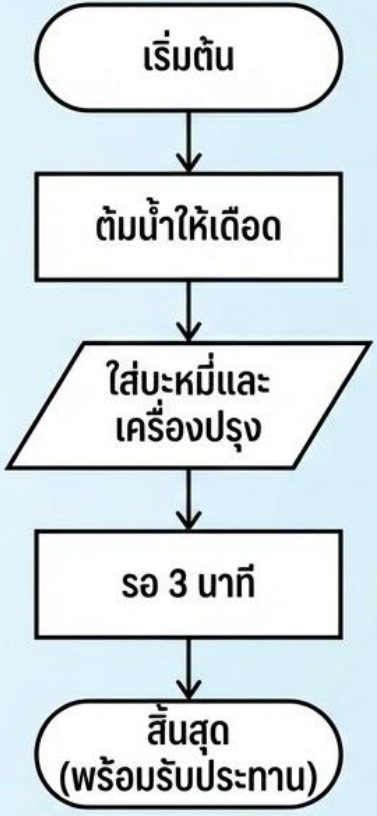
คู่มือการเขียน Flowchart เบื้องต้น

ทำความเข้าใจและสร้างผังงานอย่างมีประสิทธิภาพ

Flowchart คืออะไร?

- แผนภาพแสดงขั้นตอนของกระบวนการหรือระบบ
- ใช้สัญลักษณ์มาตรฐานสื่อความหมาย
- ช่วยให้เห็นภาพรวมและเข้าใจง่าย

ตัวอย่าง: การต้มบะหมี่กึ่งสำเร็จรูป



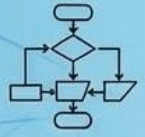
สัญลักษณ์พื้นฐานที่ควรรู้

สัญลักษณ์	ชื่อ	ความหมาย
	Start/End (จุดเริ่มต้น/สิ้นสุด)	จุดเริ่มต้นหรือสิ้นสุดของกระบวนการ
	Process (การทำงาน)	การทำงาน การประมวลผล หรือกิจกรรม
	Decision (การตัดสินใจ)	จุดที่ต้องเลือกหรือตรวจสอบเงื่อนไข (ใช่/ไม่ใช่)
	Input/Output (รับ/แสดงข้อมูล)	การนำเข้าข้อมูลหรือแสดงผลลัพธ์
	Flowline (ทิศทาง)	แสดงลำดับและทิศทางการไหลของงาน

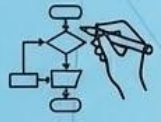
ขั้นตอนการสร้าง Flowchart

1. **กำหนดเป้าหมายและขอบเขต** (Define Goal & Scope)
   ระบุสิ่งที่ต้องการทำและจุดเริ่มต้น-สิ้นสุด

2. **ระบุขั้นตอนหลักตามลำดับ** (List Steps in Order)
   เขียนขั้นตอนการทำงานทั้งหมดออกมา



3. **เลือกสัญลักษณ์ที่เหมาะสม** (Choose Symbols)
   ใช้สัญลักษณ์มาตรฐานแทนแต่ละขั้นตอน

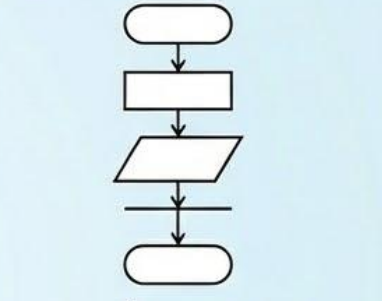


4. **วาดผังงานและเชื่อมโยง** (Draw & Connect)
   วาดและเชื่อมโยงด้วยลูกศรแสดงทิศทาง

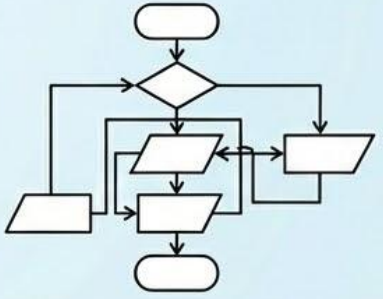
5. **ตรวจสอบและแก้ไข** (Review & Revise) ตรวจสอบความถูกต้องและปรับปรุงใหยให้ชัดเจน

เคล็ดลับสำหรับการเขียน Flowchart ที่ดี

- ใช้สัญลักษณ์ให้ถูกต้องและสื่อความหมาย
- เชื่อนถั้งเขน
- เขียนทิศทางจากบนลงล่างหรือซ้ายไปขวา
- ทำให้ผังงานดูสะอาดตา ไม่ซับซ้อนเกินไป
- ใช้คำกระชับและชัดเจนในแต่ละขั้นตอน



ดี (Good)



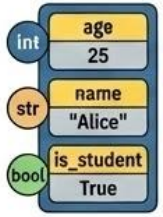
ควรปรับปรุง (Needs Improvement)

พื้นฐานการเขียนโปรแกรมภาษาไพธอน (Python Programming Basics)

เพื่อเป็นรากฐานในการเรียนรู้ต่อยอด (For Building a Strong Foundation)

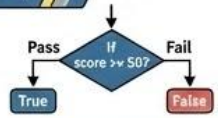
1. ตัวแปรและชนิดข้อมูล (Variables & Data Types)

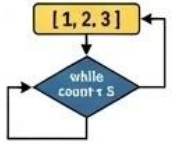
- **ตัวแปร (Variable):** กล่องเก็บข้อมูล ตั้งชื่อให้สื่อความหมาย
- **ชนิดข้อมูล (Data Types):**
  - **int:** จำนวนเต็ม (e.g., 10, -5)
  - **float:** ทศนิยม (e.g., 3.14, -0.5)
  - **str:** ข้อความ (e.g., 'Hello', 'Python')
  - **bool:** ค่าความจริง (True, False)



2. โครงสร้างควบคุม (Control Structures)

- **เงื่อนไข (Conditions):**
  - **if, elif, else**: ตรวจสอบเงื่อนไขเพื่อตัดสินใจ
- **การวนซ้ำ (Loops):**
  - **for loop**: ทำซ้ำตามจำนวนรอบที่กำหนด หรือตามสมาชิกในลิสต์
  - **while loop**: ทำซ้ำตราบเท่าที่เงื่อนไขยังเป็นจริง





3. ฟังก์ชัน (Functions)

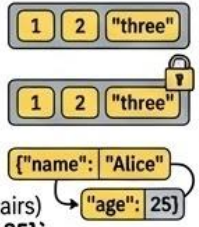
- **การสร้างฟังก์ชัน (Defining Functions):** ใช้ `def` ตามด้วยชื่อฟังก์ชัน และวงเล็บ `()`
- **การเรียกใช้ฟังก์ชัน (Calling Functions):** ใช้ชื่อฟังก์ชันและใส่อาร์กิวเมนต์ (arguments) ในวงเล็บ
- **การส่งค่ากลับ (Return Values):** ใช้ `return` เพื่อส่งผลลัพธ์ออกจากฟังก์ชัน

```
def greet(name):
    return "Hello, " + name
```

```
greet("Bob")
"Hello, Bob"
```

4. โครงสร้างข้อมูล (Data Structures)

- **ลิสต์ (List):** เก็บข้อมูลหลายค่าเรียงลำดับ แก้ไขได้ `[1, 2, "three"]`
- **ทูเพิล (Tuple):** เก็บข้อมูลหลายค่าเรียงลำดับ แก้ไขไม่ได้ `(1, 2, "three")`
- **ดิกชันนารี (Dictionary):** เก็บข้อมูลแบบคู่คีย์-ค่า (key-value pairs) แก้ไขได้ `{"name": "Alice", "age": 25}`



5. การจัดการข้อผิดพลาด (Error Handling)

- **try-except**: ดักจับและจัดการกับข้อผิดพลาดที่อาจเกิดขึ้นระหว่างการทำงาน
- ป้องกันโปรแกรมหยุดทำงานกะทันหัน

```
try:
    code
    cause an error
    ...
```

```
except:
    "Grace message"
    ...
```

การใช้งานฟังก์ชัน `print()` ใน Python
(การแสดงผลข้อมูลออกทางหน้าจอ)

โครงสร้างและข้อควรจำ (Structure & Notes)

```
print(ข้อมูล1, ข้อมูล2, ...)
```

- ใช้สำหรับแสดงผลข้อมูล (ข้อความ, ตัวเลข, ตัวแปร)
- สามารถแสดงข้อมูลได้หลายตัวพร้อมกัน โดยใช้จุลภาค (,) คั่น
- เมื่อใช้จุลภาค (,) จะมีช่องว่างคั่นระหว่างข้อมูลให้อัตโนมัติ

ตัวอย่างการใช้งาน (Code Examples)

```
# 1. แสดงข้อความและตัวเลข
print("สวัสดี Python", 2024)
--> Output: สวัสดี Python 2024
```

```
# 2. การใช้ตัวแปร
name = "สมหญิง"
score = 95.5
print("ชื่อ:", name, "คะแนน:", score)
--> Output: ชื่อ: สมหญิง คะแนน: 95.5
```

```
# 3. การเชื่อมข้อความ (String Concatenation)
greeting = "สวัสดี"
print(greeting + " " + name)
--> Output: สวัสดี สมหญิง
```

การใช้งานฟังก์ชัน `input()` ใน Python

(การรับข้อมูลจากผู้ใช้)

คำอธิบาย

- ใช้สำหรับรับข้อมูลจากผู้ใช้งานผ่านคีย์บอร์ด
- แสดงข้อความแจ้งเตือน (prompt) ให้ผู้ใช้ทราบก่อนรับค่า
- หยุดการทำงานชั่วคราวเพื่อรอให้ผู้ใช้พิมพ์และกด Enter
- ข้อควรจำ: ข้อมูลที่ได้รับจะเป็นชนิดข้อความ (String) เสมอ

ตัวอย่างโค้ด

```
# รับชื่อจากผู้ใช้
name = input("กรุณาป้อนชื่อของคุณ: ")

# แสดงผลลัพธ์
print("สวัสดีคุณ " + name)
```

ผลลัพธ์การรัน (Output)

```
กรุณาป้อนชื่อของคุณ: สมชาย
สวัสดีคุณ สมชาย
```

ยินดีต้อนรับสู่โลกของ Python

ชื่อวิชา / ชื่อผู้สอน

Ice Breaking

- ใครคิดว่าการเขียนโปรแกรมเป็นเรื่องของเด็ก IT บ้าง?
- ใครติดใช้ Excel ทำงานทุกอย่างบ้าง?

เป้าหมายวันนี้

เราไม่ได้จะมาสร้าง Facebook ใหม่ แต่เราจะเรียนรู้วิธี “สั่งงานคอมพิวเตอร์ให้ทำงานแทนเรา” ในมุมมองนักบริหาร

ทำไมเด็กบริหารต้องเรียน Python?

Why Python for Business?

เปรียบเทียบ: Excel vs. Python

1 ล้านบรรทัด (1 Million Rows)	1 ล้านบรรทัด (1 Million Rows)
ค้าง (Crashes)	สบายมาก (Smooth/Easy)

ตัวอย่างงานจริง (Real-world Examples)

ดึงข้อมูลลูกค้าจาก Shopee/Lazada มาวิเคราะห์อัตโนมัติ (Data Scraping)

ตอบแชทลูกค้าอัตโนมัติ (Chatbot)

ทำนายยอดขายเดือนหน้า (Data Science)

Key takeaway: Python คือภาษาที่ 'อ่านง่ายเหมือนภาษาอังกฤษ' และนิยมที่สุดในโลกธุรกิจตอนนี้.

คอมพิวเตอร์คิดอย่างไร? (Concept)

The Concept: Input -> Process -> Output

Input: วัตถุดิบ
(เช่น ราคาต้นทุน, จำนวนสินค้า)

Process: กรรมวิธี/สูตรคำนวณ
(เช่น เอา ราคา × จำนวน)

Output: ผลลัพธ์ที่ต้องการ
(เช่น ยอดขายรวม)

เปรียบเทียบกับการทำอาหาร (Cooking Analogy)

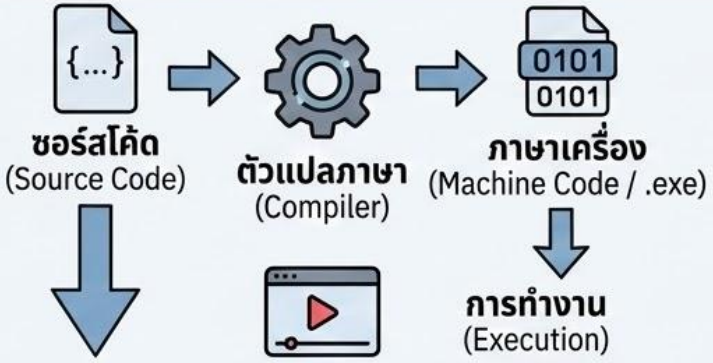
วัตถุดิบ

ปรุง

อาหารจานเสร็จ

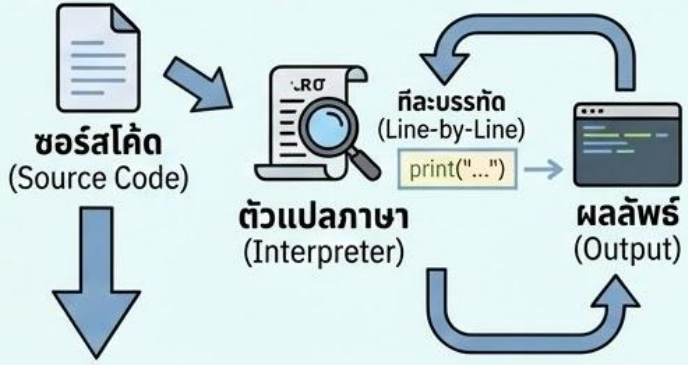
ประเภทและการทำงานของตัวแปรภาษาโปรแกรม
(Types and Working Patterns of Programming Language Translators)

1. คอมไพเลอร์ (Compiler)



- **แปลทั้งหมดในครั้งเดียว** (Translates all at once)
- **สร้างไฟล์พร้อมทำงาน** (Creates ready-to-run file)
- **ทำงานได้เร็ว** (Fast execution)
- **แก้ไขยาก** (Hard to debug)

2. อินเทอร์พรีเตอร์ (Interpreter)



- **แปลและทำทีละบรรทัด** (Translates & runs line-by-line)
- **ไม่สร้างไฟล์ใหม่** (No new file created)
- **ทำงานช้ากว่า** (Slower execution)
- **แก้ไขง่าย** (Easy to debug)

ตัวอย่างแบบผสม: Python บน Google Colab

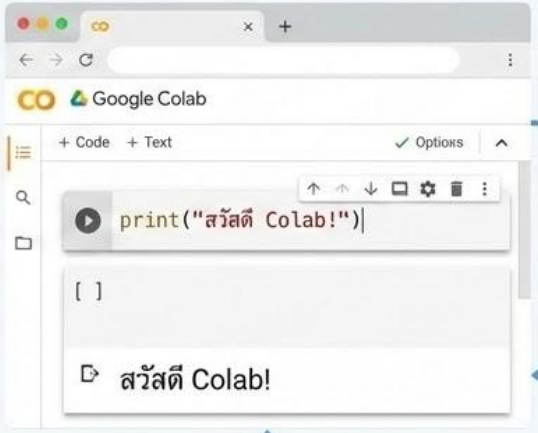
Python ใช้การทำงานแบบผสม แต่ใน Colab ผู้ใช้จะเห็นผลลัพธ์เหมือนอินเทอร์พรีเตอร์

ขั้นตอนการทำงานของ Python บน Google Colab

ฝั่งผู้ใช้
(User Side/Browser)

Google Cloud Server
(Virtual Machine)

1. เขียนโค้ดในเบราว์เซอร์



2. ส่งโค้ดไปยังเซิร์ฟเวอร์

3. เซิร์ฟเวอร์ประมวลผล

Python Kernel
ประมวลผลโค้ด
(Execute Code)

ผลลัพธ์
(Output)

4. ส่งผลลัพธ์กลับ

5. แสดงผลลัพธ์ในเบราว์เซอร์

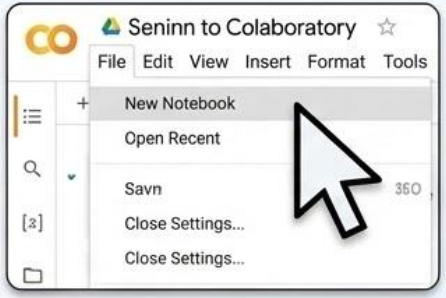
เริ่มต้นใช้งาน Google Colab

Tools: Google Colab (เขียนโค้ดบน Cloud)

1. เปิด Browser เข้า Google Colab (ไม่ต้องลงโปรแกรม)

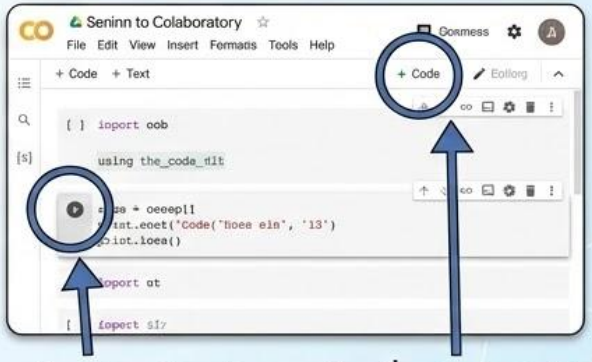
Browser ทำงานบน Cloud
(No Installation Required)

2. วิธีสร้าง New Notebook



คลิก File > New Notebook

3. ปุ่มพื้นฐาน (Basic Buttons)



ปุ่ม Play (Run):
กดเพื่อรันโค้ด

ปุ่มเพิ่ม Code Cell:
กดเพื่อเพิ่มเซลล์โค้ด

Note: เหมือน Google Docs แต่พิมพ์และรันโค้ดได้
(Like Google Docs, but for running code)

คำสั่งแรกในชีวิต (Print)

Hello World & Print Function Code

คำอธิบาย (Explanation)

print() คือคำสั่งให้คอมพิวเตอร์ ‘พูด’ หรือแสดงผลออกมาทางหน้าจอ

“ ”

ข้อความต้องอยู่ภายใต้เครื่องหมายคำพูด "" (Double Quote) หรือ '' (Single Quote)

Code ตัวอย่าง (Example): Python

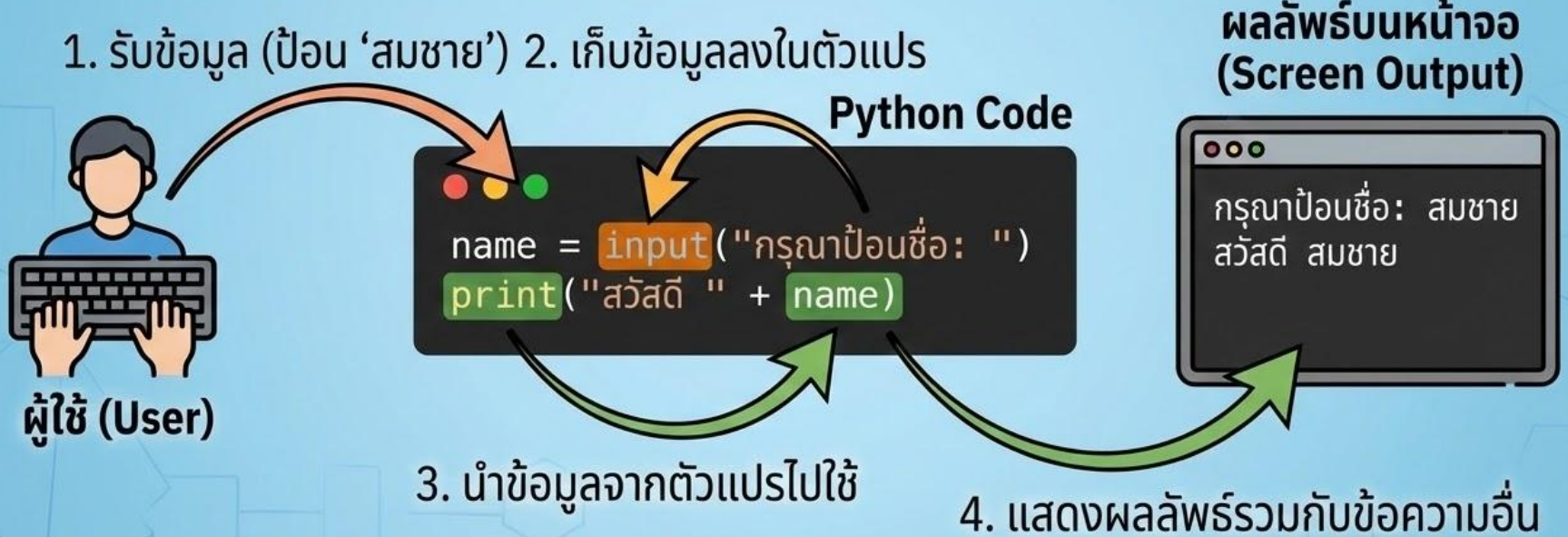
```python
print("Hello World")
print("สวัสดีชาวบริหารธุรกิจ")
```

ผลลัพธ์ (Output)

```
Hello World
สวัสดีชาวบริหารธุรกิจ
```

Challenge เล็กๆ! ให้พิมพ์ชื่อเล่นและรหัสนักศึกษาตัวเองออกมา (เช่น print(‘ชื่อเล่น: ... รหัส: ...’))

ภาพรวมการทำงาน: การรับและแสดงผลข้อมูล
(input & print flow)



กล่องเก็บข้อมูล (Variables)

Variables (ตัวแปร) Code ตัวอย่าง:

คำอธิบาย (Explanation)

เปรียบเทียบ ตัวแปร เหมือน "กล่อง" ที่เราแปะป้ายชื่อไว้

product_name คือป้ายชื่อกล่อง, “Iphone 15” คือของที่ใส่ลงไป

```
product_name = "Iphone 15"
price = 30000
quantity = 2

print(product_name)
print(price)
```

กฎเหล็ก (Rules):

- ชื่อตัวแปรห้ามเว้นวรรค (ใช้ _ แทน)
- ห้ามขึ้นต้นด้วยตัวเลข

ชนิดของข้อมูล (Data Types)

Data Types ที่ต้องรู้

คำอธิบาย: สอนแค่ 3 อย่างที่ใช้บ่อยในธุรกิจ:

String (str): ข้อความ
(ต้องมี "" ครอบ) เช่น "Somsak", "BKK"

Integer (int): จำนวนเต็ม
(นับชิ้น, นับคน) เช่น 10, 500

Float (float): ทศนิยม
(เงิน, ภาษี, เกรด) เช่น 99.50, 7.0

ระวัง: "10" (ข้อความ) ไม่เท่ากับ 10 (ตัวเลข)

เครื่องคิดเลขทางธุรกิจ (Math Operations)

Basic Math & Calculation

Code ตัวอย่าง:

```
cost = 100
price = 150
profit = price - cost

print("กำไรต่อชิ้นคือ:", profit)
```

คำอธิบาย (Explanation)

สอนเครื่องหมาย: +, -, * (คูณ), / (หาร)

เน้นย้ำ: คอมพิวเตอร์คำนวณจากขวาไปซ้าย (เอาผลลัพธ์ขวา ไปเก็บในตัวแปรซ้าย)

Challenge: ให้ลองคำนวณ $Total = Price \ \times Quantity$

การรับค่าจากผู้ใช้ (Input)

Input Function (Interactive Program)

Code ตัวอย่าง:

```
name = input("กรุณากรอกชื่อของคุณ: ")
print("สวัสดีคุณ", name)
```

input() คือการเปิดช่องให้ User พิมพ์ข้อมูลเข้าไป

สำคัญมาก: ค่าที่ได้จาก **input()** จะเป็น **ข้อความ** (String) เสมอ แม้จะพิมพ์เลข

ถ้าจะเอาไปคำนวณ ต้องแปลงร่างก่อน เช่น int(input(...))

ชนิดของตัวแปรใน Python (Python Data Types)

เนื้อหาเสริม: พื้นฐานสำคัญสำหรับการเขียนโปรแกรมและ Data Analytics

กลุ่มที่ 1: ตัวเลข (Numeric Types)

ใช้สำหรับการคำนวณทางคณิตศาสตร์ สถิติ และการเงิน

Integer (int): จำนวนเต็ม (ไม่มีทศนิยม)

- จำนวนพนักงาน (300)
- ปียอดขาย (2024)
- จำนวนสินค้าคงคลัง (-5)

Floating Point (float):

จำนวนจริง (มีทศนิยม)

- ราคาสินค้า (99.50)
- อัตราดอกเบี้ย (0.07)
- เกรดเฉลี่ย (3.56)

ชนิดของตัวแปรใน Python (ต่อ) - ข้อความ

เนื้อหาเสริม: พื้นฐานสำคัญสำหรับการเขียนโปรแกรมและ Data Analytics

กลุ่มที่ 2: ข้อความ (Sequence Types)

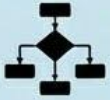
ใช้สำหรับเก็บชื่อ รายละเอียด หรือรหัสที่ไม่ใช้คำนวณ

String (str): ชุดตัวอักษร ต้องอยู่ภายใต้เครื่องหมาย `""` หรือ `''`

- ชื่อลูกค้า (`"Somsak"`)
- รหัสไปรษณีย์ (`"10110"`)
- เบอร์โทร (`"0812345678"`)

ชนิดของตัวแปรใน Python (ต่อ) - ตรรกะ

เนื้อหาเสริม: พื้นฐานสำคัญสำหรับการเขียนโปรแกรมและ Data Analytics



กลุ่มที่ 3: ตรรกะ (Boolean Type)

ใช้สำหรับการตัดสินใจ (Decision Making) ว่าจริงหรือเท็จ

Boolean (bool): มีแค่ 2 ค่าเท่านั้น คือ **True** (จริง) หรือ **False** (เท็จ)

- สถานะสมาชิก (`is_member = True`)
- สินค้าหมดสต็อกหรือไม่ (`is_out_of_stock = False`)

ชนิดของตัวแปรใน Python (ต่อ) – กลุ่มข้อมูล

เนื้อหาเสริม: พื้นฐานสำคัญสำหรับการเขียนโปรแกรมและ Data Analytics

กลุ่มที่ 4: กลุ่มข้อมูล (Collection Types)
(อาจแค่เกริ่นไว้)

List (list): เก็บข้อมูลหลายๆ ตัวในตัวแปรเดียว (เหมือนคอลัมน์ใน Excel)

- `prices = [100, 250, 300]`

	A	B
1	prices	
2	100	
3	250	
4	300	

ทำไม "10" (Text) กับ 10 (Number) ถึงไม่เหมือนกัน?

จุดที่มือใหม่สับสนที่สุด และสาเหตุที่ทำให้โปรแกรม Error บ่อยที่สุด

เลข 10 (int):

เหมือน ‘เงินเหรียญสิบบาท’
-> คุณเอาไปบวกกับเหรียญอื่นได้ (ซื้อของได้)

ข้อความ "10" (str):

เหมือน ‘รูปภาพวาดเหรียญสิบบาท’ บนกระดาษ -> มันดูเหมือนเงิน แต่คุณเอาไปซื้อของไม่ได้

การแปลงชนิดตัวแปร (Type Casting)

เมื่อข้อมูลมาผิดประเภท เราต้องเปลี่ยนร่างมันก่อน (Data Cleaning)

int(): แปลงเป็นจำนวนเต็ม

int("500") -> 500
int(99.99) -> 99
(ตัดทศนิยมทิ้งดื้อๆ ไม่ปัดเศษ)

float(): แปลงเป็นทศนิยม

float("10") -> 10.0
float(5) -> 5.0

str(): แปลงเป็นข้อความ

str(2024) -> "2024"
(ใช้เมื่อต้องการแทรกตัวเลขเข้าไปในประโยคยาวๆ)

ตัวอย่าง Code: Python

```
cost = "500"          # มาเป็น String (อาจมาจาก input)
# total = cost + 50  <-- แบบนี้จะ Error!

cost = int(cost)     # แปลงร่างเป็นตัวเลขก่อน
total = cost + 50    # คำนวณได้แล้ว = 550
```

ตัวอย่างการแปลงชนิดข้อมูล (Type Casting) - ส่วนที่ 1

ฟังก์ชันพื้นฐานในการเปลี่ยนประเภทข้อมูล (จำเป็นสำหรับ Data Cleaning)

int(): แปลงเป็นจำนวนเต็ม (Integer)

```
x_str = "150"
x_int = int(x_str)
print(x_int)   # ผลลัพธ์: 150 (type: int)

หมายเหตุ: ถ้าแปลงทศนิยม จะตัดส่วนทศนิยมทิ้ง
print(int(99.99))  # ผลลัพธ์: 99
```

float(): แปลงเป็นทศนิยม (Float)

```
y_str = "3.14159"
y_float = float(y_str)
print(y_float)   # ผลลัพธ์: 3.14159 (type: float)

z_int = 50
print(float(z_int))
# ผลลัพธ์: 50.0
```

str(): แปลงเป็นข้อความ (String)

```
age = 25
msg = "ฉันอายุ " + str(age) + " ปี"
print(msg)  # ผลลัพธ์: "ฉันอายุ 25 ปี"

pi = 3.14
print("ค่า PI คือ " + str(pi))
# ผลลัพธ์: "ค่า PI คือ 3.14"
```

ตัวอย่างการแปลงชนิดข้อมูล (Type Casting) - ส่วนที่ 2

การใช้งานจริง: แก้ปัญหาเมื่อรับข้อมูล Input (ข้อมูลที่ได้มักเป็น String เสมอ)

⚠ ปัญหา: ไม่ได้แปลงชนิดข้อมูล (เกิด Error)

```
# จำลองการรับ input จากผู้ใช้ (ได้เป็น String)
price_input = "500"
tax_rate = 0.07

# พยายามคำนวณภาษี (String * Float)
tax = price_input * tax_rate
print(tax)
```

```
TypeError: can't multiply sequence by
non-int of type 'float'
```

คอมพิวเตอร์ไม่สามารถนำข้อความไปคูณกับตัวเลขทศนิยมได้

✅ วิธีแก้ปัญหา: ใช้ Type Casting แปลงก่อนคำนวณ

```
# จำลองการรับ input จากผู้ใช้
price_input = "500"
tax_rate = 0.07

# แปลง String ให้เป็น Float ก่อนนำไปคำนวณ
price_float = float(price_input)
tax = price_float * tax_rate

print("ภาษีที่ต้องจ่าย: " + str(tax) + " บาท")
```

```
ภาษีที่ต้องจ่าย: 35.0 บาท
```

แปลง input เป็นตัวเลขก่อน แล้วจึงนำไปคำนวณ และแปลงผลลัพธ์กลับเป็น String เพื่อแสดงผล

ผลลัพธ์เมื่อคอมพิก์เมื่อคอมพิวเตอร์ทำงานบบทางธุรกิจ และผลกระทบทางธุรกิจ

ผลลัพธ์เมื่อคอมพิวเตอร์ทำงาน:

แบบตัวเลข (Math): 🧮
10 + 10 -> ผลลัพธ์: 20
คอมพิวเตอร์จะมองเป็นคณิตศาสตร์

แบบข้อความ (Concatenation): 🔗
"10" + "10" -> ผลลัพธ์: "1010"
คอมพิวเตอร์จะมองเป็นการ 'เอาคำมาต่อกัน'
(เหมือน "Hello" + "World")

ในทางธุรกิจ (สำคัญมาก!) ⚠️

ถ้านักศึกษาดึงข้อมูลจาก Excel หรือ Database แล้วรหัสสินค้าคือ 100 แต่เป็น Text แล้วเผลอเอาไปบวกกับยอดขาย ผลลัพธ์จะพังทันทีครับ

4. การประกาศชนิดตัวแปรมีผลต่อหน่วยความจำ (Memory) อย่างไร?

ในภาษายุคเก่า (เช่น C, Java) ต้องจองที่ดิน แต่ Python เป็นภาษาแบบ Dynamic Typing คอมพิวเตอร์จะจัดการให้เอง

ผลกระทบต่อหน่วยความจำ: ขนาดพื้นที่ (Size) 📏

Integer: กินพื้นที่น้อย (เปรียบเหมือนเก็บเสื้อยืด 1 ตัว) 👕

String: กินพื้นที่ตามความยาวตัวอักษร (เปรียบเหมือนเก็บหนังสือ ยิ่งหนายิ่งกินที่เยอะ) 📚

Float: กินพื้นที่มากกว่า Integer เล็กน้อยเพื่อเก็บความละเอียดทศนิยม

ประสิทธิภาพในการประมวลผล (Processing Speed) ⏱️

สำคัญสำหรับ Big Data

🚀 การคำนวณ Integer เร็วที่สุดสำหรับคอมพิวเตอร์

🚀 การจัดการ String ช้าที่สุด 🐌

ตัวอย่าง: ถ้าเราเก็บรหัสนักศึกษาเป็นตัวเลข (6401001) จะประหยัดที่และค้นหาเร็วกว่าเก็บเป็นข้อความ ("6401001") หากมีข้อมูลเป็นล้านคน

ความฉลาดของ Python (Overhead) 🧠

เนื่องจาก Python ไม่ต้องประกาศประเภทตัวแปร (เช่น พิมพ์ x = 10 ได้เลย)

Python จะต้องแอบใช้หน่วยความจำส่วนหนึ่งเพื่อจดจำว่า 'x ตอนนี้เป็นตัวเลขนะ"

นี่คือเหตุผลว่าทำไม Python เขียนง่าย แต่อาจจะทำงานช้ากว่าภาษา C เล็กน้อย แต่มักไม่ใช่ปัญหาในงานบริหารธุรกิจทั่วไปครับ

ฟังก์ชัน print() : การแสดงผลลัพธ์ (Output)

คือคำสั่งพื้นฐานที่สุดที่เปรียบเสมือน "ปาก" ของโปรแกรม
ใช้สื่อสารผลลัพธ์ออกมาทางหน้าจอ

โครงสร้างและไวยากรณ์ (Syntax)

```
print(ข้อมูลที่ต้องการแสดง)
```

1.1 การแสดงข้อความปกติ (String)

ต้องอยู่ภายใต้เครื่องหมายคำพูด (Quote) เสมอ จะเป็น ' ' (Single Quote) หรือ " " (Double Quote) ก็ได้

ตัวอย่าง:

```
print("Hello Python")
```

```
print('สวัสดีครับ')
```

1.2 การแสดงตัวเลข (Number)

ใส่ตัวเลขได้เลย ห้ามใส่เครื่องหมายคำพูด
มิเช่นนั้นคอมพิวเตอร์จะมองเป็นข้อความ

ตัวอย่าง:

```
print(2024)
```

```
print(3.14)
```

1.3 การแสดงค่าจากตัวแปร (Variable)

ใส่ชื่อตัวแปรลงไปได้เลย
ห้ามใส่เครื่องหมายคำพูด

ตัวอย่าง:

```
revenue = 50000
```

```
print(revenue) # ผลลัพธ์: 50000
```

1.4 การแสดงผลผสมกัน (ข้อความ + ตัวแปร) – วิธีที่ 1

วิธีที่ 1: ใช้เครื่องหมายจุลภาค (Comma ,) คั่น
คอมพิวเตอร์จะเติมช่องว่าง (Space) ให้อัตโนมัติระหว่างคำ

ตัวอย่าง:

```
name = "Somchai"
age = 20
print("ลูกค้าชื่อ", name, "อายุ", age, "ปี")
# ผลลัพธ์: ลูกค้าชื่อ Somchai อายุ 20 ปี
```

1.4 การแสดงผลผสมกัน (ข้อความ + ตัวแปร) – วิธีที่ 2

วิธีที่ 2: f-string (Format String)
แนะนำวิธีนี้เพราะอ่านง่าย
พิมพ์ตัว f ไว้หน้าเครื่องหมายคำพูด แล้วใส่ตัวแปรไว้ในปีกกา { }

ตัวอย่าง:

```
total_price = 107
print(f"ราคารวมภาษีคือ {total_price} บาท")
# ผลลัพธ์: ราคารวมภาษีคือ 107 บาท
```

2. ฟังก์ชั่น input() : การรับค่า (Input)

คือคำสั่งที่เปรียบเสมือน "หู" ของโปรแกรม
ใช้รอรับข้อมูลที่ผู้ใช้พิมพ์ผ่านคีย์บอร์ด

โครงสร้างและไวยากรณ์ (Syntax)

```
ตัวแปร = input("ข้อความคำถาม")
```

ตัวแปร: ต้องมีตัวแปรมารอรับค่าเสมอ (ทางซ้ายมือ)
ข้อความคำถาม (Prompt): ข้อความที่อยู่ในวงเล็บ จะถูกแสดงขึ้นมาเพื่อบอกให้ผู้ใช้รู้ว่าต้องพิมพ์อะไร

กฎเหล็กของ input()

"ค่าที่ได้จากฟังก์ชัน input() จะเป็นชนิดข้อความ (String) เสมอ"
แม้ว่าผู้ใช้จะพิมพ์ตัวเลขลงไป คอมพิวเตอร์ก็จะมองว่าเป็นตัวอักษร

2.1 รับค่าข้อความทั่วไป (เช่น ชื่อ, ที่อยู่)

ตัวอย่าง:

```
customer_name = input("กรุณากรอกชื่อลูกค้า: ")
print(f"ยินดีต้อนรับคุณ {customer_name}")
```

 # ผลลัพธ์การทำงาน:
กรุณากรอกชื่อลูกค้า: **สมชาย**
ยินดีต้อนรับคุณ สมชาย

2.2 รับค่าตัวเลขเพื่อนำไปคำนวณ (การแปลงร่าง)

ต้องใช้ฟังก์ชั่นแปลงค่า (int() หรือ float()) มาครอบ input() อีกที

ตัวอย่าง:

```
# แบบจำนวนเต็ม (Integer)
amount = int(input("กรุณาระบุจำนวนสินค้า (ชิ้น): "))

# แบบทศนิยม (Float)
price = float(input("กรุณาระบุราคา (บาท): "))

total = amount * price
print(f"ยอดรวมทั้งหมด {total} บาท")
```

 # ผลลัพธ์การทำงาน:
กรุณาระบุจำนวนสินค้า (ชิ้น): 5
กรุณาระบุราคา (บาท): 20.5
ยอดรวมทั้งหมด 102.5 บาท

ตัวอย่างที่ 1: การรับค่าตัวเลขจำนวนเต็ม (int)

- โปรแกรมรับค่าอายุจากผู้ใช้ (string) แล้วแปลงเป็นจำนวนเต็ม **(int)** เพื่อคำนวณอายุในปีหน้า

```
# รับค่าอายุจากผู้ใช้
age_str = input("กรุณากรอกอายุของคุณ: ") input()

# แปลงข้อความ (string) เป็นจำนวนเต็ม (int)
age_int = int(age_str) int()

# คำนวณอายุในปีหน้า
next_year_age = age_int + 1

print("ปีหน้าคุณจะมีอายุ:", next_year_age, "ปี") print()
```

ผลลัพธ์ตัวอย่าง:

```
กรุณากรอกอายุของคุณ: 25
ปีหน้าคุณจะมีอายุ: 26 ปี
```

ตัวอย่างที่ 2: การรับค่าตัวเลขทศนิยม (float)

- โปรแกรมรับราคาสินค้าจากผู้ใช้ (string) แล้วแปลงเป็นทศนิยม (float) เพื่อคำนวณภาษีมูลค่าเพิ่ม 7% และราคารวม

```
# รับราคาสินค้าจากผู้ใช้
price_str = input("กรุณากรอกราคาสินค้า: ") input()

# แปลงข้อความ (string) เป็นทศนิยม (float)
price_float = float(price_str) float()

# คำนวณภาษี 7% และราคารวม
tax = price_float * 0.07
total_price = price_float + tax

print("ภาษีมูลค่าเพิ่ม (7%):", tax, "บาท")
print("ราคารวมทั้งสิ้น:", total_price, "บาท") print()
```

ผลลัพธ์ตัวอย่าง:

```
กรุณากรอกราคาสินค้า: 100.50
ภาษีมูลค่าเพิ่ม (7%): 7.035 บาท
ราคารวมทั้งสิ้น: 107.535 บาท
```

3. ตัวดำเนินการทางคณิตศาสตร์ (Operators) และลำดับความสำคัญ

Python สามารถทำงานเป็นเครื่องคิดเลขได้ทันที
โดยใช้สัญลักษณ์สากลทางคอมพิวเตอร์

ตารางตัวดำเนินการ (Operators)

สัญลักษณ์	ความหมาย	ตัวอย่าง	ผลลัพธ์	หมายเหตุ
+	บวก	10 + 5	15	
-	ลบ	10 - 5	5	
*	คูณ	10 * 5	50	ใช้ดอกจัน (Star)
/	หาร	10 / 2	5.0	ผลลัพธ์เป็นทศนิยมเสมอ
**	ยกกำลัง	2 ** 3	8	(2^3) ใช้ดอกจัน 2 อัน
%	หารเอาเศษ (Modulo)	10 % 3	1	(10 หาร 3 เหลือเศษ 1)

(หมายเหตุ: % มักใช้เช็คเลขคู่เลขคี่ หรือจัดรอบการทำงาน)

ลำดับความสำคัญของเครื่องหมาย (Order of Operations)

คอมพิวเตอร์ไม่ได้คำนวณจากซ้ายไปขวาเสมอไป
แต่จะทำตามกฎทางคณิตศาสตร์ PEMDAS

Parentheses ( ) : วงเล็บ (สำคัญที่สุด ทำก่อนเสมอ)
Exponents ** : เลขยกกำลัง
M/D Multiply/Divide * / : คูณและหาร (ศักดิ์เท่ากัน ทำจากซ้ายไปขวา)
A/S Add/Subtract + - : บวกและลบ (ศักดิ์เท่ากัน ทำจากซ้ายไปขวา)

ตัวอย่างความผิดพลาดถ้าไม่เข้าใจลำดับ (Trap for Beginners)

โจทย์: ต้องการหา "ค่าเฉลี่ย" ของราคาสินค้า 2 ชิ้น (100 บาท และ 200 บาท)

❌ เขียนแบบผิด:

```
average = 100 + 200 / 2
print(average) # ผลลัพธ์จะได้ 200.0 (ผิด!)
# เพราะคอมพิวเตอร์เอา 200/2 ก่อน (=100) แล้วค่อยบวก 100
```

✅ เขียนแบบถูก (ใช้วงเล็บ):

```
average = (100 + 200) / 2
print(average) # ผลลัพธ์จะได้ 150.0 (ถูกต้อง)
# คอมพิวเตอร์ทำในวงเล็บก่อน (100+200=300) แล้วค่อยหาร 2
```

การหาร 3 แบบใน Python
(3 Types of Division in Python)

การหารไม่ได้มีแค่แบบเดียวครับ แต่มีถึง 3 รูปแบบ
ซึ่งแต่ละแบบมีประโยชน์ทางธุรกิจที่แตกต่างกันอย่างชัดเจน

1. การหารปกติ (Standard Division) /

- **หลักการ:** เหมือนเครื่องคิดเลขทั่วไป คือหารแล้วได้ผลลัพธ์ออกมาเป็นทศนิยม (Float) เสมอ แม้ว่าจะหารลงตัวก็ตาม
- **ผลลัพธ์:** ได้ค่าละเอียด (มีจุดทศนิยม)
- **ตัวอย่างทางธุรกิจ:** การหาค่าเฉลี่ย, การแปลงค่าเงิน, การคำนวณภาษี

ตัวอย่าง Code:

```
print(10 / 2) # ผลลัพธ์: 5.0 (สังเกตว่ามี .0)
print(5 / 2)  # ผลลัพธ์: 2.5
```

2. การหารปัดเศษทิ้ง (Floor Division) //

- **หลักการ:** หารแล้ว "ตัดเศษทศนิยมทิ้งทั้งหมด" (เอาแค่จำนวนเต็ม) ไม่มีการปัดขึ้นใดๆ ทั้งสิ้น
- **ผลลัพธ์:** ได้ค่าเป็น จำนวนเต็ม (Integer)
- **ตัวอย่างทางธุรกิจ:** การนับจำนวนกล่องสินค้า, การจัดคนเข้าทีม (คน 2.5 คนไม่มีจริง ต้องเป็น 2 คน)

ตัวอย่าง Code:

```
print(10 // 3) # ผลลัพธ์: 3 (จริงๆ ได้ 3.333... แต่ตัด .333 ทิ้ง)
print(5 // 2)  # ผลลัพธ์: 2 (จริงๆ ได้ 2.5 แต่ตัด .5 ทิ้ง)
```

3. การหารเอาเศษ (Modulo) %

- **หลักการ:** หารแล้วไม่สนใจผลลัพธ์ แต่สนใจแค่ "เศษที่เลือ "เศษที่เหลือ"
- **ผลลัพธ์:** ได้ค่าเป็น เศษ (Integer)
- **ตัวอย่างทางธุรกิจ:** หาสินค้าที่เหลือเข้ากล่องไม่ได้ (ตกค้าง), การจัดคิว, การหาเลขคู่/เลขคี่

ตัวอย่าง Code:

```
print(10 % 3)  # ผลลัพธ์: 1 (เพราะ 10 หาร 3 ได้ 3 ครั้ง เหลือเศษ 1)
print(10 % 2)  # ผลลัพธ์: 0 (เพราะหารลงตัว เศษเป็น 0)
```

Business Case & ข้อควรระวัง (Business Case & Warning)

💡 ยกตัวอย่างให้เห็นภาพ (Business Case: Logistics) - โรงงานผลิตน้ำดื่ม

สมมติมีน้ำดื่ม 100 ขวด, ลังใส่ได้ 12 ขวด (bottles=100, capacity=12)

```
# 1. หารปกติ (/): สัดส่วน
print(bottles / capacity)  # ผลลัพธ์: 8.333...
# 2. หารปัดเศษทิ้ง (//): จำนวนลังเต็ม
print(bottles // capacity) # ผลลัพธ์: 8
# 3. หารเอาเศษ (%): เศษสต๊อก
print(bottles % capacity)  # ผลลัพธ์: 4
```

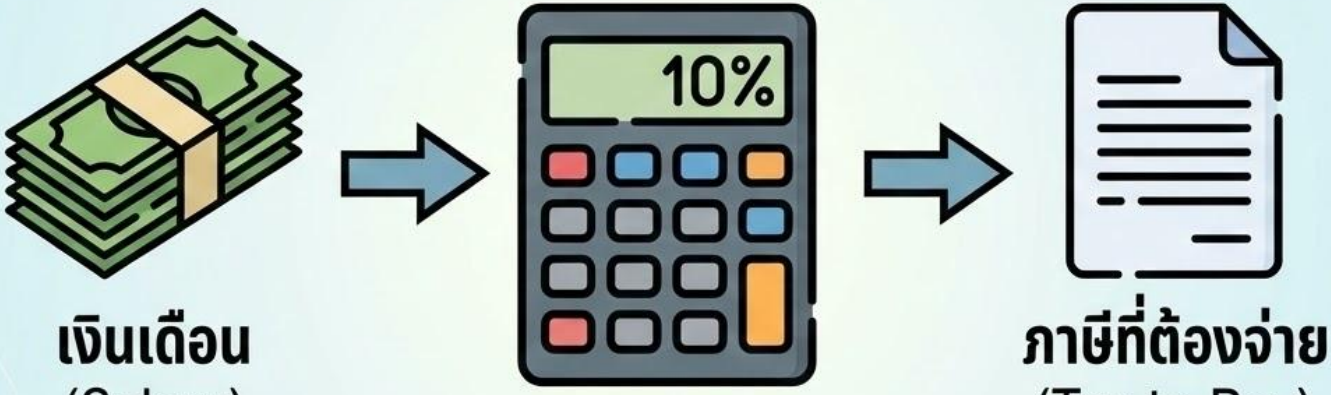
⚠️ ข้อควรระวัง (Error) - การหารด้วยศูนย์ (Division by Zero)

ห้ามเอาเลข 0 ไปเป็นตัวหารเด็ดขาด

```
print(10 / 0)   # ผลลัพธ์: ZeroDivisionError
```

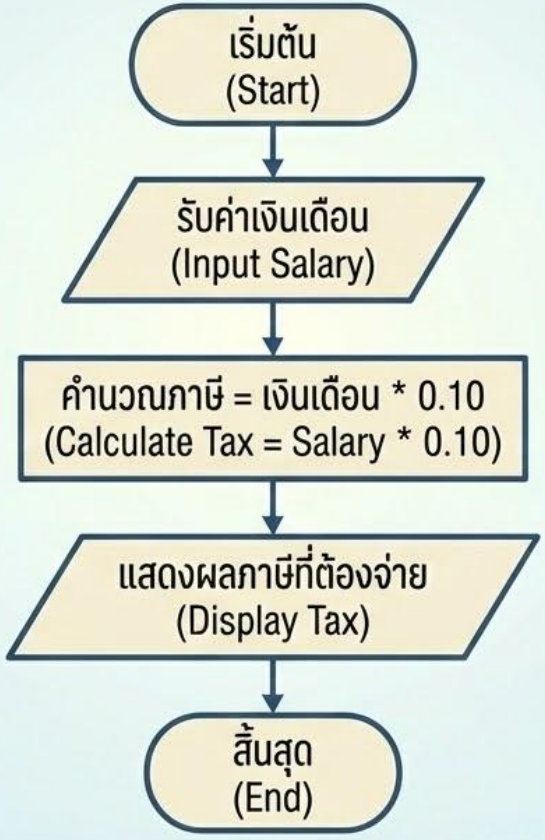
คำแนะนำ: ก่อนเขียนสูตรหารต้องมั่นใจว่าตัวหาร (ตัวส่วน) ไม่มีทางเป็น 0

ตัวอย่างการเขียนโปรแกรมคำนวณภาษีเงินได้ 10% (Tax Calculation Program Example)



คำนวณภาษี 10% จากเงินเดือนที่ได้รับ และแสดงผลลัพธ์
(Calculate 10% tax from received salary and display the result).

ส่วนที่ 1: Flowchart (แผนผังการทำงาน)



ส่วนที่ 2: ขั้นตอนที่มนุษย์เข้าใจได้ (Human-Readable Steps)

1. เริ่มต้นโปรแกรม
2. รับข้อมูลเงินเดือนจากผู้ใช้
3. คำนวณภาษี 10% โดยนำเงินเดือนมาคูณด้วย 0.10
4. แสดงผลลัพธ์ภาษีที่ต้องจ่าย
5. สิ้นสุดการทำงาน

ส่วนที่ 3: ตัวอย่าง Coding (Python Example)

```python
# โปรแกรมคำนวณภาษี 10% จากเงินเดือน
salary = float(input("กรุณากรอกเงินเดือน: ")) # รับค่าเงินเดือนและแปลงเป็นทศนิยม
tax = salary * 0.10                           # คำนวณภาษี 10%
print("ภาษีที่คุณต้องจ่ายคือ:", tax, "บาท")      # แสดงผลลัพธ์ภาษี
```