What is difference between React Native, Flutter and Kivy?

- React native depends on JS and React mobile app, supports cross platform and uses with iOS, Arduino and web. React native is supported by meta (Facebook).it is strong in backend because of JS. It doesn't have built in components; you can control by coding to build control.
- Flutter depends on Dart language, supports cross platforms like web, mac, iOS, window and Arduino. Flutter is supported by Google. It has built in components and automatic control.it has high performance in animation.it is high speed in execution of process.
- Kivy depends on python. It is strong in desktop application. It is also used in mobile applications.

What are drag and drop Python GUI designers?

- PyQt.
- Tkinter.

SOLID principles

- SOLID Principle in Programming: Understand With Real Life Examples - GeeksforGeeks
- SOLID Principles in Arabic - #5 Dependency inversion - YouTube

Type of SW architecture patterns

- Types of Software Architecture Patterns - GeeksforGeeks

What is paging?

- Paging is a memory management scheme that eliminates the need for a contiguous allocation of physical memory. The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging. The basic purpose of paging is to separate each procedure into pages. Additionally, FRAMES WILL BE USED TO SPLIT THE MAIN MEMORY. THIS SCHEME PERMITS THE PHYSICAL ADDRESS SPACE OF A PROCESS TO BE NON – CONTIGUOUS.

- IN PAGING, THE PHYSICAL MEMORY IS DIVIDED INTO FIXED-SIZE BLOCKS CALLED PAGE FRAMES, WHICH ARE THE SAME SIZE AS THE PAGES USED BY THE PROCESS. THE PROCESS'S LOGICAL ADDRESS SPACE IS ALSO DIVIDED INTO FIXED-SIZE BLOCKS CALLED PAGES, WHICH ARE THE SAME SIZE AS THE PAGE FRAMES. WHEN A PROCESS REQUESTS MEMORY, THE OPERATING SYSTEM ALLOCATES ONE OR MORE PAGE FRAMES TO THE PROCESS AND MAPS THE PROCESS'S LOGICAL PAGES TO THE PHYSICAL PAGE FRAMES.
- THE MAPPING BETWEEN LOGICAL PAGES AND PHYSICAL PAGE FRAMES IS MAINTAINED BY THE PAGE TABLE, WHICH IS USED BY THE MEMORY MANAGEMENT UNIT TO TRANSLATE LOGICAL ADDRESSES INTO PHYSICAL ADDRESSES. THE PAGE TABLE MAPS EACH LOGICAL PAGE NUMBER TO A PHYSICAL PAGE FRAME NUMBER.
    - Paging in Operating System - GeeksforGeeks

WHAT IS FRAGMENTATION?

- THE PROCESS OF DIVIDING A COMPUTER FILE, SUCH AS A DATA FILE OR AN EXECUTABLE PROGRAM FILE, INTO FRAGMENTS THAT ARE STORED IN DIFFERENT PARTS OF A COMPUTER'S STORAGE MEDIUM, SUCH AS ITS HARD DISC OR RAM, IS KNOWN AS FRAGMENTATION IN COMPUTING. WHEN A FILE IS FRAGMENTED, IT IS STORED ON THE STORAGE MEDIUM IN NON-CONTIGUOUS BLOCKS, WHICH MEANS THAT THE BLOCKS ARE NOT STORED NEXT TO EACH OTHER.
- TYPE OF FRAGMENTATION:
    - INTERNAL FRAGMENTATION: INTERNAL FRAGMENTATION OCCURS WHEN THERE IS UNUSED SPACE WITHIN A MEMORY BLOCK. FOR EXAMPLE, IF A SYSTEM ALLOCATES A 64KB BLOCK OF MEMORY TO STORE A FILE THAT IS ONLY 40KB IN SIZE, THAT BLOCK WILL CONTAIN 24KB OF INTERNAL FRAGMENTATION. WHEN THE SYSTEM EMPLOYS A FIXED-SIZE BLOCK ALLOCATION METHOD, SUCH AS A MEMORY ALLOCATOR WITH A FIXED BLOCK SIZE, THIS CAN OCCUR.

- EXTERNAL FRAGMENTATION: EXTERNAL FRAGMENTATION OCCURS WHEN A STORAGE MEDIUM, SUCH AS A HARD DISC OR SOLID-STATE DRIVE, HAS MANY SMALL BLOCKS OF FREE SPACE SCATTERED THROUGHOUT IT. THIS CAN HAPPEN WHEN A SYSTEM CREATES AND DELETES FILES FREQUENTLY, LEAVING MANY SMALL BLOCKS OF FREE SPACE ON THE MEDIUM. WHEN A SYSTEM NEEDS TO STORE A NEW FILE, IT MAY BE UNABLE TO FIND A SINGLE CONTIGUOUS BLOCK OF FREE SPACE LARGE ENOUGH TO STORE THE FILE AND MUST INSTEAD STORE THE FILE IN MULTIPLE SMALLER BLOCKS. THIS CAN CAUSE EXTERNAL FRAGMENTATION AND PERFORMANCE PROBLEMS WHEN ACCESSING THE FILE.
- What is Fragmentation in Operating System? - GeeksforGeeks

CPU SCHEDULING

- CPU Scheduling in Operating Systems - GeeksforGeeks
- Types of CPU Scheduling algorithms (opengenus.org)

HOW TO SET PRIORITY IN QUOTE IN PYTHON

- Introduction to Priority Queues in Python | Built In
- 3 WAYS TO BUILD PRIORITY QUEUES IN PYTHON
    - USING LIST: THIS STRATEGY IS EFFICIENT IF YOU DON'T NEED TO MAKE MANY INSERTIONS.
    - USING HEAPQ: THIS VERSION SUPPORTS O(LOGN) TIME FOR INSERTION AND THE SMALLEST ELEMENT.
    - USING QUEUE.PRIORITYQUEUE: THIS APPROACH SUPPORTS CONCURRENT PROCESSES AND IT'S A CLASS INTERFACE.