مكتب

Quick Reference Notes

Week 7: Making Decisions with `if` Statements

7th Grade Computer Science

1 Introduction

1.1 This Week's Big Question

How can we make a computer interact intelligently with us? This week, we'll teach Python to make decisions - responding differently based on different situations. Just like you decide whether to wear a jacket based on the weather, Python can make choices too!

Prerequisites

Before starting this week, you should be comfortable with:

- Creating and using variables (Week 3)
- Boolean expressions and comparison operators (Week 5)
- Drawing flowcharts with decision diamonds (Week 6)

1.2 What You Already Know

You've learned to create boolean expressions like `age >= 13` that evaluate to `True` or `False`. You know how to compare values using operators like `>`, `<`, and `==`. You've also used if blocks in Scratch and drawn decision diamonds in flowcharts. Now we'll connect all these skills to make Python actually do something based on those comparisons!

1.3 What You'll Be Able to Do

By the end of this week, you'll:

- Write programs that respond differently to different inputs
- Make Python "think" and make decisions
- Trace through conditional code step-by-step
- Create password checkers and age verifiers

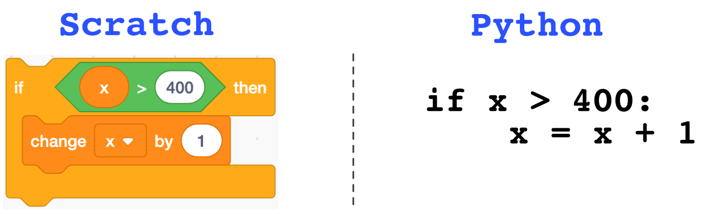
2 VIDEO 1: Introduction to if Statements

2.1 Making Programs Smart

So far, our programs run every line of code from top to bottom. But real programs need to make choices! An `if` statement lets Python execute code only when a condition is `True`.

2.2 From Scratch Blocks to Python

Remember the `if` block in Scratch? Python's `if` statement does the same thing!



```
if x > 400:
    x = x + 1
```

Aspect	Scratch	Python
Visual Design	Puzzle-piece shapes	Text-based code
Structure	Automatic *(blocks fit together)*	Manual *(you are responsible for everything)*
Logic	Checks condition → runs code	Checks condition → runs code

The core idea is the same - both check a condition and run code if it's True!

2.3 Your First if Statement

Let's see the magic in action:

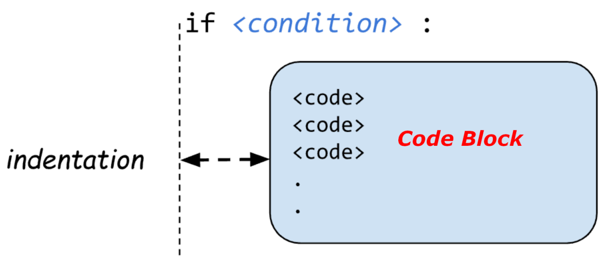
```python
age = 15

if age >= 13:
    print("You can create a social media account!")
    print("Remember to stay safe online!")

print("Thanks for using our app!")
```

When you run this code: - Python checks if `age >= 13` is `True` *(it is, since 15 >= 13)*
- Since it's `True`, Python runs the indented code
- Then continues with the rest of the program

2.4 Anatomy of an if Statement



Rule

`if` Statement Pattern:

```python
if <boolean expression>:
    # This code runs if boolean expression is True
    # All indented lines are part of the if
# This line always runs (not indented)
```

Remember you have to put a colon (:) at the end of an `if` statement. You also have to indent code that is supposed to be inside the `if` statement.

2.5 When the Condition is False

What happens when the condition is False?

```python
temperature = 10

if temperature > 25:
    print("It's hot outside!")
    print("Don't forget water!")

print("Have a great day!")  # This always prints
```

Output:

```
Have a great day!
```

The indented lines were skipped because `10 > 25` is `False`. Python never goes inside the if because the condition wasn't met.

Landmark Moment

The `if` statement is your first tool for making interactive, intelligent programs. This is where your code starts to "think"!

Quick Check

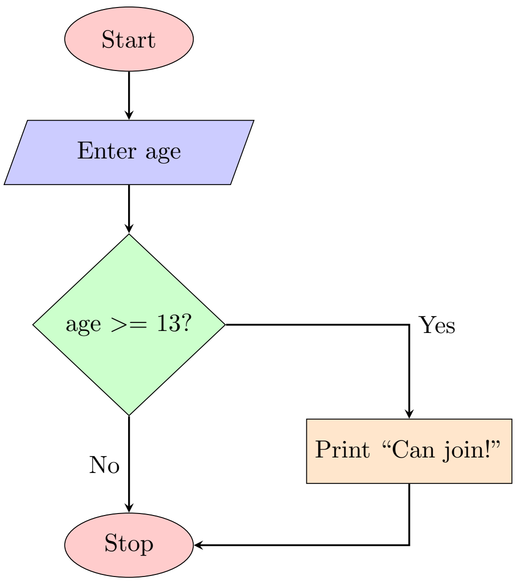
What will this code print if `score = 75`?

```python
if score > 80:
    print("Excellent!")
print("Test complete")
```

3 VIDEO 2: From Flowcharts to Python

3.1 Flowchart Decisions Become if Statements

Remember the diamond shapes in flowcharts from Week 6? Those decision points translate directly into Python if statements!



This flowchart becomes Python code:

```python
age = int(input("Enter age: "))

if age >= 13:
    print("Can join!")

# Program continues here (where paths merge)
```

Rule

Flowchart to Python Translation:

- Diamond (decision) → `if` statement
- Condition in diamond → condition after `if`
- "Yes" path → indented code block
- "No" path → else block *(lecture 8)*
- Paths merging → back to normal flow

3.2 Building Conditions

A condition is any expression that evaluates to True or False. We use the comparison operators from Week 5:

```python
password = "python123"

# Check exact match
if password == "python123":
    print("Access granted!")

# Check if NOT equal
if password != "12345":
    print("Good - you avoided a weak password!")

# Numeric comparisons
score = 85
if score >= 60:
    print("You passed!")
```

4 VIDEO 3: Simple Conditions and Indentation

4.1 Common Condition Patterns

Here are patterns you'll use often:

```python
# Pattern 1: Checking boundaries
age = int(input("Enter your age: "))
if age < 18:
    print("You're a minor")

# Pattern 2: Checking for specific values
day = input("What day is it? ")
if day == "Friday":
    print("TGIF!")

```

```python
# Pattern 3: Using calculations in conditions
price = 150
budget = 200
if price <= budget:
    print("You can afford it!")
```

Tip

When planning programs with decisions, draw a flowchart first! The diamonds become your `if` statements.

4.2 Indentation Matters!

Python uses indentation to know what's inside the if statement:

Common Error

Indentation is Python's grammar!

Wrong - No indentation

```python
if score > 90:
print("A+")  # IndentationError!
```

Wrong - Inconsistent indentation

```python
if score > 90:
    print("Great job!")
  print("A+")  # IndentationError!
```

Correct - Consistent 4 spaces

```python
if score > 90:
    print("Great job!")
    print("A+")
```

4.3 The Tabs vs. Spaces Problem

One of the trickiest indentation issues happens when you mix tabs and spaces. They might look the same, but Python sees them differently!

Common Error

Never mix tabs and spaces!

This looks fine but causes errors:

```python
if temperature > 30:
->  print("Hot day!")       # This line uses a TAB
    print("Stay cool!")     # This line uses 4 SPACES

# Error: TabError: inconsistent use of tabs and spaces
```

The → symbol shows where a tab character is hiding!

Tip

Avoiding Tab/Space Problems:

- Configure your editor to show whitespace characters
- Set your editor to convert tabs to spaces automatically
- Always use 4 spaces (Python's standard)
- If you get a TabError, check each line's indentation
- Copy-pasting code often causes tab/space mixing!

Quick Check

Fix the indentation error:

```python
temperature = 35
if temperature > 30:
print("It's very hot!")
print("Stay hydrated!")
```

5 VIDEO 4: Analyzing Situations and Tracing Code

5.1 Thinking Before Coding

Before writing any `if` statement, analyze the situation.

Example Scenario: School Canteen System

The canteen wants a program that:

- Gives a 10% discount if it's Friday
- Calculates the final price after any discount

Let's analyze what decisions we need:

1. **When to give discount?** → **if** day equals "Friday"
2. **How much discount?** → 10% of the original cost
3. **What's the final price?** → Original cost minus discount

This thinking process helps you identify where you need `if` statements!

5.2 Tracing Code Step-by-Step

Let's trace through this canteen program line by line:

```python
day = "Friday"
cost = 50
if day == "Friday":
    discount = cost * 0.1
    cost = cost - discount
    print("Friday discount applied!")
print("Final cost:", cost)
```

Line	Variable Values	Output	Notes
1	day = “Friday”		day gets “Friday”
2	day = “Friday”, cost = 50		cost gets 50
3	day = “Friday”, cost = 50		Check: “Friday” == “Friday”? `True`
4	day = “Friday”, cost = 50, discount = 5		discount calculated
5	day = “Friday”, cost = 45, discount = 5		cost updated
6	day = “Friday”, cost = 45	`"Friday discount applied!"`	
7	day = “Friday”, cost = 45	`"Final cost:  45"`	

Table 1: Trace table of canteen program execution

Rule

Tracing Steps:

1. Track all variable values
2. Note when conditions are checked
3. Show `True`/`False` results
4. Follow execution path
5. Record all output

Quick Check

Trace this code when temperature = 15:

```python
message = "Weather update: "
if temperature < 20:
    message = message + "Cold day!"
print(message)
```

Show complete trace table. What will print?

6 VIDEO 5: Using Comparison Operators in Conditions

```python
username = input("Enter username: ")

# Check exact match (case sensitive!)
if username == "admin":
    print("Welcome, administrator!")

# Alphabetical comparison
if username < "m":
    print("Your username is in the first half of the alphabet")
```

Common Error

String comparisons are case-sensitive!

```python
name = "Ahmed"
if name == "ahmed":  # This is False!
    print("Welcome")
```

Common Error

Common Mistakes to Avoid

Mistake 1: Using = instead of ==

```python
password = "secret"
if password = "secret":      # SyntaxError!
    print("Matched")
if password == "secret":     # Correct
    print("Matched")
```

Mistake 2: Forgetting the colon

```python
if age > 18                  # SyntaxError!
    print("Adult")
if age > 18:                 # Correct
    print("Adult")
```

Rule

if Statement Checklist:

- ✓ Used `if` keyword
- ✓ Condition evaluates to True/False
- ✓ Added colon (:) after condition
- ✓ Indented the code block (4 spaces)
- ✓ Used == for comparison (not =)

6.1 Tracing Through Code

Let's trace through this step-by-step:

```python
age = 15
fee = 100
student_card = "no"
if age < 12:
    fee = fee / 2
if age < 18:
    student_card = "yes"
    fee = fee - 10
print("Fee:", fee)
print("Student card:", student_card)
```

Line	Variable Values	Output	Notes
1	age = 15		age initialized
2	age = 15, fee = 100		fee initialized
3	age = 15, fee = 100, student_card = "no"		card status set
4	age = 15, fee = 100, student_card = "no"		Check: 15 < 12? False
5	age = 15, fee = 100, student_card = "no"		condition false, skip this line
6	age = 15, fee = 100, student_card = "no"		Check: 15 < 18? True
7	age = 15, fee = 100, student_card = "yes"		card updated
8	age = 15, fee = 90, student_card = "yes"		fee reduced by 10
9	age = 15, fee = 90, student_card = "yes"	`"Fee: 90"`	
10	age = 15, fee = 90, student_card = "yes"	`"Student card: yes"`	

Table 2: Trace table showing two independent if statements with different effects

Quick Check

What's the output if `x = 5` and `y = 10`?

```python
if x * 2 == y:
    print("Double match!")
if x + y > 12:
    print("Big sum!")
print("Done")
```

7 VIDEO 6: Summary

7.1 The Power of Decision Making

This week, we gave Python the ability to think and make choices:

- **if statements** let code run conditionally
- **Conditions** are boolean expressions that guide decisions
- **Indentation** tells Python what's inside the if block
- **Comparison operators** create the conditions we need

8 Quick Reference

Quick Reference

Week 7 if Statement Patterns:

```
# Basic structure
if condition:
    # indented code runs if True

# Common conditions
if value == target:         # Exact match
if value != wrong:          # Not equal
if number > minimum:        # Greater than
if score >= passing:        # Greater or equal
if age < limit:             # Less than
if price <= budget:         # Less or equal

# Remember:
# -  Colon after condition
# -  4 spaces for indent
# -  == compares,  = assigns
```

9 Reflection

Think about your learning journey this week:

1. How did your experience with Scratch if blocks help you understand Python if statements? What felt familiar? What was new?
2. Python uses indentation to show what's inside an if statement. How is this different from other ways of grouping code? Do you find it easier or harder to read?
3. When you got an IndentationError today, how did you figure out what was wrong? What strategy worked for you?
4. Think of a time this week when you made a decision (like what to wear or what to eat). How would you express that decision as an if statement?
5. What was the most challenging part of learning if statements? What finally made it "click" for you?

Next week, we'll expand our decision-making powers with `else` and `elif` to handle multiple possibilities!