(**Subject coordinator**)

- Week 1-7
- Test 1- Week 7
- asfli@ntu.edu.sg

Asst/P Timothy Van Bremen

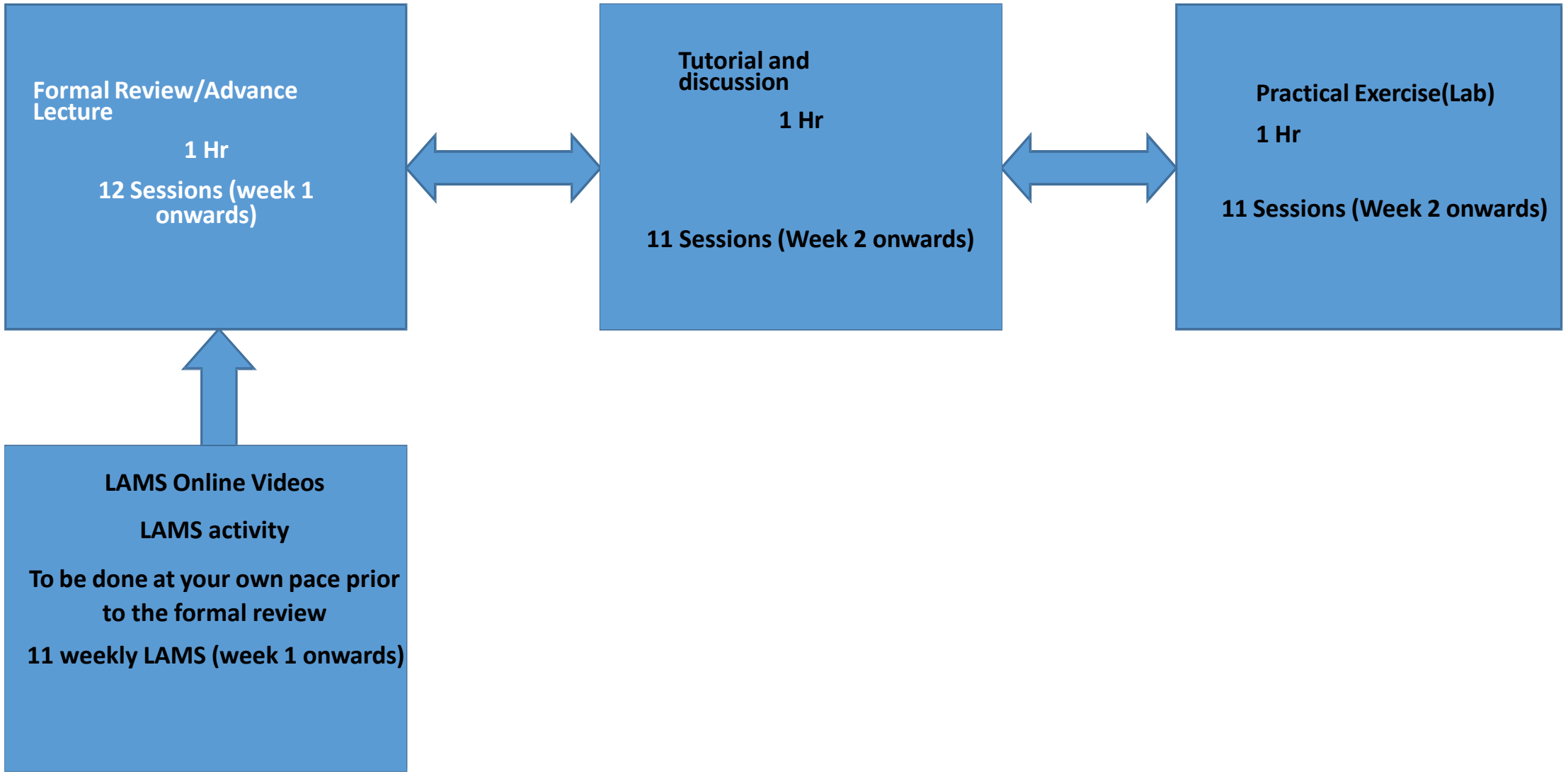
- Lecture, Tutorial, Lab: Week 8-12
- Mini Group Project
- Test 2- Week 14
- timothy.vanbremen@ntu.edu.sg

Technology-Enhanced Learning

- Own pace
- Own time
- Own place
- More interaction and examples in face-to-face sessions

Technology-Enhanced Learning (TEL) courses are conducted with a mix of online and face-to-face sessions.

Participants access and study course materials online before meeting classmates and their professors in face-to-face discussions and brainstorming sessions.


Formal Review/Advance Lecture
1 Hr
12 Sessions (week 1 onwards)
Tutorial and discussion
1 Hr
11 Sessions (Week 2 onwards)
Practical Exercise(Lab)
1 Hr
11 Sessions (Week 2 onwards)
LAMS Online Videos
LAMS activity
To be done at your own pace prior to the formal review
11 weekly LAMS (week 1 onwards)

Planned Weekly Schedule

Week	Topic	Readings	Revision Lecture	Class in labs: Tutorial	Class in labs: Lab
1	Course Introduction: Computational Thinking Concepts Programming languages: Machine language and high-level programming language Basic internal operation of computer Binary and Hexadecimal Numbers, ALU operation	On-line Video Week 1	Course Briefing		
2	Basic program structure: Pseudo code and flowchart; Data type and Variables	On-line Video Week 2	Review/Advance Lecture on CPU internal operation	Discussion question #1(week1 LAMS) CPU internal operation	Practical Exercise #1: Familiarization with Raspberry Pi (RPi) board.Python/C/Java on RPi. Remote Access of RPI, IDE for Python programming
3	Basic program structure: Boolean relational Operators, selection	On-line Video Week 3	Review/Advance Lecture on Flowchart Psuedo Code, Data Type & Variable	Discussion question #2(week2 LAMS) Flowchart Psuedo Code, Data Type & Variable	Practical Exercise #2: Variable, input, print
4	Basic program structure: Repetition	On-line Video Week 4	Review/Advance Lecture on Boolean relational Operators, selection	Discussion question #3(week3 LAMS) Boolean relational Operators, selection	Practical Exercise #3: Selection
5	Abstraction: Data abstraction (Data structure)	On-line Video Week 5	Review/Advance Lecture on Repetition	Discussion question #4(week4 LAMS) Repetition	Practical Exercise #4: Repetition

6	Abstraction: Procedural abstraction(function development)	On-line Video Week 6	Review/Advance Lecture on Data abstraction (Data structure)	Discussion question #5(week5 LAMS) Data abstraction (Data structure)	Practical Exercise #5 : Data abstraction (Data structure)
7	Exception handling	On-line Video Week 7	Review/Advance Lecture on Procedural abstraction(function development) part 1	Discussion question #6 (week 6 LAMS) Procedural abstraction(function development) part 1	Practical Exercise #6: Procedural abstraction(function development) part 1
Saturday of week 7: Quiz 1: MCQ					
Recess week					
8	Jupyter Notebook Introduction	On-line Video Jupiter Notebook tutorial	Review/Advance Procedural abstraction(function development) part 2 Lecture on Exception handling	Discussion question #7 Procedural abstraction(function development) part 2 Exception handling	Practical Exercise #7: Procedural abstraction(function development) part 2 Exception handling
9	Decomposition Divide and Conquer Recursion Case studies	On-line Video Week 9	Mini Project briefing Jupyter Notebook Demo, Case studies	Discussion question #8 Jupyter Notebook application	Practical Exercise #8: Jupyter Notebook application
10 (HBL)	Pattern recognition Complexity	On-line Video Week 10	Review/Advance Lecture on Decomposition	Discussion question #9 (week 9 LAMS) Decomposition Recursion	Practical Exercise #: Decomposition
11	Algorithms design Sorting algorithms Searching algorithms	On-line Video Week 11	Review/Advance Lecture on Pattern Recognition	Discussion question #10(week 10 LAMS) Pattern recognition Iterative accumulation complexity	Practical Exercise #10: Pattern recognition

12	NO LAMS VIDEO	Review/Advance Lecture on Algorithms design	Discussion question #11 (week 11 LAMS) Algorithms design	Practical Exercise #11: Algorithms design
13		No class	Mini project (Attendance is optional, but the TA will be present.)	
14	**Monday: Quiz 2: coding. Sunday: Deadline of Mini project.**			

Course review/advanced Lecture

– Topic review & clarification, advanced topics

– ONLINE lecture, Monday 9:30-10:20

– **start from week 1**

1

Course Content

Course Information

Course Content

LAMS Lectures (Pre-recorded)

This folder contains lecture notes and learning activities, including video lectures and simple revision
week's topics. All revision questions in the learning activities are single-attempt and will be graded bas
to the Assessment for LAMS TEL MCQs, which will count as 10% of your final grade. E-learning lecture

Review/Advanced Lectures (Online face-to-face)

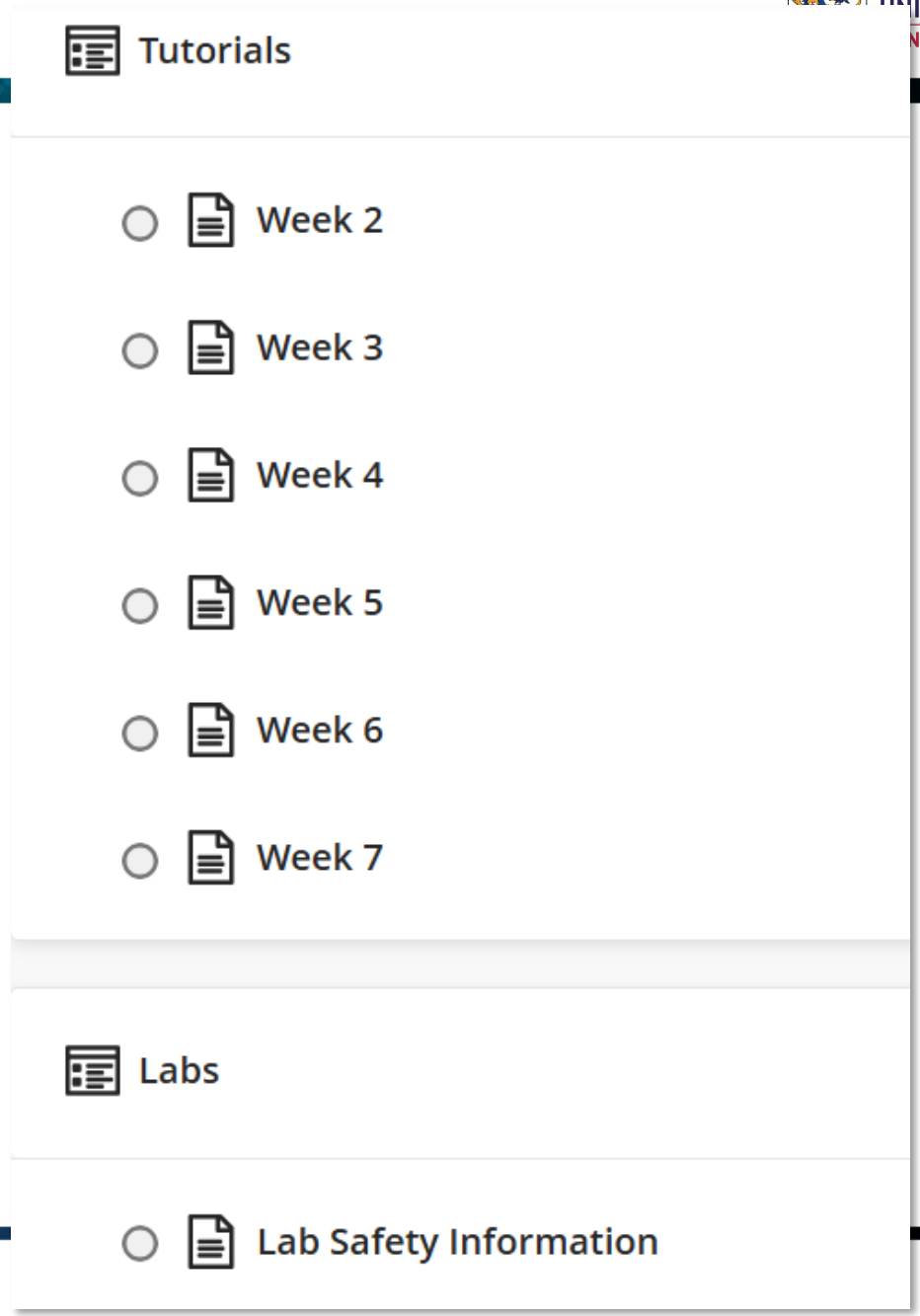
We will have a weekly Review Lecture to go over the critical parts of the LAMS video and provide more
LAMS video. This section will contain the slides and supporting material related to the Review Lecture
and posted on NTULearn. Please feel free to follow the material along with the recorded videos and re

Tutorials and Labs

- **Weekly 2-hour Tutorial and Lab class**
  - **start from week 2**
  - Group Discussions
  - Practical Hands-on exercises

2

Note: One hour tutorial + One hour lab in the same CCDS lab



LAMS pre-recorded Lecture video and LAMS Learning Activity

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

LAMS Lectures (Pre-recorded)

Visible to students

This folder contains lecture notes and learning activities, in
week's topics. All revision questions in the learning activitie
contribute to the Assessment for LAMS TEL MCQs, which w
this folder.

3

LAMS Week 1

Release conditions Edit release conditions

This folder consists of lecture notes and learning activities
following modules: • Course Introduction: Computational T
Language • Basic Computer Operations: Binary and Hexade

LAMS Week 1.1

Release conditions Edit release conditions

This folder consists of lecture notes and learning activities
following modules: Course Introduction: Computational Th
Language

Lecture Notes: Programming Languages

Hidden from students

Instructions

Hidden from students

Learning Activity: Course Introduction

Hidden from students

No due date

This sequence of learning activities consists of video lectures and simple questior
Watch the video lectures. Write down key concepts learned from each video. Ans
button to view the online lesson. When prompted, please allow the loading of 'un
online lesson will open in a new window and the browser should be maximised f
learning activities will be credited towards your participation in the course.

Learning Activity: Programming Languages

Hidden from students

No due date

This sequence of learning activities consists of video lectures and simple questior
Watch the video lectures. Write down key concepts learned from each video. Ans
button to view the online lesson. When prompted, please allow the loading of 'un
online lesson will open in a new window and the browser should be maximised f

Tutors (CCDS faculty)

Chong Leng Leng Josephine	3	FBAC1	FCS2	FCS5			
Erik Cambria	6	FCEC	FCSA	FCSI	FCSJ	FDAB	FDAC
Gu Yan Chloe	5	FCEB	FCE1	FCE2	FCE3	FCEA	
Hui Siu Cheung	2	FAIA	FCSD				
Li Fang	1	FCMC					
Ong Chin Ann	4	FCSC	FCSE	FDDA	FECD2		
Quek Hiok Chai	2	FBAC2	FACD				
Sourav Saha Bhowmick	6	FCSF	FCSG	FCSH	FDAA	FDAD	FDAE
Tan Wen Jun	6	FDAF	FCS6	FCMD	FCS1	FCS3	FCS4
Timothy van Bremen	2	FCED	FECD1				
Yu Han	4	FCMB	FCS7	FMAC1	FMAC2		
Zhang Jiehuang	5	FBAC3	FCMA	FCSB	FEL2	FEL3	
Zheng Jianmin	2	FDDB	FDDC				

Lab supervisors (PhD students)

GS assigned (default is 4 groups, numbers in brackets show group # which is not 4)	GS Email	Lab Group 1	Lab Group 2	Lab Group 3	Lab Group 4	Lab Group 5
AJITH SANTHISENAN (1)	SANTHISE002@e.ntu.edu.sg	FCS4				
BURTON-BARR JONATHAN WESTON (2)	BURT0002@e.ntu.edu.sg	FCSF	FCE1			
FERREIRA CARDOSO TELES FORTES ARMANDO LUIS (5)	ARMANDOL001@e.ntu.edu.sg	FBAC1	FACD	FDAD	FCSH	FCEA
HO ZHENG YI (1)	ZHENGYI001@e.ntu.edu.sg	FCSI	~~FCMD~~			
HU YIHUA(5)	YIHUA001@e.ntu.edu.sg	FBAC3	FCED	FCSA	FCS3	FCSC
HUANG YIFAN(5)	YIFAN005@e.ntu.edu.sg	FCE2	FCS5	FCE3	FDAC	FDAA
JING JIAZHENG (1)	JIAZHENG001@e.ntu.edu.sg	FBAC2				
KEVIN CONSTANTINE HALIM(5)	KEVINCON001@e.ntu.edu.sg	FDDB	FCEC	FCS1	FAIA	
LEE ZHENG HAN NICHOLAS (4)	LE0003AS@e.ntu.edu.sg	FCMB	FCSJ	FCSD	FCMD	~~FCSI~~
LI SHENGGUI (2)	SHENGGUI001@e.ntu.edu.sg	FCMC	FCEB			
LIAO CHANG (4)	CHANG019@e.ntu.edu.sg	FDAE	FEL2	FEL3	FCS6	
SHEN CHENG (4)	S230143@e.ntu.edu.sg	FCSB	FMAC1	FDAF	FCSE	~~FCS4~~
SUN WENHAO (5)	WENHAO006@e.ntu.edu.sg	FCMA	FCSG	FCS7	FDDC	FECD2
XU MUYU (5)	MUYU001@e.ntu.edu.sg	FDDA	FDAB	FCS2	FMAC2	FECD1

Course Assessments

No.	Component	Weightage	Team/Individual
1	Continuous Assessment (CA): Assignment(TEL MCQs)	10	Individual
2	Continuous Assessment (CA): Test/Quiz(Online MCQs based quizzes)	15	Individual
3	Continuous Assessment (CA): Test/Quiz(Online MCQ/coding based quizzes)	45	Individual
4	Continuous Assessment (CA): Project(Group project)	30	Team

Weekly LAMS Knowledge Check Questions (10%)

3

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Single attempt

Graded based on correctness.

Online MCQ quiz(15%)

- **Date and Time: Week 7 Saturday**
- **Online MCQ quiz(15%)**
  - **Duration: 1 hours**
  - **Question type: Single (default)/Multiple correct (state specially) answer MCQs.**
- **Content: Week 1 – Week 7 lecture content (until procedure abstraction part 1).**
- **Number of questions: 30.**
- **Venue: CCDS labs**

Logistical details will be announced as the date gets closer, serving as a reminder as well.

Online coding quiz(45%)

- **Date and Time: Week 14 Monday**
- **Online coding quiz(45%)**
  - **Duration: 2 hours**
  - **Content: all content.**
  - **Venue: CCDS labs**

Logistical details will be announced as the date gets closer, serving as a reminder as well.

Group Assignment 30%
Deadline: **Sunday of Week 14**

- Group Assignment will be release in week 8
- Sunday of week 14: code, report submission, and Group Assignment oral presentation deadline
- Oral presentation:
  - Via Physical/Online meeting
  - Arrange Group Assignment oral presentation date and time earlier with your TA
  - Every student must take turn to present

slido

Please download and install the Slido app on all computers you use

Have you learned Python before?

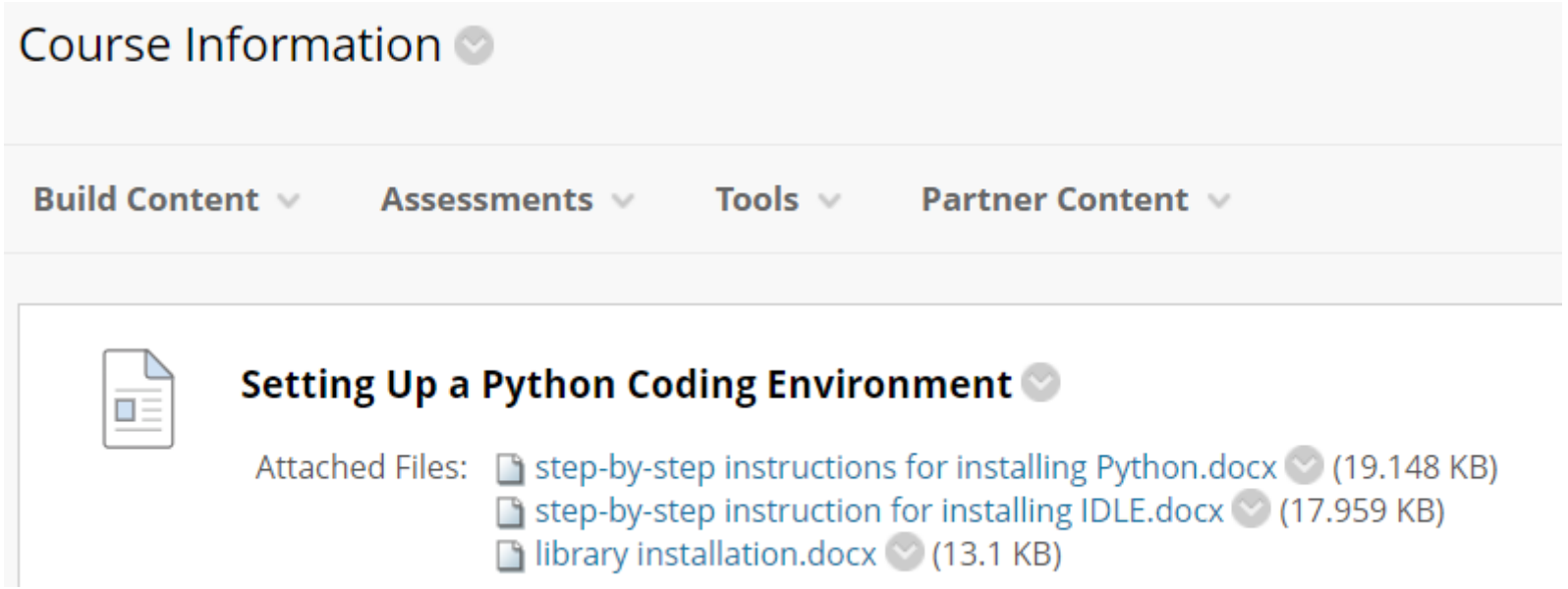
ⓘ Start presenting to display the poll results on this slide.

You are free to choose your preferred Python IDE

https://jolcasa.amebaownd.com/posts/40300740/

https://www.simplilearn.com/tutorials/python-tutorial/python-ide

Preparation



Always download the latest version.

python.org/downloads/

Python | PSF | Docs | PyPI | Jobs | Community

python™

Donate | Search | GO | Socialize

About | Downloads | Documentation | Community | Success Stories | News | Events

Download the latest version for Windows

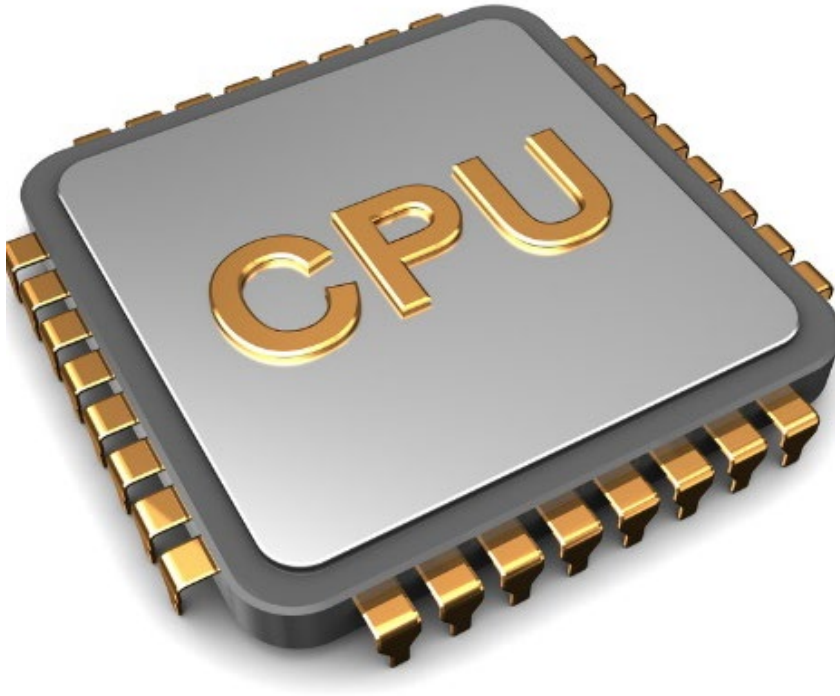
Download Python 3.11.4

Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python 3.12? Prereleases, Docker images

MACHINE LANGUAGE

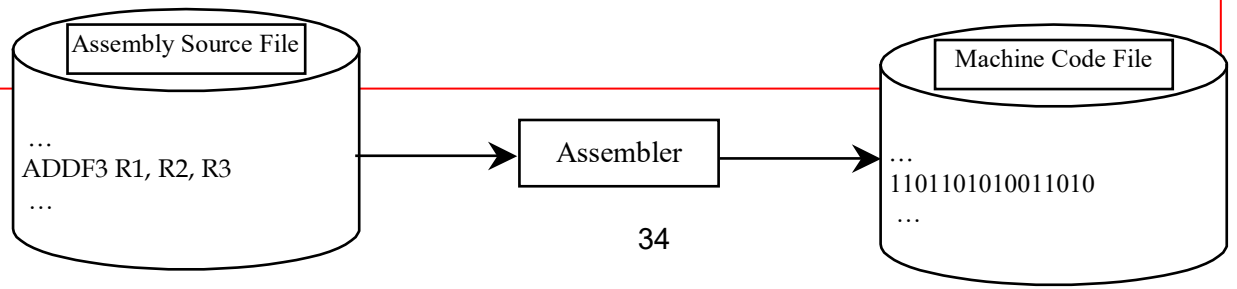
- Central Processing Units (CPUs) have different instruction sets, defining different languages, called the **machine language**
- Machine language is a set of primitive instructions built into every computer. The instructions are in the form of binary code, so you have to enter binary codes for various instructions. Program with native machine language is a tedious process. Moreover the programs are highly difficult to read and modify. For example, to add two numbers, you might write an instruction in binary like this:
-
    - 1101101010011010



ASSEMBLY LANGUAGE

Assembly languages were developed to make programming easy. Since the computer cannot understand assembly language, however, a program called assembler is used to convert assembly language programs into machine code. For example, to add two numbers, you might write an instruction in assembly code like this:

ADDF3 R1, R2, R3



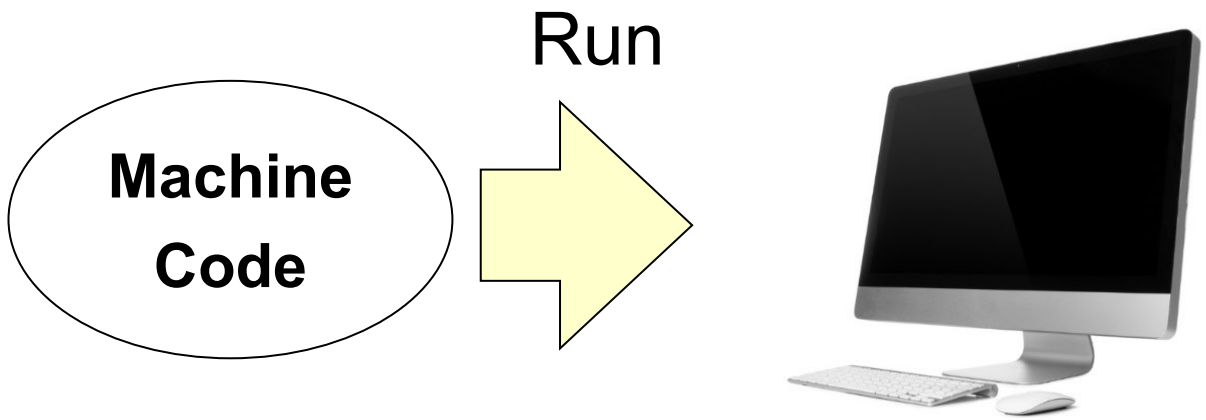
HIGH-LEVEL LANGUAGE

The high-level languages are English-like and easy to learn and program. For example, the following is a high-level language statement that computes the area of a circle with radius 5:

```
area = 5 * 5 * 3.1415;
```

MACHINE LANGUAGE VS. PROGRAMMING LANGUAGE

To run a programme on a computer, the programme instructions must be in the corresponding machine language so that the instructions can be understood and run by the CPU

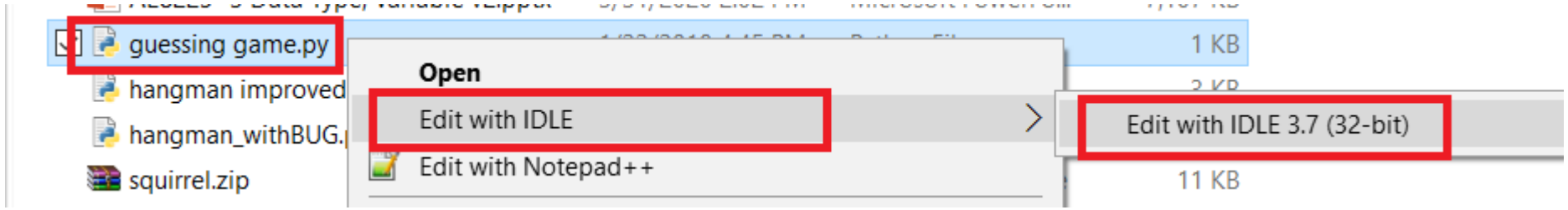


Python installation

https://www.python.org/downloads/

Always download the latest version.

Note: Includes IDLE. SO after install Python, IDLE is automatically installed.



guessing game.py - D:\LF work\LF current Courses\MSc SSS\2019T3\Lecture notes\Week 1\guessing game.py

File Edit Format Run Options Window Help

Python Shell

Check Module Alt+X

Run Module F5

```
import rand
n = random.
guess = int                    integer from 1 to 99: "))
while n !=
    print
    if guess < n:
        print ("guess is low")
        guess = int(input("Enter an integer from 1 to 99: "))
    elif guess > n:
        print ("guess is high")
        guess = int(input("Enter an integer from 1 to 99: "))
    else:
        print ("you guessed it!")
        break
    print
```

Turtle - advanced Sakura.py - F:\LF work\LF running cour

File Edit Format Run Options Window Help

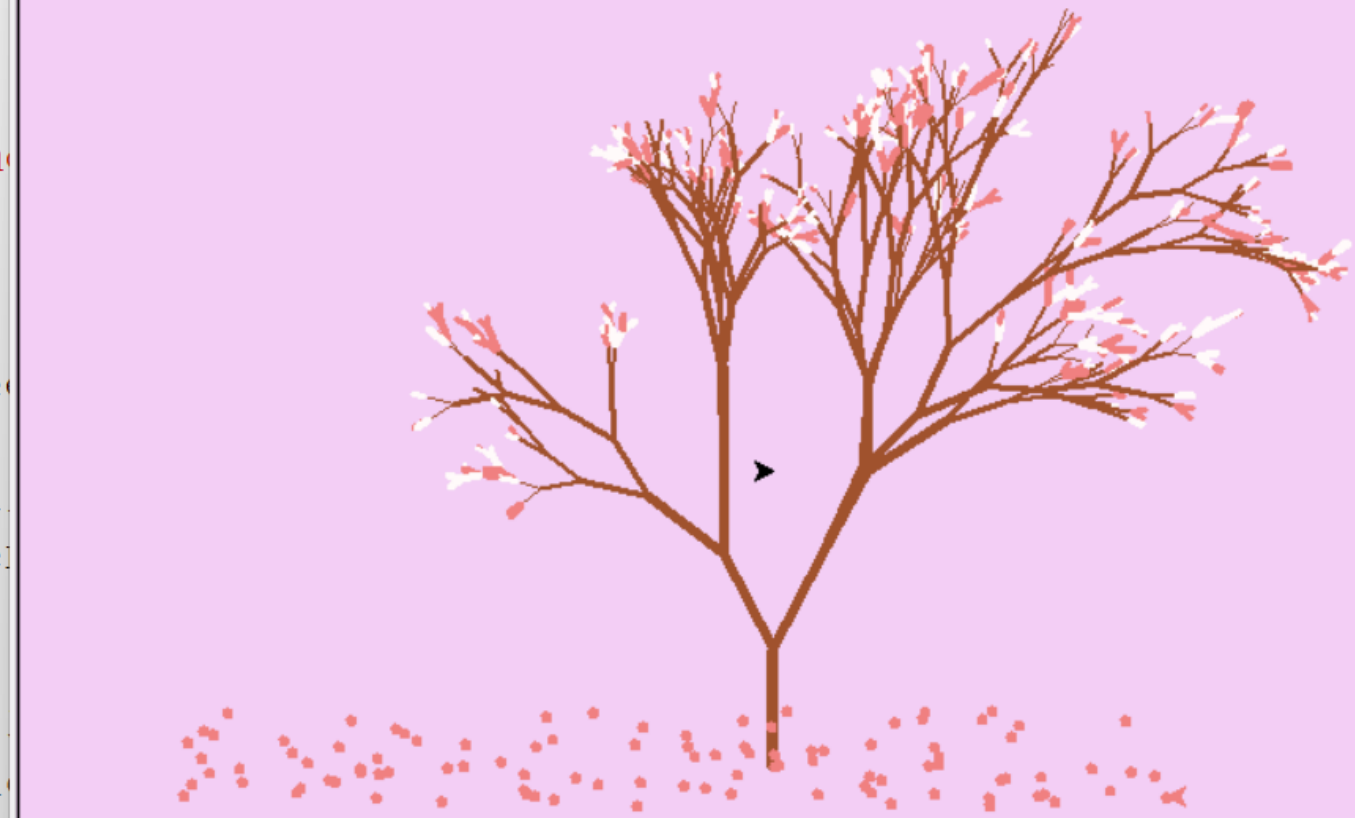
```
##https://programsol
'''
Python Program to Cr
'''

# Import necessary m
import turtle
import random
from turtle import *
from time import sle

static_turtle = turt
frame = turtle.Scree

def tree(branch_len,
    if branch_len >
        if 9 <= bran
            if rando
                inne
            else:
                inne
            inner_tu
```

Python Turtle Graphics



Python Tutor: Visualize code in Python, JavaScript, C, C++, and Java

Python 3.6
known limitations

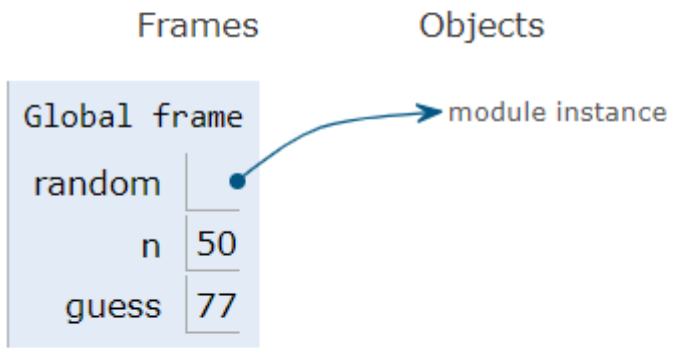
```
1  import random
2  n = random.randint(1, 99)
3  guess = int(input("Enter an integer from 1 to 99: "))
4  while n != "guess":
5      print
6      if guess < n:
7          print ("guess is low")
8          guess = int(input("Enter an integer from 1 to 9
9      elif guess > n:
10         print ("guess is high")
11         guess = int(input("Enter an integer from 1 to 9
12     else:
13         print ("you guessed it!")
14         break
15     print
```

Edit this code

line that just executed
next line to execute

Print output (drag lower right corner to resize)

```
Enter an integer from 1 to 99: 77
guess is high
```



https://pythontutor.com/

The provided image shows a Python program with a simple number-guessing game, and it is being visualized using an execution tracer tool that highlights the flow of the program. Here's a breakdown and explanation of the code and the visualization:

Code Explanation

1. **Imports and Initialization**:

```python
import random
n = random.randint(1, 99)
```

   - The `random` module is imported.
   - A random integer `n` between 1 and 99 is generated and stored.

2. **User Input and Loop**:

```python
guess = int(input("Enter an integer from 1 to 99: "))
while n != guess:
    print
    if guess < n:
        print("guess is low")
        guess = int(input("Enter an integer from 1 to 99: "))
    elif guess > n:
```

Python is a popular programming language for many reasons

Simplicity and Readability:

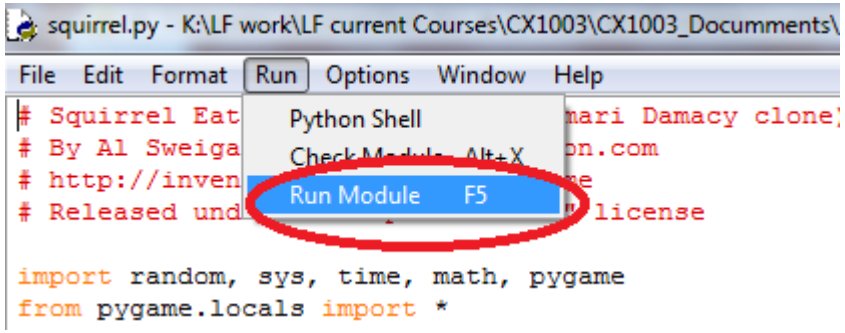
Versatility:

Extensive Libraries and Frameworks:

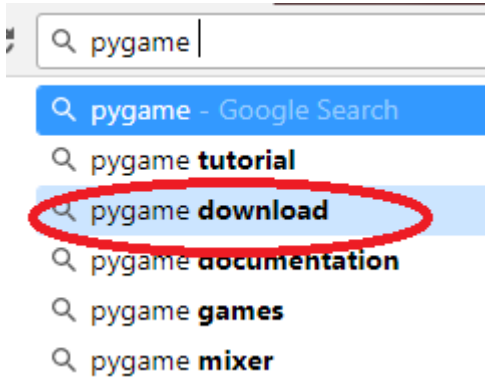
High Demand:

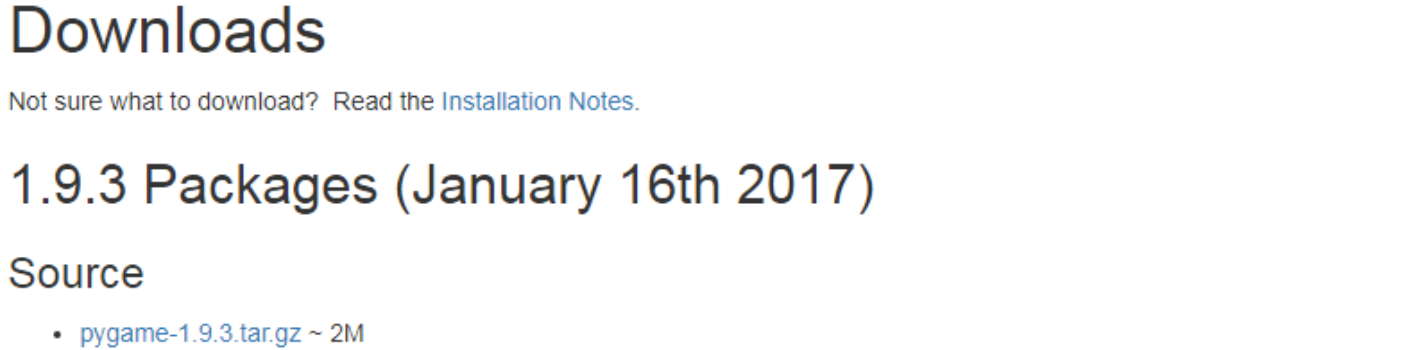
Rapid Prototyping:

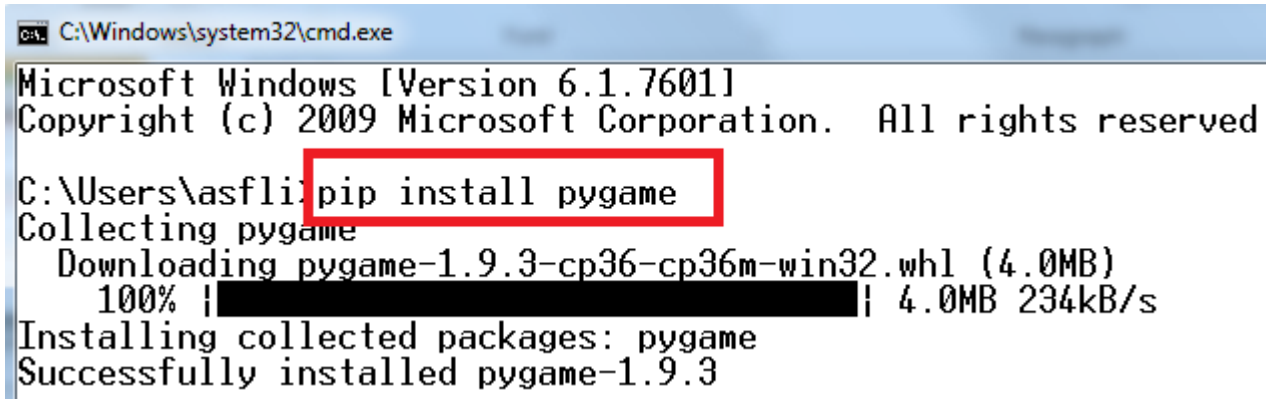
Edit/Run Python Program: Install useful modules



Pygame:





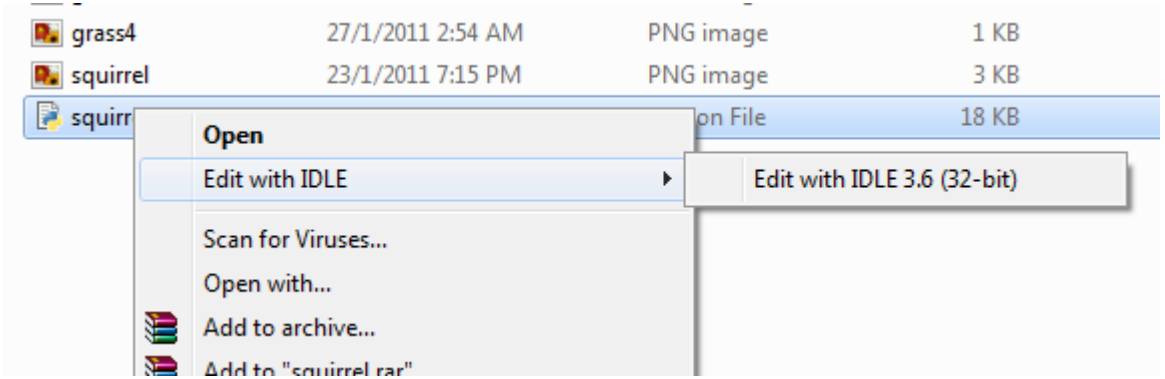


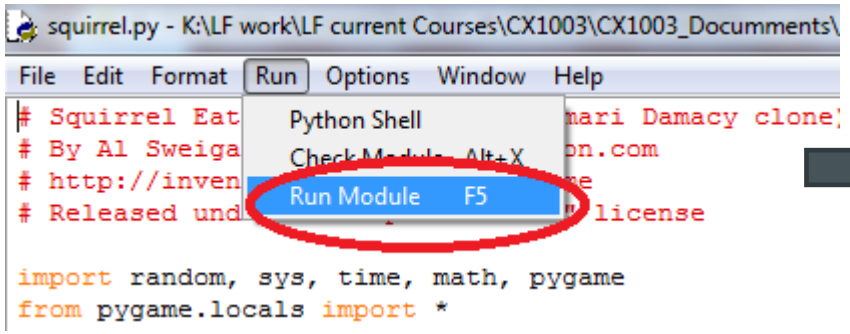
pip (package manager)

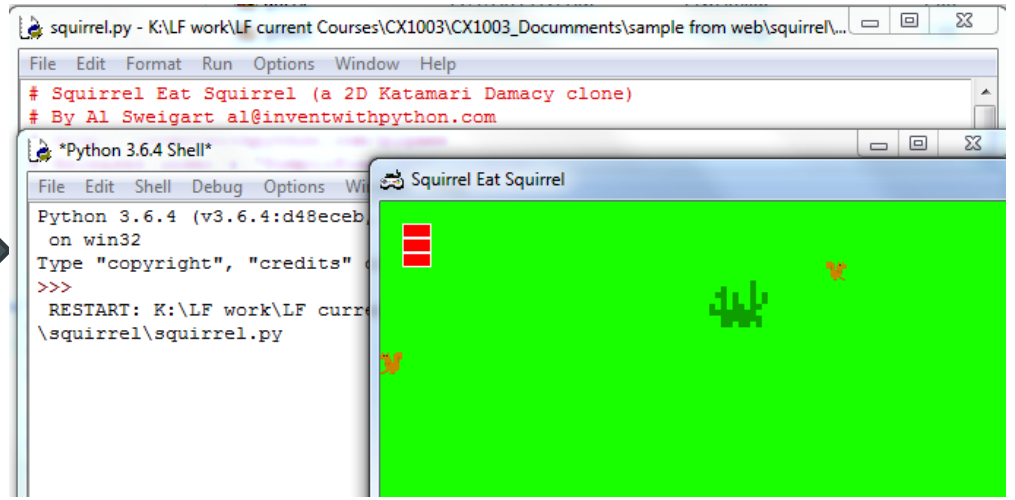
pip is a package management system used to install and manage software packages written in Python.

```
python.exe -m pip install --upgrade pip
```

Edit/Run Python Program







Any Question?