MIIA0106

Python and C Programming Language

อาจารย์ สุทิศ องอาจ

ภาคการศึกษา 2 ปีการศึกษา 2567

MIIA0106	Python and C Programming Language		SUN-เช้า-23/03/2568
A	MON/2	D604	[CPA/1,CPAI/1,EMAT/1]#ALL#
	MON/3	MII203	
	MON/3	MII202 A	
	MON/3	MII202 B	
B	SAT/2	MII207 B	[CPA+/1,EMAT+/1,EMA/2]#ALL#
	SUN/1	MII202 A	
	SUN/1	MII202 B	
EP	- /0	-	

Update V1 2024-11-16

Update V2 2024-11-20

Update V3 2024-11-30

3. คาบที่ 4: ทบทวน และ ทดสอบย่อย (6 ชม.)

1.1. ตัวแปร ชนิดข้อมูล และค่าคงที่ใน python และ C++

หัวข้อ	Python	C++
การกำหนดตัวแปร	ไม่ต้องระบุชนิดข้อมูล	ต้องระบุชนิดข้อมูล
ชนิดข้อมูลพื้นฐาน	int, float, str, bool	int, float, double, char, bool
ค่าคงที่	ใช้ convention (ตัวพิมพ์ใหญ่)	ใช้ const หรือ #define

ชนิดข้อมูล	Python	C++	คำอธิบาย
int	x = 10	int x = 10;	จำนวนเต็ม
float	y = 3.14	float y = 3.14;	จำนวนทศนิยม
str / char	name = "Alice"	char name = 'A';	ข้อความใน Python / ตัวอักษรใน C++
bool	is_active = True	bool isActive = true;	ค่าจริง/เท็จ
list	numbers = [1, 2, 3]	ใช้ std::vector	รายการที่เก็บข้อมูลหลายค่า
tuple	coords = (10, 20)	ใช้ std::pair หรือ std::tuple	คล้าย list แต่เปลี่ยนค่าไม่ได้
dict	person = {"name": "A"}	ใช้ std::map หรือ std::unordered_map	เก็บข้อมูลแบบคู่ key-value
set	unique_vals = {1, 2, 3}	std::set	เก็บค่าที่ไม่ซ้ำกัน
double	-	double z = 3.14159;	จำนวนทศนิยมความละเอียดสูง (เฉพาะ C++)
string	-	std::string name = "A";	ข้อความใน C++ (ใช้ #include <string>)

ใน Python และ C++ มีความแตกต่างในเรื่องของตัวแปร ชนิดข้อมูล และค่าคงที่ดังนี้:

1.1.1.Python

Python เป็นภาษาแบบ dynamic typing ดังนั้นชนิดข้อมูลไม่จำเป็นต้องระบุเมื่อกำหนดตัวแปร

ตัวแปร

ชนิดข้อมูลพื้นฐานใน Python: int (จำนวนเต็ม) float (จำนวนทศนิยม) str (ข้อความ) bool (ค่าจริง/เท็จ)

list (รายการ)

tuple (ทูเพิล)

dict (พจนานุกรม)

set (เซ็ต)

การกำหนดตัวแปร:

```
x = 10          # int
y = 3.14        # float
name = "Alice"  # str
is_active = True # bool
```

ตรวจสอบชนิดข้อมูล:

```
print(type(x))  # <class 'int'>
```

ค่าคงที่

Python ไม่มี const แบบใน C++ แต่เรามักใช้ convention โดยการตั้งชื่อตัวแปรด้วยตัวพิมพ์ใหญ่:

```
PI = 3.14159
GRAVITY = 9.8
```

1.1.2.ภาษา C++

ชนิดข้อมูลพื้นฐานใน C++:

int (จำนวนเต็ม)

float (จำนวนทศนิยม)

double (จำนวนทศนิยมความละเอียดสูง)

char (ตัวอักษร)

bool (ค่าจริง/เท็จ)

C++ เป็นภาษาแบบ static typing ซึ่งต้องระบุชนิดข้อมูลเมื่อกำหนดตัวแปร

```
int x = 10;       // จำนวนเต็ม
float y = 3.14;   // จำนวนทศนิยม
char name = 'A';  // ตัวอักษร
bool isActive = true; // ค่าจริง/เท็จ
```

ค่าคงที่

ใช้ const หรือ #define เพื่อกำหนดค่าคงที่:

```
const double PI = 3.14159;
#define GRAVITY 9.8
```

ตรวจสอบค่า (ไม่มี type() แบบ Python):

```
cout << typeid(x).name() << endl; // ต้องใช้ <typeinfo>
```

1.2. สรุปคำสั่ง Input/Output พื้นฐานใน Python และ C++

หัวข้อ	Python	C++
คำสั่ง Input	input()	cin หรือ getline()
คำสั่ง Output	print()	cout
การจัดรูปแบบ Output	f-string, .format(), %	fixed, setprecision, หรือ manipulators
ชนิดข้อมูล Input	ทุกอย่างเป็น str ต้องแปลงเอง	รองรับหลายชนิดข้อมูลในตัว

1.2.1. Python

Input

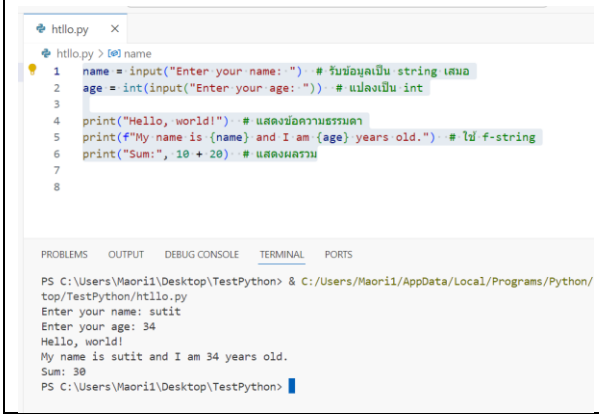
ใช้ฟังก์ชัน **input()** เพื่อรับข้อมูลจากผู้ใช้:

```
name = input("Enter your name: ")  # รับข้อมูลเป็น string เสมอ
age = int(input("Enter your age: "))  # แปลงเป็น int
```

Output

ใช้ฟังก์ชัน print() เพื่อแสดงผลลัพธ์:

```python
name = input("Enter your name: ")  # รับข้อมูลเป็น string เสมอ
age = int(input("Enter your age: "))  # แปลงเป็น int

print("Hello, world!")  # แสดงข้อความธรรมดา
print(f"My name is {name} and I am {age} years old.")  # ใช้ f-string
print("Sum:", 10 + 20)  # แสดงผลรวม
```

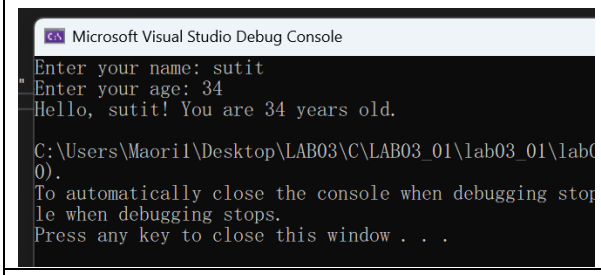


1.2.2. C++

Inputใช้คำสั่ง cin สำหรับรับข้อมูล: และ Output ใช้คำสั่ง cout เพื่อแสดงผลลัพธ์:

```cpp
#include <iostream> // สำหรับ cout และ cin
#include <string>   // สำหรับ string และ getline
using namespace std;

int main() {
    // ตัวแปรสำหรับเก็บข้อมูล
```

```
    int age;
    string name;

    // Output: การแสดงผลด้วย cout
    cout << "Enter your name: ";

    // Input: การรับข้อมูลด้วย cin
    getline(cin, name); // รับข้อมูลเป็น string ทั้งบรรทัด

    cout << "Enter your age: ";
    cin >> age; // รับข้อมูลตัวเลข

    cin.ignore(); // ล้าง newline ที่เหลือจาก cin
    // แสดงผลข้อมูลที่รับมา
    cout << "Hello, " << name << "! You are " << age << " years old." << endl;


    return 0; // โปรแกรมสิ้นสุดการทำงาน
}
```



1.3. สรุปตัวดำเนินการทางคณิตศาสตร์และตรรกะใน Python และ C++

ตัวดำเนินการ (Operators) เป็นเครื่องมือที่ใช้ในการดำเนินการกับตัวแปรหรือค่าต่าง ๆ ในภาษา Python และ C++ โดยแบ่งออกเป็นกลุ่มสำคัญ ได้แก่ ตัวดำเนินการทางคณิตศาสตร์และตรรกะ

กลุ่มตัวดำเนินการ	Python	C++
คณิตศาสตร์	+, -, *, /, //, %, **	+, -, *, /, %, pow()
ตรรกะ	and, or, not	&&, `
เปรียบเทียบ	==, !=, <, >, <=, >=	==, !=, <, >, <=, >=

1.3.1. ตัวดำเนินการทางคณิตศาสตร์

ประเภท	Python	C++	คำอธิบาย
การบวก	+	+	บวกค่าของสองตัว
การลบ	-	-	ลบค่าของสองตัว
การคูณ	*	*	คูณค่าของสองตัว
การหาร	/	/	หารค่าของสองตัว และผลลัพธ์เป็นทศนิยม
การหารเอาส่วนเต็ม	//	/	หารค่าของสองตัวและคืนค่าที่ปัดเศษลง
การหารเอาเศษ mod	%	%	คืนค่าเศษจากการหาร
ยกกำลัง	**	pow(base, exp) หรือ ^ (ในบางบริบท)	ยกกำลังค่าของตัวแปร
เพิ่มค่าตัวเอง	+=	+=	เพิ่มค่าตัวแปรด้วยค่าที่กำหนด
ลดค่าตัวเอง	-=	-=	ลดค่าตัวแปรด้วยค่าที่กำหนด

```python
a = 10
b = 3

print(a + b)  # ผลลัพธ์: 13
print(a - b)  # ผลลัพธ์: 7
print(a * b)  # ผลลัพธ์: 30
print(a / b)  # ผลลัพธ์: 3.333...
print(a // b) # ผลลัพธ์: 3
print(a % b)  # ผลลัพธ์: 1
print(a ** b) # ผลลัพธ์: 1000
```

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    int a = 10, b = 3;

    cout << a + b << endl;  // ผลลัพธ์: 13
    cout << a - b << endl;  // ผลลัพธ์: 7
    cout << a * b << endl;  // ผลลัพธ์: 30
    cout << a / b << endl;  // ผลลัพธ์: 3
    cout << a % b << endl;  // ผลลัพธ์: 1
    cout << pow(a, b) << endl;  // ผลลัพธ์: 1000
    return 0;
}
```

1.3.2. ตัวดำเนินการทางตรรกะ

ประเภท	Python	C++	คำอธิบาย
AND	and	&&	คืนค่า True ถ้าทั้งสองเงื่อนไขเป็นจริง
OR	or	||	
NOT	not	!	เปลี่ยนค่าของเงื่อนไขกลับด้าน

```python
x = True
y = False

print(x and y)  # ผลลัพธ์: False
print(x or y)   # ผลลัพธ์: True
print(not x)    # ผลลัพธ์: False
```

```cpp
#include <iostream>
using namespace std;

int main() {
    bool x = true, y = false;

    cout << (x && y) << endl;  // ผลลัพธ์: 0 (False)
    cout << (x || y) << endl;  // ผลลัพธ์: 1 (True)
    cout << (!x) << endl;      // ผลลัพธ์: 0 (False)
    return 0;
}
```

1.3.3. ตัวดำเนินการเปรียบเทียบ

ประเภท	Python	C++	คำอธิบาย
เท่ากัน	==	==	ตรวจสอบว่าสองค่าเท่ากันหรือไม่
ไม่เท่ากัน	!=	!=	ตรวจสอบว่าสองค่าไม่เท่ากันหรือไม่
มากกว่า	>	>	ตรวจสอบว่าสิ่งแรกมากกว่าสิ่งที่สองหรือไม่
น้อยกว่า	<	<	ตรวจสอบว่าสิ่งแรกน้อยกว่าสิ่งที่สองหรือไม่
มากกว่าหรือเท่ากับ	>=	>=	ตรวจสอบว่าสิ่งแรกมากกว่าหรือเท่ากับสิ่งที่สอง
น้อยกว่าหรือเท่ากับ	<=	<=	ตรวจสอบว่าสิ่งแรกน้อยกว่าหรือเท่ากับสิ่งที่สอง

```python
a = 10
b = 20

print(a == b)  # ผลลัพธ์: False
print(a != b)  # ผลลัพธ์: True
print(a > b)   # ผลลัพธ์: False
print(a < b)   # ผลลัพธ์: True
print(a >= b)  # ผลลัพธ์: False
print(a <= b)  # ผลลัพธ์: True
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 10, b = 20;

    cout << (a == b) << endl;  // ผลลัพธ์: 0 (False)
    cout << (a != b) << endl;  // ผลลัพธ์: 1 (True)
    cout << (a > b) << endl;   // ผลลัพธ์: 0 (False)
    cout << (a < b) << endl;   // ผลลัพธ์: 1 (True)
    cout << (a >= b) << endl;  // ผลลัพธ์: 0 (False)
    cout << (a <= b) << endl;  // ผลลัพธ์: 1 (True)
    return 0;
}
```

1.4. สรุปคำสั่งตัดสินใจใน Python และ C++

คำสั่ง	Python	C++
if	ใช้ if, elif, else	ใช้ if, else if, else
switch-case		ใช้ switch และ case พร้อม break
break	ใช้ในลูป (หยุดการทำงานในลูป)	ใช้ในลูปและ switch-case (หยุดทันที)
continue	ใช้ในลูป (ข้ามคำสั่งที่เหลือในรอบนั้น)	ใช้ในลูป (ข้ามคำสั่งที่เหลือในรอบนั้น)

1.4.1. คำสั่ง if

if เงื่อนไข:
คำสั่งเมื่อเงื่อนไขเป็นจริง
elif เงื่อนไขอื่น:
คำสั่งเมื่อเงื่อนไขอื่นเป็นจริง
else:
คำสั่งเมื่อไม่มีเงื่อนไขใดเป็นจริง
	if (เงื่อนไข) {
	คำสั่งเมื่อเงื่อนไขเป็นจริง;
	} else if (เงื่อนไขอื่น) {
	คำสั่งเมื่อเงื่อนไขอื่นเป็นจริง;
	} else {
	คำสั่งเมื่อไม่มีเงื่อนไขใดเป็นจริง;
	}

```python
x = 10
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
else:
    print("Negative")
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int x = 10;
    if (x > 0) {
        cout << "Positive" << endl;
    }
    else if (x == 0) {
        cout << "Zero" << endl;
    }
    else {
        cout << "Negative" << endl;
    }
    return 0;
}
```

1.4.2.คำสั่ง switch-case

match ตัวแปรหรือเงื่อนไข:
case เงื่อนไข1:
คำสั่งเมื่อเงื่อนไขตรงกับกรณีที่ 1
case เงื่อนไข2:
คำสั่งเมื่อเงื่อนไขตรงกับกรณีที่ 2
case _:
คำสั่งเมื่อไม่มีกรณีใดตรง (default)
	switch (ตัวแปรหรือเงื่อนไข) {
	case ค่า1:
	คำสั่งเมื่อเป็นค่า1;
	break;
	case ค่า2:
	คำสั่งเมื่อเป็นค่า2;
	break;
	default:
	คำสั่งเมื่อไม่มีค่าใดตรง;
	}

```python
choice = 2

match choice:
    case 1:
        print("You selected 1")
    case 2:
        print("You selected 2")
    case 3:
        print("You selected 3")
    case _:
        print("Invalid selection")
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int choice = 2;

    switch (choice) {
    case 1:
        cout << "You selected 1" << endl;
        break;
    case 2:
        cout << "You selected 2" << endl;
        break;
    case 3:
        cout << "You selected 3" << endl;
        break;
    default:
        cout << "Invalid selection" << endl;
    }

    return 0;
}
```

1.5. สรุปคำสั่งวนลูปใน Python และ C++

1.5.1.สรุปโครงสร้างและการใช้งาน

โครงสร้างลูป	Python	C++
For Loop	for ตัวแปร in ช่วง:	for (การเริ่มต้น; เงื่อนไข; การเพิ่ม/ลดค่า) { ... }
While Loop	while เงื่อนไข:	while (เงื่อนไข) { ... }
Do-While Loop	ไม่มี	do { ... } while (เงื่อนไข);
Break	break	break;
Continue	continue	continue;

1.5.2.สรุปคำสั่งวนลูปใน Python และ C++

ประเภทลูป	Python Syntax	C++ Syntax
for Loop	for i in range(start, end):	for (int i = start; i < end; i++) { ... }
while Loop	while condition:	while (condition) { ... }
do-while Loop	ไม่มีใน Python	do { ... } while (condition);
break	break	break;
continue	continue	continue;

1.5.3.For Loop

ใช้เมื่อเราทราบจำนวนรอบการวนล่วงหน้า เช่น การวนซ้ำในช่วงของค่าที่กำหนด

for ตัวแปร in ช่วงของค่าหรือ iterable:
คำสั่งที่ต้องการทำซ้ำ
	for (การกำหนดค่าเริ่มต้น; เงื่อนไข; การเพิ่ม/ลดค่า) {
	คำสั่งที่ต้องการทำซ้ำ;
	}

`# วนซ้ำตั้งแต่ 1 ถึง 5`
`for i in range(1, 6):`
`  print(f"รอบที่ {i}")`
	`#include <iostream>`
	`using namespace std;`
	
	`int main() {`
	`  for (int i = 1; i <= 5; i++) {`
	`  cout << "รอบที่ " << i << endl;`
	`  }`
	`  return 0;`
	`}`

1.5.4.While Loop

ใช้เมื่อยังไม่ทราบจำนวนรอบที่แน่นอน แต่ทราบเงื่อนไขการหยุดลูป

while เงื่อนไข:
คำสั่งที่ต้องการทำซ้ำ
	while (เงื่อนไข) {
	คำสั่งที่ต้องการทำซ้ำ;
	}

`# วนลูปจนกว่าจะถึง 5`
`count = 1`
`while count <= 5:`
`  print(f"รอบที่ {count}")`
`  count += 1`
	`#include <iostream>`
	`using namespace std;`
	
	`int main() {`
	`  int count = 1;`
	`  while (count <= 5) {`
	`  cout << "รอบที่ " << count << endl;`
	`  count++;`
	`  }`
	`  return 0;`
	`}`

1.5.5.Do-While Loop (เฉพาะ C++)

ใช้เมื่อเราต้องการให้โค้ดในลูปทำงานอย่างน้อย 1 ครั้งก่อนตรวจสอบเงื่อนไข

```
do {
    คำสั่งที่ต้องการทำซ้ำ;
} while (เงื่อนไข);
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int count = 1;
    do {
        cout << "รอบที่ " << count << endl;
        count++;
    } while (count <= 5);
    return 0;
}
```

1.6. คำสั่งพิเศษ: Break และ Continue

ใช้ในทุกลูปเพื่อควบคุมการทำงาน

break: หยุดลูปทันที

continue: ข้ามคำสั่งที่เหลือในรอบนั้น และเริ่มรอบใหม่

```python
# Break
for i in range(1, 6):
    if i == 3:
        break
    print(i)

# Continue
for i in range(1, 6):
    if i == 3:
        continue
    print(i)
```

```cpp
#include <iostream>
using namespace std;

int main() {
    for (int i = 1; i <= 5; i++) {
        if (i == 3) break; // หยุดลูป
        cout << i << endl;
    }

    for (int i = 1; i <= 5; i++) {
        if (i == 3) continue; // ข้ามรอบนี้
        cout << i << endl;
    }

    return 0;
}
```

1.7. ทดสอบย่อยครั้งที่ 1 (5 คะแนน)

ข้อสอบจะคัดมาจาก การทดลอง