مكتب

Quick Reference Notes

Week 6: Planning Before Coding

7th Grade Computer Science

1 Introduction

1.1 This Week's Big Question

What turns human ideas into step-by-step code? This week, we'll discover the secret tools programmers use to plan their solutions before writing a single line of Python. Just like architects draw blueprints before building, we'll learn to design our programs first!

Prerequisites

Before starting this week, you should be comfortable with:

- Basic Python syntax and structure
- Creating and using variables (Week 3)
- Using `input()` and `print()` for user interaction (Week 4)
- Expressions and operators (Week 5)

1.2 What You Already Know

You've written Python programs that store data, get user input, and calculate results. You know how to make Python do what you want - but have you ever gotten stuck figuring out where to start or what to code next?

1.3 What You'll Be Able to Do

By the end of this week, you'll:

- Write pseudocode to plan any program in plain language
- Draw flowcharts to visualize program logic
- Break complex problems into manageable pieces
- Save time by thinking before typing

2 VIDEO 1: Planning with Pseudocode

2.1 What is Pseudocode?

Pseudocode is a way to plan your program using plain english language - like writing a recipe before cooking. It's "fake code" that helps you think through the logic without worrying about Python syntax.

2.2 Why Write Pseudocode First?

Think about making biryani:

- Would you start cooking without knowing the steps?
- What if you forgot to soak the rice first?
- What if you added spices at the wrong time?

Programming is the same! Planning helps you:

- Spot problems before coding
- Organize your thoughts
- Explain your idea to others
- Code faster with fewer bugs

2.3 Pseudocode Example: Temperature Converter

Let's plan a program that converts Celsius to Fahrenheit:

Pseudocode

```
1. Ask user for temperature in Celsius
2. Get the input and store it
3. Convert text input to a number
4. Calculate Fahrenheit using formula: (C × 9/5) + 32
5. Display the result with a friendly message
```

The psudocode above now directly translate to Python as follows:

```python
# Step 1 & 2: Get input
celsius = input("Enter temperature in Celsius: ")

# Step 3: Convert to number
celsius = float(celsius)

# Step 4: Calculate
fahrenheit = (celsius * 9/5) + 32

# Step 5: Display
print(f"{celsius} C equals {fahrenheit} F")
```

Rule

Pseudocode Guidelines:

- Use simple, clear, natural language *(English)*
- Number your steps
- Focus on what to do, not how
- Include all major decisions and actions
- Don't worry about perfect syntax *(because we're using English not Python)*

2.4 Another Example: Number Guessing Hint

Pseudocode

```
1. Set secret number to 42
2. Ask user to guess a number
3. Get their guess and convert to integer
4. if guess equals secret number then
      Say "Correct!"
   else if guess is less than secret number then
      Say "Too low!"
   else
      Say "Too high!"
```

Note that we're using `if` conditions in the psudocode above, but we haven't learned about them in Python yet. But you've seen these in Scratch, so you can still understand the logic and what the program above is meant to do.

Tip

Start with pseudocode for every program, even simple ones. It's like reading through a recipe before cooking - you need to know all the ingredients and steps before you start, or you might realize halfway through that you're missing something important... when it's too late!

Quick Check

Write pseudocode for a program that:

1. Asks for your age
2. Calculates how many months old you are
3. Displays the result

3 VIDEO 2: Planning with Flowcharts

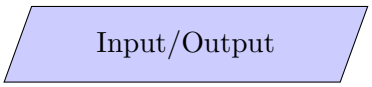
3.1 Visualizing Logic with Flowcharts

Flowcharts are diagrams that show how a program flows from start to finish. They use shapes to represent different types of actions - like a map for your code!

3.2 Basic Flowchart Symbols

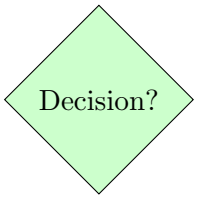
Every shape has a meaning:

Oval: Beginning or end of program



Parallelogram: Getting input or showing output

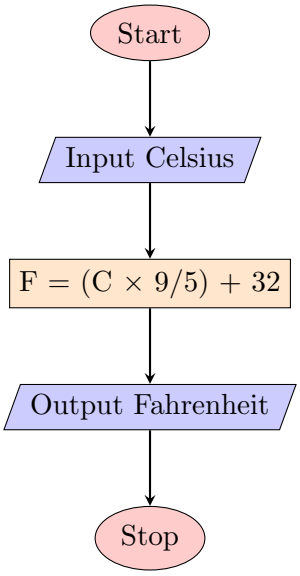
Rectangle: Calculations or actions



Diamond: Questions with Yes/No answers. These are like decisions and loops in scratch

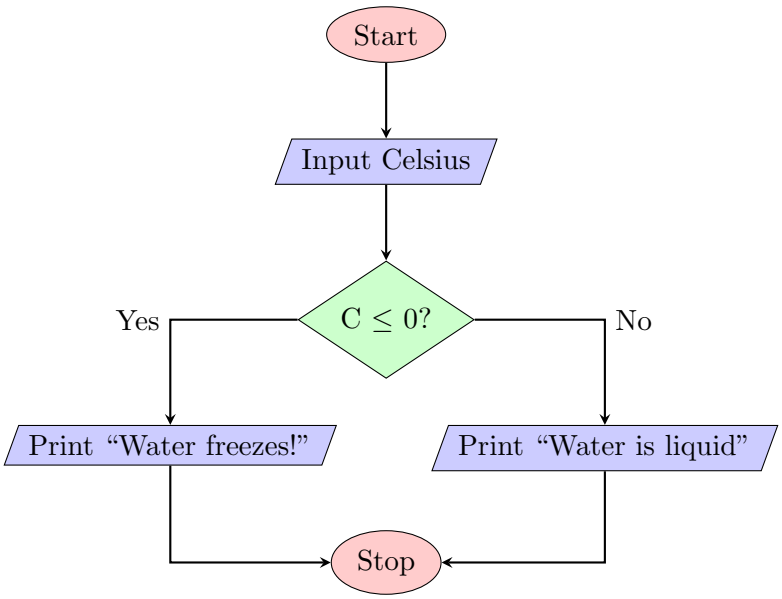
3.3 Simple Flowchart Example

Here's our temperature converter as a flowchart:



3.4 Flowchart with Decision

Let's add a decision - checking if water would freeze:



Rule

Flowchart Rules:

- Always start with "**Start**" and end with "**Stop**"
- Arrows show the flow direction
- Each symbol has one entry point
- Decisions have two exits: Yes and No
- Keep it simple and clear

Common Error

Common Flowchart Mistakes:

- Forgetting Start/Stop ovals
- Missing arrows between shapes
- Dead ends - every path must reach Stop
- Using wrong shapes (e.g., rectangle for input)

Quick Check

Match the action to the correct flowchart shape:

1. Calculate area = length × width → ?
2. Ask user for their name → ?
3. Check if age ≥ 13 → ?
4. Begin the program → ?

4 VIDEO 3: Problem Decomposition

4.1 Breaking Down Big Problems

Problem decomposition means breaking a complex problem into smaller, manageable pieces. It's like eating a pizza - you don't swallow it whole, you eat it slice by slice!

4.2 The Decomposition Process

Let's learn the 5-step process of decomposition with a simple example:

Problem: Find the largest of three numbers entered by the user

Step 1: Understand the Big Problem

- **Goal:** Get 3 numbers, find which is biggest, tell the user

Step 2: Identify Major Tasks

1. Get three numbers from user
2. Find the largest number
3. Display the result

Step 3: Break Down Each Task

Task 1: Get numbers
- Ask for first number
- Ask for second number
- Ask for third number

Task 2: Find largest
- Compare first with second
- Compare winner with third

Task 3: Display
- Show which number is largest

Step 4: Order the Pieces
Must get numbers before comparing them!
1. Input → 2. Process → 3. Output

Step 5: Connect the Pieces
Numbers from **Task 1** → feed into → **Task 2** → result goes to → **Task 3**

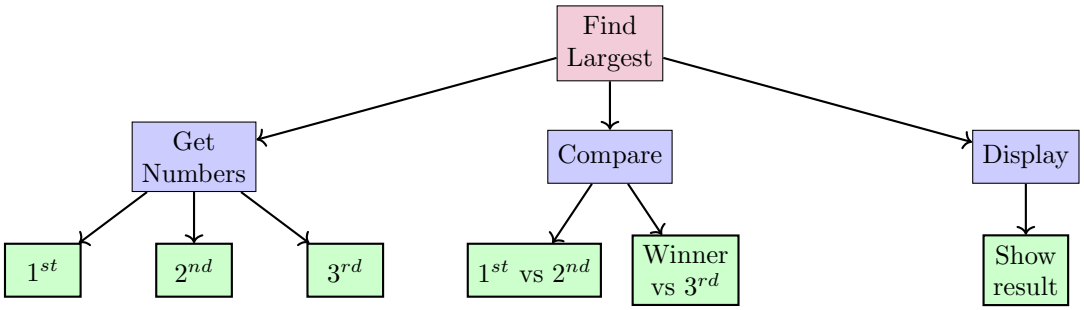
4.3 From Decomposition to Pseudocode

Now our pseudocode almost writes itself:

Pseudocode

1. Get first number from user
2. Get second number from user
3. Get third number from user
4. Set largest = first number
5. `if` second number > largest then
   largest = second number
6. `if` third number > largest then
   largest = third number
7. Display "The largest number is: " + largest

4.4 Visual Decomposition



Tip

Quick Decomposition Recipe:

1. One sentence goal
2. 3-5 main chunks
3. Break each chunk into 2-4 pieces
4. Order them logically
5. Show how data flows between them *(i.e arrange in proper order)*

4.5 Real-World Decomposition

This process works for everything:

- **Making tea**: Boil water → Add tea → Add sugar → Serve
- **Homework**: List subjects → Do easiest first → Check each → Pack bag
- **Coding**: Understand → Plan → Code → Test

Landmark Moment

Problem decomposition is a superpower! Master this skill and no problem will seem too big!

Quick Check

Decompose this problem: "Check if someone can create a social media account *(must be 13+)*"
What are the three main tasks?

1. Get their _____
2. Check if _____
3. Tell them _____
4. Otherwise tell them _____

5 VIDEO 4: Summary

5.1 From Ideas to Code

This week, we learned three powerful planning tools:

1. **Pseudocode** - Write your logic in plain language first
2. **Flowcharts** - Draw your program's path visually
3. **Decomposition** - Break big problems into bite-sized pieces

These aren't just school exercises - real programmers use these tools every day!

Tip

You can use any of these tools individually:
Write psudocode to outline your logic, and move straight to coding.

Or use these tools together in case of a complex program:
For example, you might start with pseudocode, then draw a flowchart to visualize it, and finally decompose the problem into smaller tasks.

It's completely upto you what you're comfortable with. The key is to plan before you code.

6 Quick Reference

Quick Reference

Week 6 Planning Toolkit:
Pseudocode Format:

```
1. Main step
   1.1. Sub-step
   1.2. Sub-step
2. if condition then
      Do this
   else
      Do that
```

Flowchart Symbols:

- Oval = Start/Stop
- Parallelogram = Input/Output
- Rectangle = Process/Action
- Diamond = Decision (Yes/No)

Decomposition Steps:

1. Understand the goal
2. List main tasks
3. Break down each task
4. Order the pieces
5. Connect everything

6.1 Reflection

Think about your learning this week:

1. Write pseudocode for a program that converts kilometers to miles
2. What's the benefit of drawing a flowchart before coding?
3. Decompose this problem: "Create a number guessing game where the computer picks a random number"
4. Which planning tool (pseudocode, flowchart, or decomposition) do you find most helpful? Why?

Next week, we'll use these planning skills as we learn about `if` statements and making decisions in our code!