บทที่ 2 ตัวแปร นิพจน์ คำสั่งรับข้อมูล และแสดงผลข้อมูล

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

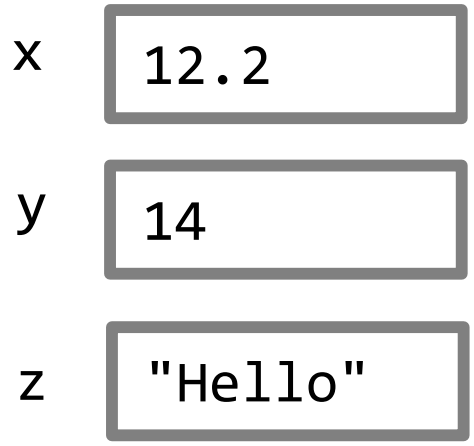
วัตถุประสงค์

- สามารถสร้างตัวแปรและใช้งานตัวแปรได้อย่างถูกต้อง
- สามารถแสดงข้อความออกทางจอภาพและรับข้อมูลจากคีย์บอร์ดได้
- สามารถใช้ Operator ทางคณิตศาสตร์ได้
- เขียนโปรแกรมประมวลผลตัวอักษรได้
- เปลี่ยนข้อความเป็นจำนวนเต็มหรือทศนิยมได้

2. ตัวแปร (Variables)

- ตัวแปรเป็นชื่อสำหรับเรียกหน่วยความจำ
- ตัวแปรมีหน้าที่ 2 แบบ
  - เก็บข้อมูล (store data)
  - นำไปใช้งาน (retrieve data)

```
x = 12.2
y = 14
z = "Hello"
```



- การสร้างตัวแปรทำได้โดยการใช้เครื่องหมายกำหนดค่า (=) Assignment operator
- ตัวแปรมีชนิดของข้อมูล (data type: string, int, float)

2.1 ตัวแปร - การตั้งชื่อ

- ชื่อตัวแปรจะต้องไม่เป็นคำสงวน (reserved word)
- อักขระตัวแรกต้องเป็นอักษรภาษาอังกฤษ หรือ underscore
- อักขระตัวถัดไปสามารถมีตัวเลขได้
- ไม่สามารถมีช่องว่างภายในชื่อได้
- ตัวอักษรตัวใหญ่ และตัวเล็ก เป็นคนละชื่อกัน (case sensitive)
- ควรตั้งชื่อตัวแปรให้มีความหมาย
- ตัวแปรปรกติ ควรใช้อักษรตัวเล็ก

2.1.1 คำสงวน (Reserved word) ในภาษาไพธอน

False	class	return	is	finally
None	if	for	lambda	continue
True	def	from	while	nonlocal
and	del	global	not	with
as	elif	try	or	yield
assert	else	import	pass	break
except	in	raise		

Hint : ในโปรแกรม VScode Reserved Word จะเป็น สีม่วง

2.1.2 ตัวอย่างการตั้งชื่อและกำหนดค่าตัวแปร

ตั้งชื่อตัวแปรถูกต้อง

```
_money = 99.25
num = 191
my_name = "John"
name2 = 'Luke'
area = 4*5
income = 25_000_67
addr = 0x64ab_41CE
```

ตั้งชื่อตัวแปรผิด

```
$money = 14.125
lambda = 2.7
my-name= "Elizabeth"
First name= "John"
percent% = 60
V1.5 = 23
```

แตกต่างกัน: myname MyName myName mynamE

2.1.3 ตัวอย่างการตั้งชื่อไม่เหมะสม


```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
437.5
```

```
width = 35.0
height = 12.50
area = width * height
print("area =",area)
```

```
area = 437.5
```

2.2 ตัวแปรกับชนิดของข้อมูล (Variable and Data type)

Data type

- `string` เก็บข้อมูลข้อความ
- `int` เก็บข้อมูลจำนวนเต็ม
- `float` เก็บข้อมูลทศนิยม
- `bool` เก็บข้อมูล True หรือ False

2.3.1 ตัวแปรชนิดข้อความ


```
str1 = "Hello1 "
str2 = 'Hello2'
str3 = "I'm a programmer."
str4 = "Hello"
```

- คร่อมด้วยเครื่องหมาย single quote(') หรือ double quote(")
- ไม่ใช่เครื่องหมายแบบเอียง (‘’) หรือ (“ ”)
- ถ้าคร่อมด้วย single quote สามารถใส่ double quote ภายในได้
- ถ้าคร่อมด้วย double quote สามารถใส่ single quote ภายในได้

2.3.2 ตัวแปรชนิดข้อความ การตรวจสอบชนิดข้อมูล


```
str1 = "Hello1"
str2 = 'Hello2'
str3 = "Hello"
str4 = "I'm a programmer."
print("str1 =", str1, " type =>", type(str1))
print("str2 =", str2, " type =>", type(str2))
print(str3, str4)
```

```
str1 = Hello1  type => <class 'str'>
str2 = Hello2  type => <class 'str'>
Hello I'm a programmer.
```

- type เป็นฟังก์ชั่นสำหรับตรวจสอบชนิดข้อมูล (Data Type)

2.3.3 ตัวแปรชนิดข้อความ การแสดงผล เอฟสตริง


```
str1,str2 = "Hello",'Linda'
print(f"line1 --{str1} {str2}--")
print(f"line2 --{str1+str2}--")
print(f"line3 = --{str1:10s}--")
print(f"line4 = --{str1:^10s}--")
print(f"line5 = --{str1:<10s}--")
print(f"line6 = --{str1:>10s}--")
```

```
line1 --Hello Linda--
line2 --HelloLinda--
line3 = --Hello     --
line4 = --  Hello   --
line5 = --Hello     --
line6 = --     Hello--
```

2.4.1 จำนวนเต็ม การกำหนดค่า


```
x = 123
print(x, x*1000, x*5000)
print(f"x = {x*5000:,}")
print(f"x = --{x:5d}--")
print(f"x = --{x:<5d}--")
print(f"x = --{x:>5d}--")
print(f"x = --{x:^5d}--")
```

```
123 123000 615000
615,000
x = --  123--
x = --123  --
x = --  123--
x = -- 123 --
```

2.4.2 จำนวนเต็มเลขฐานสิบหก กำหนดค่าและแสดงผล


```
x = 0x7f
print("x =",x)
print(f"x = {x*10000:,}")
print(f"x = --{x:5x}--")
print(f"x = --{x:<5x}--")
print(f"x = --{x:>5x}--")
print(f"x = --{x:^5x}--")
```

```
x = 127
x = 1,270,000
x = --   7f--
x = --7f   --
x = --   7f--
x = -- 7f  --
```

2.4.3 จำนวนเต็มฐานสิบหก แสดงผลเป็นตัวพิมพ์ใหญ่


```
x = 0x10cb
print("x =",x)
print(f"x = {x*10000:,}")
print(f"x = --{x:8X}--")
print(f"x = --{x:<8X}--")
print(f"x = --{x:>8X}--")
print(f"x = --{x:^8X}--")
```

```
x = 4299
x = 42,990,000
x = --    10CB--
x = --10CB    --
x = --    10CB--
x = --  10CB  --
```

2.4.4 จำนวนเต็ม แสดงผลเป็นเลขฐานสอง


```
x = 127
print("x =",x)
print(f"x = {x*10000:,}")
print(f"x = --{x:12b}--")
print(f"x = --{x:<12b}--")
print(f"x = --{x:>12b}--")
print(f"x = --{x:^12b}--")
```

```
x = 127
x = 1,270,000
x = --     1111111--
x = --1111111     --
x = --     1111111--
x = --  1111111   --
```

2.4.5 จำนวนเต็ม การกำหนดค่าให้อ่านง่าย


```
x = 100000000000    # difficult to read
print(f'x = {x:,d}')
y = 100_000_000_000     # read easier
print(f'y = {y:,d}')
```

```
x = 100,000,000,000
y = 100,000,000,000
```

2.4.6 จำนวนเต็ม การแสดงผลแบบมีศูนย์นำหน้า


```
num = 30
print(f"num = >>>{num:08d}<<<")
print(f"num = >>>{num:08x}<<<")
print(f"num = >>>{num:08X}<<<")
print(f"num = >>>{num:08b}<<<")
```

```
num = >00000030<<<
num = >0000001e<<<
num = >0000001E<<<
num = >00011110<<<
```

2.5.1 จำนวนทศนิยม (float)


```
x = 12.25
print(x, type(x))
print(f"1. x = --{x:0.2f}--")
print(f"2. x = --{x:0.8f}--")
print(f"3. x = --{x:10.3f}--")
print(f"4. x = --{x:10.0f}--")
print(f"5. x = --{x:^10.0f}--")
```

```
12.25 <class 'float'>
1. x = --12.25--
2. x = --12.25000000--
3. x = --    12.250--
4. x = --        12--
5. x = --    12    --
```

2.5.2 จำนวนทศนิยม (float)


```
x = 1024.25
print(x, type(x))
print(f"line1 {x:,}" )
print(f"line2 {x:f}" )
print(f"line3 {x:,f}" )
print(f"line4 {x:0.1f}" )
print(f"line5 {x:0.2f}" )
print(f"line6 {x:0.3f}" )
```

```
1024.25 <class 'float'>
line1 1,024.25
line2 1024.250000
line3 1,024.250000
line4 1024.2
line5 1024.25
line6 1024.250
```

2.5.3 จำนวนทศนิยม (float) - การจัดรูปแบบ


```
x = 12.125_125
print(x, type(x))
print(f"1. x = --{x:0.20f}--")
print(f"2. x = --{x:0.8f}--")
print(f"3. x = --{x:<10.3f}--")
print(f"4. x = --{x:>10.3f}--")
print(f"5. x = --{x:^10.3f}--")
```

```
12.125125 <class 'float'>
1. x = --12.12512500000000059686--
2. x = --12.12512500--
3. x = --12.125     --
4. x = --     12.125--
5. x = --  12.125   --
```

2.5.4 จำนวนทศนิยม (float)


```
x = 1.25e3
print("x =", f"{x:,}", type(x))
print(f"line1 {x:,}" )
print(f"line2 {x:f}" )
print(f"line3 {x:,f}" )
print(f"line4 {x:0.1f}" )
print(f"line5 {x:0.2f}" )
print(f"line6 {x:0.3f}" )
```

```
x = 1,250.0 <class 'float’>
line1 1,250.0
line2 1250.000000
line3 1,250.000000
line4 1250.0
line5 1250.00
line6 1250.000
```

2.5.5 แสดงผลจำนวนทศนิยม ระบุตำแหน่ง


```
x = 1.25e3
print("x =", f"{x:,}", type(x))
print(f"line1 {x:,}" )
print(f"line2 {x:f}" )
print(f"line3 {x:,f}" )
print(f"line4 {x:0.1f}" )
print(f"line5 {x:0.2f}" )
print(f"line6 {x:0.3f}" )
```

```
x = 1,250.0 <class 'float’>
line1 1,250.0
line2 1250.000000
line3 1,250.000000
line4 1250.0
line5 1250.00
line6 1250.000
```

2.6 คำสั่ง print


```
print(arg1, arg2, arg3, . . . )
```

- print คือฟังก์ชั่น หรือเรียกว่า คำสั่ง
- การเรียกใช้งานฟังก์ชั่น จะมีวงเล็บต่อท้ายเสมอ
- สิ่งที่อยู่วงเล็บเรียกว่า อาร์กิวเม้นต์ (argument)
- อาร์กิวเม้นต์แต่ละตัว จะคั่นด้วยเครื่องหมายคอมม่า
- เมื่อทำงาน จะเว้นช่องว่าง(space) 1 ช่องของแต่ละอาร์กิวเม้นต์
- เมื่อทำงานเสร็จ จะมีการขึ้นบรรทัดใหม่ให้อัตโนมัติ

2.6.1 คำสั่ง print


```
print(arg1, arg2, . . . sep=" ", end="\n");
```

- อาร์กิวเม้นต์ sep คือการแสดงผลระหว่าง arg1,arg2,…
- อาร์กิวเม้นต์ตัวสุดท้าย end="" คือตัวอักขระว่างเปล่า
- ถ้าเราไม่ใส่ จะหมายถึง end="\n" คือจะมีการขึ้นบรรทัดใหม่
- เราสามารถใส่เป็นอย่างอื่นได้ เช่น end="$\n" ในกรณีนี้จะมีการแสดงผล $ ก่อนขึ้นบรรทัดใหม่

2.7.1 เครื่องหมายคำนวณทางคณิตศาสตร์

เครื่องหมาย	การทำงาน	ตัวอย่าง	ผลลัพธ์
+	บวก	`print(5+3)`	`8`
-	ลบ	`print(5-3)`	`2`
*	คูณ	`print(5*3)`	`15`
/	หาร (classic)	`print(35/7)` `print(5/3)`	`5.0` `1.6666666666666667`

%	เศษการหาร
	(mod)
		`print(35%7)`
		`print(39%7)`
		`print(35.0%7)`
			`0`
			`4`
			`0.0`

//	หาร (floor)	`print(35//9)` `print(35.5//9)`	`3` `3.0`
**	ยกกำลัง	`print(2**10)` `print(4**0.5)`	`1024` `2.0`

2.7.2 เครื่องหมายแบบลดรูป

เครื่องหมาย	ตัวอย่างการใช้งาน	ตัวอย่างรูปแบบเต็ม
`+=`	`y += x`	`y = y + x`
`-=`	`y -= x`	`y = y - x`
`*=`	`y *= x`	`y = y * x`
`/=`	`y /= x`	`y = y / x`
`%=`	`y %= x`	`y = y % x`
`//=`	`y //= x`	`y = y // x`
`**=`	`y **= x`	`y = y ** x`

2.7.3 ลำดับการดำเนินการ (operator precedence)

ลำดับความสำคัญ	เครื่องหมาย
1	`()`
2	`**`
3	`*, /, //, %`
4	`+, -`
5	`<, <=, >, >=, ==, !=` `is, is not, in, not in`
6	`not`
7	`and`
8	`or`
9	`*=, /=, %=, +=, -=`

2.8 การรับข้อมูล (function input)


```
x = input("Enter something : ")
print(x,type(x))
```

```
Enter something : 706
706 <class 'str'>
```

```
Enter something : KMITL
KMITL <class 'str'>
```

```
Enter something : 3.14159265358979323846264338327795
3.14159265358979323846264338327795 <class 'str'>
```

- ข้อมูลที่รับเข้ามาจะเป็นข้อความ (string) เสมอ

2.9 การรับอินพุท 2 จำนวน


```
message = input("Enter x y : ")
print(message,type(message))
```

```
Enter x y : 5 7
5 7 <class 'str'>
```

- ในการรับข้อมูล บรรทัดแรก ด้วยคำสั่ง input
- เมื่อผู้ใช้ป้อนค่า `5 7` แล้วกด Enter
- ข้อมูลจะถูกนำไปเก็บที่ตัวแปร **`message`** เป็นข้อมูลชนิดข้อความ เราต้องการแยกเป็น 2 ข้อความก่อน

2.9 การรับอินพุท 2 จำนวน


```
m = input("Enter x y : ")
print("m => ",m,type(m))
x, y = m.split()
print("x => ",x, type(x))
print("y => ",y, type(y))
```

```
Enter x y : 5 7
m =>  5 7 <class 'str'>
x =>  5 <class 'str'>
y =>  7 <class 'str'>
```

2.9 การรับอินพุท 2 จำนวน


```
m = input("Enter x y : ")
print("m => ",m,type(m))
x, y = m.split()
print("x => ",x, type(x))
print("y => ",y, type(y))
```

```
Enter x y : 5 7
m =>  5 7 <class 'str'>
x =>  5 <class 'str'>
y =>  7 <class 'str'>
```

2.10 การแปลงข้อความเป็นจำนวนเต็ม/จำนวนทศนิยม


```
m = input("Enter age height : ")
print("m => ",m,type(m))
age, height = m.split()
print("age => ",age, type(age))
print("height => ",height, type(height))
age = int(age)
height = float(height)
print("age => ",age, type(age))
```

2.10 การแปลงข้อความเป็นจำนวนเต็ม/จำนวนทศนิยม


```
Enter age height : 18 180.5
m =>  18 180.5 <class 'str'>
age =>  18 <class 'str'>
height =>  180.5 <class 'str'>
age =>  18 <class 'int'>
height =>  180.5 <class 'float'>
print("height => ",height, type(height))
```

2.11 Casting (type conversion)


```
x = int(1)
y = int(2.8)
z = int("3")


x = float(1)
y = float(2.8)
z = float("3")
w = float("4.2")


x = str("s1")
y = str(2)
z = str(3.0)
```

2.12 คำอธิบายโปรแกรม (comment)

- คือส่วนของโปรแกรมที่ไม่มีผลต่อการทำงาน
- เมื่อโปรแกรมทำงานจะข้ามส่วนนี้ไป
- ใช้เครื่องหมาย #
- ตั้งแต่เครื่องหมาย # ไปจนจบบรรทัด จะไม่นำไปทำงาน
- จะต้องไม่เป็นส่วนหนึ่งของข้อความ
- ใช้เพื่อการเพิ่มคำอธิบายเพื่อให้อ่าน source code ได้ง่ายขึ้น
- ใช้เพื่อตรวจสอบการทำงานของโปรแกรม