เอกสารประกอบการสอนรายวิชา
477-201 การเขียนโปรแกรมคอมพิวเตอร์
การเขียนโปรแกรมภาษา Python เบื้องต้น
(Basic Python Programming)

ดร. จันทวรรณ ปิยะวัฒน์

สาขาวิชาระบบสารสนเทศทางธุรกิจ

ภาควิชาบริหารธุรกิจ คณะวิทยาการจัดการ

มหาวิทยาลัยสงขลานครินทร์

ภาคการศึกษาที่ 1/2562

คำนำ

เอกสารประกอบการสอนเล่มนี้จัดทำขึ้นสำหรับการสอนรายวิชา 477-201 การเขียนโปรแกรมคอมพิวเตอร์ (Computer Programming) ในภาคการศึกษาที่ 1 ปีการศึกษา 2562 ซึ่งเป็นรายวิชาบังคับของนักศึกษาหลักสูตรระบบสารสนเทศทางธุรกิจ ชั้นปีที่ 2 ภาควิชาบริหารธุรกิจ คณะวิทยาการจัดการ มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่ จำนวน 3 หน่วยกิต 3(2-2-5) เป็นการสอนทฤษฎี 2 ชั่วโมงต่อสัปดาห์ ปฏิบัติ 2 ชั่วโมงต่อสัปดาห์ และนักศึกษาควรศึกษาค้นคว้าด้วยตัวเอง 5 ชั่วโมงต่อสัปดาห์

วิชา 477-201 การเขียนโปรแกรมคอมพิวเตอร์ มีจุดมุ่งหมายให้นักศึกษาได้มีพื้นฐานความรู้ความเข้าใจในหลักการเขียนโปรแกรมคอมพิวเตอร์เบื้องต้นด้วยภาษา Python ส่วนประกอบต่างๆของโปรแกรมคอมพิวเตอร์และภาษา Python สามารถเขียนโปรแกรมคอมพิวเตอร์พื้นฐานด้วยภาษา Python ได้ตามการวิเคราะห์และออกแบบขั้นตอนการทำงานของโปรแกรมอย่างมีระบบ สามารถเขียนโปรแกรมแบบมีเงื่อนไขเพื่อการตัดสินใจ เขียนคำสั่งเพื่อให้โปรแกรมทำงานวนซ้ำได้ และเข้าใจการใช้งานโมดูลส่วนเสริมต่างๆ ของโปรแกรมภาษา Python เพื่อนำความรู้เหล่านี้ไปใช้ในการเขียนโปรแกรมระดับในระดับที่ยากขึ้นซึ่งได้แก่ การเขียนโปรแกรมแบบฟังก์ชันและการเขียนโปรแกรมเชิงวัตถุได้

หนังสือเล่มนี้ได้จัดแบ่งเนื้อหาออกเป็น 11 บท ในแต่ละบทจะมีแบบฝึกหัดท้ายบทเพื่อให้ผู้เรียนได้ลองวิเคราะห์และออกแบบแนวทางแก้ไขปัญหาและพัฒนาออกมาเป็นโปรแกรมด้วยภาษา Python ที่ได้เรียนรู้ไปแล้วได้ ทั้งนี้ผู้จัดทำหวังเป็นอย่างยิ่งว่าเอกสารประกอบการสอนฉบับนี้จะให้ความรู้และเป็นประโยชน์แก่ผู้เรียนและผู้อ่านทุกๆ ท่าน เพื่อสร้างความรู้ความเข้าใจในการฝึกเขียนโปรแกรมคอมพิวเตอร์เบื้องต้นให้ดียิ่งขึ้น หากมีข้อเสนอแนะประการใด ผู้จัดทำขอรับไว้ด้วยความขอบพระคุณยิ่ง

ดร.จันทวรรณ ปิยะวัฒน์

สาขาวิชาระบบสารสนเทศทางธุรกิจ

ภาควิชาบริหารธุรกิจ คณะวิทยาการจัดการ

มหาวิทยาลัยสงขลานครินทร์

สารบัญ

คำนำ iii

1 ส่วนประกอบต่างๆ ของภาษา Python 1

1.1 ตัวแปร (Variables) . . . 1

1.2 การตั้งชื่อตัวแปร . . . 1

1.3 การตั้งชื่อตัวแปรพร้อมกันหลายตัวแปร . . . 2

1.4 คำสงวน (Keywords) . . . 2

1.5 เลขประจำตัวตำแหน่งของตัวแปร . . . 3

1.6 ชนิดของข้อมูล (Types) . . . 3

1.7 เครื่องหมายสำหรับการคำนวณทางคณิตศาสตร์ . . . 5

1.7.1 การคำนวณทางคณิตศาสตร์ (Arithmetic Operators) . . . 5

1.7.2 รูปแบบการเขียนการคำนวณทางคณิตศาสตร์แบบย่อ . . . 6

1.7.3 การจัดการข้อความด้วยเครื่องหมายทางคณิตศาสตร์ . . . 7

1.8 Expressions และ Statements . . . 7

1.9 การเขียนข้อความอธิบายโปรแกรมโดยการใช้ Comment . . . 8

1.10 Source Code . . . 8

1.11 คำสั่ง `print` (ตัวแปรหรือข้อมูล) . . . 9

1.12 การใช้คำสั่ง `input()` รับค่าจากแป้นพิมพ์ . . . 9

1.13 แบบฝึกหัด . . . 10

2 ประโยคเงื่อนไขในภาษา Python (Conditional Statements) 13

2.1 การเปรียบเทียบค่า (Boolean Expressions) . . . 13

2.2 ตัวดำเนินการทางตรรกศาสตร์ (Logical Operators) . . . 14

2.3 การใช้คำสั่ง `if` เพื่อเลือกเงื่อนไข . . . 16

2.4 การใช้ `if` กับ `else` . . . 16
2.5 Chained Expressions . . . 17
2.6 Nested Expressions . . . 17
2.7 แบบฝึกหัด . . . 18

3 การเขียนและใช้งานฟังก์ชัน (Functions) 19
3.1 การเรียกใช้ฟังก์ชัน . . . 19
3.2 การเรียกใช้โมดูล (Modules) . . . 19
3.3 การสร้างฟังก์ชันในภาษา Python . . . 20
3.4 ขอบเขตของตัวแปร . . . 21
3.5 ฟังก์ชัน `return` . . . 22
3.6 การคืนค่าจากฟังก์ชัน . . . 22
3.7 การเขียนโปรแกรมเชิงฟังก์ชัน (Functional Programming) . . . 23
3.8 แบบฝึกหัด . . . 24
3.9 ฟังก์ชันที่เรียกตัวเอง (Recursion) . . . 24

4 การใช้ประโยคสั่งทำงานวนซ้ำ 27
4.1 ฟังก์ชัน `range()` . . . 27
4.2 คำสั่ง `for` . . . 27
4.3 คำสั่ง `while` . . . 28
4.4 แบบฝึกหัด . . . 31
บรรณานุกรม . . . 32

สารบัญรูป

1.1 ลำดับในการคำนวณ 5

สารบัญตาราง

1.1 คำสงวนในภาษา Python 3
1.2 สัญลักษณ์การคำนวณทางคณิตศาสตร์ 6
1.3 สัญลักษณ์การคำนวณทางคณิตศาสตร์แบบย่อ 6

2.1 ตัวดำเนินการเปรียบเทียบในภาษา Python 13
2.2 ตารางผลการใช้ and 15
2.3 ตารางผลการใช้ or 15
2.4 ตารางผลการใช้ not 15

บทที่ 1

ส่วนประกอบต่างๆ ของภาษา Python

1.1 ตัวแปร (Variables)

ตัวแปร (Variables) คือชื่อที่ผู้เขียนโปรแกรมกำหนดขึ้นมาเอง เพื่อใช้สำหรับการเก็บค่าข้อมูลในการเขียนโปรแกรมไว้ในหน่วยความจำของเครื่องคอมพิวเตอร์ โดยในภาษา Python ไม่ต้องระบุประเภทของตัวแปรไว้ในตอนที่ประกาศการตั้งชื่อตัวแปร ดังตัวอย่างต่อไปนี้

```
>>> a = 1
>>> a
1
>>> b = 2
>>> b
2
>>> a + b
3
>>> vat = 7
>>> vat
7
```

1.2 การตั้งชื่อตัวแปร

การตั้งชื่อตัวแปรสำหรับภาษา Python มีเงื่อนไขดังนี้

1. ให้ขึ้นต้นด้วยอักษรตัวภาษาอังกฤษตัวใหญ่หรือตัวเล็กตั้งแต่ Aa ถึง Zz เท่านั้น
2. ประกอบด้วยตัวอักษรหรือตัวเลข 0 ถึงเลข 9 หรือตัวขีดล่าง Underscore (_) แต่ห้ามมีช่องว่าง
3. ตัวเลข 0-9 จะนำหน้าชื่อตัวแปรไม่ได้
4. ตัวพิมพ์เล็กและตัวพิมพ์ใหญ่เป็นตัวแปรคนละตัวกัน (Case-Sensitive) เช่น Name ไม่ใช่ตัวแปรเดียวกันกับ name

5. ใช้ใส่เครื่องหมาย = ในการตั้งตัวแปรหรือให้ค่าแก่ตัวแปร
6. การตั้งชื่อตัวแปรควรตั้งอย่างสมเหตุสมผล
7. ภาษา Python จะมีคำที่ถูกสงวนไว้ในการเขียนโปรแกรม หรือ Keywords ซึ่งห้ามนำมาใช้ในการตั้งชื่อตัวแปร ชื่อฟังก์ชัน หรือ ชื่อคลาส

1.3 การตั้งชื่อตัวแปรพร้อมกันหลายตัวแปร

ในการเขียนโปรแกรมภาษา Python ผู้เขียนโปรแกรมสามารถตั้งชื่อตัวแปรพร้อมกันได้หลายตัวแปร โดยพิมพ์ตัวแปรแต่ละตัวในบรรทัดเดียวกันและคั่นแต่ละตัวแปรด้วยเครื่องหมายคอมม่า (,) ตามด้วยเครื่องหมาย (=) และกำหนดค่าตามลงไปตามลำดับการวางตัวแปร ดังตัวอย่างต่อไปนี้

```
>>> a, b, c = 1, 'Jan', 2.36
>>> a
1
>>> b
'Jan'
>>> c
2.36
>>>
```

1.4 คำสงวน (Keywords)

คำสงวน (Keywords) ในภาษา Python จะมีการสงวนคำบางคำไว้เฉพาะเพื่อใช้เป็นคำสั่งของภาษา โดยผู้เขียนโปรแกรมไม่ควรนำมาใช้ในการตั้งชื่อตัวแปร โดยคำสงวนของภาษา Python มีดังต่อไปนี้ (Lutz, 2014)

ตารางที่ 1.1: คำสงวนในภาษา Python

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

1.5 เลขประจำตัวตำแหน่งของตัวแปร

ตัวแปรจะชี้ไปที่หน่วยความจำในเครื่องคอมพิวเตอร์ซึ่งเก็บค่าของตัวแปรหรือ Value นั้นๆ อยู่ ฉะนั้นเมื่อเราพิมพ์ a ดังในตัวอย่าง คอมพิวเตอร์จึงแสดงเลข 1 ออกมา นอกจากนี้พื้นที่ที่เก็บค่านั้นนั้นจะมีที่อยู่อยู่บนหน่วยความจำมีหมายเลขประจำตำแหน่งอีกด้วย โดยใช้คำสั่ง `id()` เพื่อแสดงเลขประจำตำแหน่ง

```
>>> a
1
>>> id(a)
1538021648
```

1.6 ชนิดของข้อมูล (Types)

สิ่งที่อยู่ในหน่วยความจำมีชนิดของข้อมูลหรือ Type อยู่ด้วย โดยใช้คำสั่ง `{type()}`เพื่อดูประเภทของข้อมูล ในภาษา Python มีประเภทของข้อมูลหลายๆ แบบ (Ramalho, 2015) ทั้งแบบที่เป็นตัวเลขแบบจำนวนเต็ม ตัวเลขแบบมีจุดทศนิยม ตัวเลขที่มีค่าเป็นบวกหรือลบ ตัวอักษร ข้อความ และตรรกศาสตร์

1. none คือ Nothing ไม่มีอะไร

2. int หรือ Integer คือตัวเลข เช่น 50 หรือ 630 เป็นต้น
3. bool หรือ Boolean คือค่าถูกผิด เช่น True หรือ False เป็นต้น
4. float หรือ floating Point คือจำนวนทศนิยม เช่น 5.6 หรือ 4.23 เป็นต้น
5. str หรือ String หรือข้อความ ซึ่งจะอยู่ภายใต้เครื่องหมาย ฟันหนู (“- ”) หรือ ฝนทอง (‘- ’) เช่น `"This is my dog. "` หรือ `'Jantawan'`

ตัวอย่างการแสดงประเภทของข้อมูลมีดังต่อไปนี้

```
>>> a = 1
>>> a
1
>>> type(a)
<class 'int'>
>>> firstname = 'Jantawan'
>>> firstname
'Jantawan'
>>> lastname = 'Piyawat'
>>> lastname
'Piyawat'
>>> id(firstname)
67626832
>>> type(firstname)
<class 'str'>
```

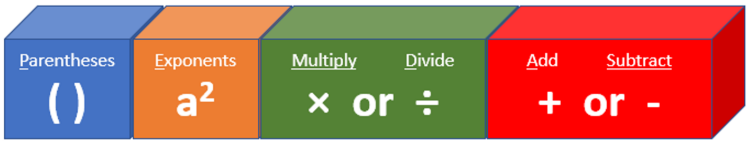
```
>>> n = None
>>> n
>>> id(n)
263420692
>>> type(n)
<class 'NoneType'>
>>> yes = True
>>> no = False
>>> type(yes)
<class 'bool'>
>>> degree = 1.1
>>> id(degree)
72213072
>>> type(degree)
<class 'float'>
```

1.7 เครื่องหมายสำหรับการคำนวณทางคณิตศาสตร์

1.7.1 การคำนวณทางคณิตศาสตร์ (Arithmetic Operators)

เครื่องหมายสำหรับการคำนวณเรียกว่า Arithmetic Operators เช่น เครื่องหมายบวก ลบ คูณ หาร การยกกำลัง การหารเอาเศษ การหารเอาจำนวนเต็ม เป็นต้น การคำนวณทางคณิตศาสตร์แบบซับซ้อนจะต้องมีลำดับในการคำนวณซึ่งเหมือนกับการคำนวณคณิตศาสตร์ทั่วไป คือ ในการแก้สมการทางคณิตศาสตร์จะต้องทำในวงเล็บก่อน ตามด้วยเลขยกกำลัง แล้วจึงตามด้วย คูณหรือหารโดยคำนวณจากซ้ายไปขวา แล้วตามด้วยบวกหรือลบโดยคำนวณจากซ้ายไปขวาเช่นกัน โดยให้จำคำว่า **PEMDAS** ซึ่งเป็นตัวอักษรภาษาอังกฤษตัวแรกของคำว่า Parentheses (วงเล็บ), Exponents (ยกกำลัง), Multiply (คูณ), Divide (หาร), Add (บวก), และ Subtract (ลบ) โดยภาษา Python ใช้สัญลักษณ์คำนวณทางคณิตศาสตร์ดังตารางต่อไปนี้ (Lubanovic, 2015)

รูปที่ 1.1: ลำดับในการคำนวณ



ตัวอย่างคำนวณทางคณิตศาสตร์ในภาษา Python มีดังต่อไปนี้

```
>>> a=1
>>> b=2
>>> a+b
3
>>> a-b
-1
>>> c = b-a
>>> c
1
```

ตารางที่ 1.2: สัญลักษณ์การคำนวณทางคณิตศาสตร์

สัญลักษณ์คำนวณ	ชื่อการคำนวณ	ตัวอย่าง
+	บวก	a + b
-	ลบ	a - b
*	คูณ	a * b
/	หาร	a / b
//	หารปัดเศษทิ้ง	a // b
%	เศษของการหาร	a % b
**	ยกกำลัง	a ** b

1.7.2 รูปแบบการเขียนการคำนวณทางคณิตศาสตร์แบบย่อ

ในภาษา Python ผู้เขียนโปรแกรมสามารถเขียนการคำนวณทางคณิตศาสตร์แบบลดรูปหรือแบบย่อได้ ดังตารางสัญลักษณ์การคำนวณทางคณิตศาสตร์แบบย่อต่อไปนี้

ตารางที่ 1.3: สัญลักษณ์การคำนวณทางคณิตศาสตร์แบบย่อ

การคำนวณ	ตัวอย่าง	เทียบเท่ากับ
+=	c += a	c = c + a
-=	c -= a	c = c - a
*=	c *= a	c = c * a
/=	c /= a	c = c / a
//=	c //= a	c = c // a
%=	c %= a	c = c % a
**=	c **= a	c = c ** a

1.7.3 การจัดการข้อความด้วยเครื่องหมายทางคณิตศาสตร์

เครื่องหมายที่เป็นการคำนวณทางคณิตศาสตร์เมื่อถูกนำมาใช้กับข้อความ (String) จะเป็นอีกความหมายหนึ่ง เช่น การใช้เครื่องหมายบวกเชื่อมต่อระหว่างสตริง 2 ตัว หรือ การใช้เครื่องหมายดอกจันเป็นการเพิ่มสตริงเดียวกันตามจำนวนครั้งของการคูณ

```
>>> firstname
'Jantawan'
>>> lastname
'Piyawat'
>>> firstname + lastname
'JantawanPiyawat'
>>> firstname + '  ' + lastname
'Jantawan Piyawat'
>>> firstname * 3
'JantawanJantawanJantawan'
```

1.8 Expressions และ Statements

Expression หมายถึงการใช้เครื่องหมายคำนวณและการใช้ตัวแปรและค่าของตัวแปรเพื่อหาผลลัพธ์ออกมา เอา Expression มาประกอบกันจะเรียกว่า Statement ดังนั้น Statement ก็คือคำสั่งเรียงต่อกันนั่นเอง เพื่อใช้ในการสั่งงานคอมพิวเตอร์ด้วยภาษาคอมพิวเตอร์

ตัวอย่าง Expression เป็นดังต่อไปนี้

```
>>> 1+2
3
```

ตัวอย่าง Statement เป็นดังต่อไปนี้

```
>>> c = a + b
>>> c
3
>>> print('hello world.')
hello world.
```

1.9 การเขียนข้อความอธิบายโปรแกรมโดยการใช้ Comment

Comment คือสิ่งที่เราเขียนใน Source Code ของโปรแกรมแต่คอมพิวเตอร์ไม่ต้องแปลผล เพื่อใช้ในการเขียนข้อความประกอบคำอธิบายในการสื่อสารระหว่างผู้เขียนโปรแกรมด้วยกัน หรือเป็นการเตือนความจำของผู้เขียนโปรแกรมเอง โดย Comment ในภาษา Python นำหน้าด้วยเครื่องหมายชาร์ป (#) แล้วหลังจากนั้นตามด้วยข้อความอะไรก็ได้ ถ้าจะเขียน Comment หลายๆ บรรทัดจะต้องใช้เครื่องหมายฟันหนู (“- ”) หรือฝนทอง (‘- ’) 3 ชุด ข้างในใส่ค่า ผลที่ได้จะมีเครื่องหมาย Backslash n (\n) หมายถึงการขึ้นบรรทัดใหม่

ตัวอย่างการใช้ Comment ในบรรทัดเดียว เป็นดังต่อไปนี้

```
>>> print('Hello world!')
Hello world!
>>> # Hello how are you doing?
```

ตัวอย่างการใช้ Comment ในหลายบรรทัด เป็นดังต่อไปนี้

```
>>> x = '''
hello
1
2
3
'''
>>> x
'\nhello\n1\n2\n3\n'
>>> print(x)
hello
1
2
3
```

1.10 Source Code

ที่ผ่านมาเป็นการเขียนโปรแกรมแบบ Interactive คือเขียนบน Python Shell แล้วโปรแกรมจะแสดงผลออกมาได้เลย ซึ่งเรียกว่าการทำงานแบบ Interpreter เป็นการใส่คำสั่งไปที่ Prompt และ Python จะแสดงผล

ของคำสั่งนั้นออกมาเลย แต่ในความเป็นจริงแล้วจะเขียนโปรแกรมหลายๆ บรรทัดแล้วสั่งโปรแกรมทำงานทีเดียวพร้อมกัน เราจะเขียนไว้ในไฟล์นั้นเรียกว่า Source Code โดยที่ Source Code ของภาษา Python นามสกุลจะเป็น `.py` เวลาใช้ที่โปรแกรม Idle ให้กดที่เมนู File เลือก New เขียน Source Code แล้วให้กด Run ถ้าหากจะกดรันโปรแกรมอีกสักครั้งให้กด F5

ตัวอย่าง Python Source Code เป็นดังต่อไปนี้

```
x = 7
y = 6
if x == y: print('x and y are equal.')
else:
    if x < y: print('x is less than y.')
    else: print('x is greater than y.')
```

ตัวอย่างผลลัพธ์ที่ได้จากการประมวลผล Source Code เป็นดังต่อไปนี้

```
x is greater than y.
```

1.11 คำสั่ง `print` (ตัวแปรหรือข้อมูล)

`print()` เป็นฟังก์ชันที่ใช้ในการแสดงผลตัวแปรหรือข้อมูลออกทางหน้าจอ ดังตัวอย่างต่อไปนี้

```
>>> print('Hello world!')
Hello world!
>>>
```

1.12 การใช้คำสั่ง `input()` รับค่าจากแป้นพิมพ์

คำสั่ง `input( Prompt)` เป็นคำสั่งสำหรับรับข้อมูลจากผู้ใช้ด้วยการพิมพ์ผ่านแป้นพิมพ์ ดังตัวอย่างต่อไปนี้

```
>>> name=input('What is your name? ')
What is your name? Jantawan
>>> print('Hello, ', name, end='.')
Hello, Jantawan.
```

1.13 แบบฝึกหัด

1. จงหาเลขประจำตำแหน่งของข้อมูลต่อไปนี้
    - love = 2
    - mom = “Jan”
    - wed = True
    - fah = 39.2
2. จงหาประเภทของข้อมูลต่อไปนี้
    - `love = 2`
    - `mom = 'Jan'`
    - `wed = True`
    - `fah = 39.2`
    - `money = '22'`
3. จงแสดงผลต่อไปนี้
    - ตั้งค่าตัวแปร dog, cat
    - แสดงข้อความ I have 3 dogs and 2 cats.
4. จงรับค่าจากผู้ใช้และแสดงผลต่อไปนี้
    - ตั้งตัวแปร name
    - รับค่าด้วยข้อความว่า กรุณาใส่ชื่อของคุณ
    - แสดงข้อความ สวัสดีค่ะคุณ
5. จงคำนวณหาค่าตัวเลขต่อไปนี้
    - หาค่าพื้นที่สี่เหลี่ยม กว้าง 5 เมตร ยาว 3 เมตร

- หาค่าพื้นที่สามเหลี่ยม สูง 5 เมตร ฐาน 3 เมตร

6. ให้ `a = 3, b = 4, c = 5` จงหาค่าต่อไปนี้

- `a == a*1`
- `a != b`
- `a > b`
- `b < c`
- `a+1 >= c`
- `c <= a+b`

บทที่ 2

ประโยคเงื่อนไขในภาษา Python (Conditional Statements)

2.1 การเปรียบเทียบค่า (Boolean Expressions)

Boolean Expressions คือ การดำเนินการเปรียบเทียบค่าเพื่อให้ได้ผลลัพธ์ออกมาเป็นถูก (**`True`**) หากเงื่อนไขเป็นจริง หรือผลลัพธ์เป็นผิด (**`False`**) หากเงื่อนไขเป็นเท็จหรือไม่เป็นจริง การเปรียบเทียบค่าใช้กับคำสั่งตรวจสอบเงื่อนไขการตัดสินใจ **`if`** และคำสั่งการทำงานวนซ้ำทั้ง **`for`** และ **`while`** เช่น ค่าของ x มากกว่าค่าของ y ใช่หรือไม่ ซึ่งผลลัพธ์จะออกมาเป็น True หรือ False โดยตัวดำเนินการเปรียบเทียบในภาษา Python ได้แสดงไว้ในตารางต่อไปนี้

ตารางที่ 2.1: ตัวดำเนินการเปรียบเทียบในภาษา Python

สัญลักษณ์	ความหมาย
x == y	x เท่ากับ y
x != y	x ไม่เท่ากับ y
x > y	x มากกว่า y
x < y	x น้อยกว่า y
x >= y	x มากกว่าหรือเท่ากับ y
x <= y	x น้อยกว่าหรือเท่ากับ y

ส่วนตัวอย่างการใช้สัญลักษณ์เปรียบเทียบค่าเป็นดังนี้

```
>>> a = 5
>>> b = 6
```

```
>>> a == b
False
>>> 5 > 6
False
>>> 5 < 6
True
>>> 5 >= 6
False
>>> print('5 == 5 :', 5 == 5)
5 == 5 : True
>>> print('12 >= 13.47 : ', 12 >= 13.47)
12 >= 13.47 : False
>>>
```

2.2 ตัวดำเนินการทางตรรกศาสตร์ (Logical Operators)

การเปรียบเทียบค่ามากกว่าหนึ่งครั้งเชื่อมต่อกันสามารถดำเนินการได้โดยใช้ตัวดำเนินการทางตรรกศาสตร์ (Logical Operators) ในภาษา Python นั้นมีตัวดำเนินการทางตรรกศาสตร์ 3 ชนิด ซึ่งได้แก่ และ (**and**) หรือ (**or**) ไม่ (**not**) เช่น `a > 2 or c > b and c >2` ตัวอย่างการใช้ตัวดำเนินการทางตรรกศาสตร์ในภาษา Python เป็นดังต่อไปนี้

```
>>> a = -4
>>> b = -2
>>> a > 2 or c > b and c> 2
False
```

ตัวดำเนินการทางตรรกศาสตร์ **and** ใช้เชื่อมสอง Expression ถ้าทั้งสอง Expression มีค่าเป็น True จะได้ผลลัพธ์เป็น **True** ส่วนตัวดำเนินการทางตรรกศาสตร์ **or** ใช้เชื่อมสอง Expression ถ้าทั้งสอง Expression มีค่าเป็น False เท่านั้น ผลลัพธ์จึงจะเป็น **False** และตัวดำเนินการทางตรรกศาสตร์ **not** จะใช้ในการกลับค่าจาก **True** เป็น **False** และจาก **False** เป็น **True** ซึ่งผลการใช้ตัวดำเนินการทางตรรกศาสตร์ทั้งสามตัว แสดงไว้ในตารางดังต่อไปนี้

ตารางที่ 2.2: ตารางผลการใช้ and

Boolean Expression 1 (BE1)	Boolean Expression 1 (BE2)	BE1 and BE2
False	False	False
False	True	False
True	False	False
True	**True**	**True**

ตารางที่ 2.3: ตารางผลการใช้ or

Boolean Expression 1 (BE1)	Boolean Expression 1 (BE2)	BE1 or BE2
False	**False**	**False**
False	True	True
True	False	True
True	True	True

ตารางที่ 2.4: ตารางผลการใช้ not

Boolean Expression	Not BE1
False	True
True	False

2.3 การใช้คำสั่ง if เพื่อเลือกเงื่อนไข

เงื่อนไขที่ใช้ในภาษา Python คือ if Statement สิ่งที่ตามหลัง if คือ Boolean Expression เรียกว่า Statement ใหญ่ และใน Statement ใหญ่ ก็มี Statement ย่อย การดูว่า Statement ย่อยอยู่ใน if Statement ใดให้ดูที่การย่อหน้าหรือ Indentation ในภาษา Python การย่อหน้าสำคัญมากจะเป็นการบอกว่าอะไรอยู่ภายในอะไร

รูปแบบของการใช้งานคำสั่ง if ในภาษา Python โดยถ้าหากเงื่อนไขเป็นจริง ตัวโปรแกรมจะประมวลผลในคำสั่ง if หลังเครื่องหมาย : ดังต่อไปนี้

```
if expresion:
    #statements
```

ตัวอย่างเช่น ให้แสดงข้อความว่า อายุต่ำกว่าเกณฑ์ ถ้าหากค่าอายุที่รับเข้ามาต่ำกว่า 18 ปี การเขียน Source Code และผลลัพธ์ของโจทย์เป็นดังนี้

```
age = int(input('Enter your age: `))
if age < 18:
   print('You are underage.')
```

```
>>>
Enter your age: 15
You are underage.
>>>
```

2.4 การใช้ if กับ else

โครงสร้างคำสั่ง if...else จะดำเนินในบล็อคคำสั่ง else ถ้าหากเงื่อนไขในคำสั่ง if นั้นเป็นเท็จ โดยมีรูปแบบการเขียนดังนี้

```
if expression:
    # statements
else:
    # statements
```

ตัวอย่างการใช้ `if...else` และผลลัพธ์เป็นดังต่อไปนี้

```
x = 15
y = 6
if x > y: print('x is greater than y.')
else: print('x is less than or equal to y.')
```

```
>>>
x is greater than y.
>>>
```

2.5 Chained Expressions

การใช้ Chained Expressions คือ การใส่ elif ไปเรื่อยๆ และเงื่อนไขสุดท้ายจะต้องใช้ else โดยไม่ต้องการระบุ Boolean Expressions ใดๆ อีกแล้วหลังจากที่ใส่ else ดังตัวอย่างและผลลัพธ์ดังต่อไปนี้

```
x = 6
y = 6
if x > y: print('x is greater than y.')
elif x < y: print('x is less than y.')
else: print('x and y are equal.')
```

```
>>>
x and y are equal.
>>>
```

2.6 Nested Expressions

if มี Statement อยู่ข้างในได้ และ else ก็มี Statement อยู่ข้างในได้เช่นกัน เรียกว่า Nested Expressions ดังตัวอย่างและผลลัพธ์ดังต่อไปนี้

```
x = 7
y = 6
if x == y: print('x and y are equal.')
else:
    if x < y: print('x is less than y.')
    else: print('x is greater than y.')
```

```
>>>
x is greater than y.
>>>
```

2.7 แบบฝึกหัด

1. จงเขียน code ต่อไปนี้
    - ให้ถามว่า Are you bored? และให้ตอบว่า y หรือ n
    - ถ้าตอบ y ให้พิมพ์ข้อความว่า Let's go outside.
2. จงเขียน code ต่อไปนี้
    - ตั้งค่าตัวแปร var รับค่าเป็นตัวเลขจำนวนเต็มจากผู้ใช้
    - ถ้า var > 100 ให้แสดงผลว่า "The value is over 100."
    - ถ้า เป็นกรณีอื่นๆ ให้แสดงผลว่า "The value is less than or equal 100."
3. จงเขียน code ต่อไปนี้
    - รับค่า a, b เป็นจำนวนเต็ม
    - แสดงผลว่า a > b หรือ a < b หรือ a = b
4. จงเขียน code ต่อไปนี้
    - รับค่า score เป็นจุดทศนิยม
    - ถ้าคะแนน 81-100 แสดงผลว่า เกรด A
    - ถ้าคะแนน 61-80 แสดงผลว่า เกรด B
    - ถ้าคะแนน 41-60 แสดงผลว่า เกรด C
    - ถ้าคะแนน 0-40 แสดงผลว่า เกรด F
    - เมื่อแสดงผลดังกล่าวแล้ว ให้แจ้งด้วยว่า "ตัดเกรดแล้ว"

บทที่ 3

การเขียนและใช้งานฟังก์ชัน (Functions)

3.1 การเรียกใช้ฟังก์ชัน

ฟังก์ชัน (Function) คือชุดคำสั่งส่วนหนึ่งของโปรแกรมที่ทำงานเพื่อวัตถุประสงค์เฉพาะอย่าง ฟังก์ชันในภาษา Python มีทั้ง Built-in ฟังก์ชันของภาษา Python เอง และฟังก์ชันที่ผู้เขียนโปรแกรมเขียนขึ้นมา การใช้ฟังก์ชันในภาษา Python จะคล้ายกับฟังก์ชันทางคณิตศาสตร์ คือ กำหนดชื่อฟังก์ชันตามด้วยสิ่งที่อยู่ในวงเล็บ ซึ่งเรียกว่า Arguments ซึ่งอาจจะมีได้มากกว่า 1 และในภาษา Python มีการกำหนดฟังก์ชันมาให้เรียกใช้ได้เลยอยู่บ้างแล้ว เช่น `type(42)` คือ การแสดงค่าประเภทของเลข 42 หรือ `id(42)` คือการแสดงตำแหน่งที่อยู่ของเลข 42 ในหน่วยความจำ ดังตัวอย่างต่อไปนี้

```
>>> type(42)
<class 'int'>
>>> a = 1
>>> id(a)
1538021648
```

3.2 การเรียกใช้โมดูล (Modules)

โมดูล (Modules) คือ ฟังก์ชันที่รวมกันไว้เป็นหมวดหมู่ และสามารถดึงมาใช้ได้ในโปรแกรมได้ด้วยการ `import` เช่น `import math` และหากเรียกใช้ฟังก์ชัน `dir(math)` จะแสดงฟังก์ชันในโมดูล math ออกมา ดังตัวอย่างต่อไปนี้

```
>>> import math
>>> type(math)
<class 'module'>
>>> id(math)
56337632
```

การใช้งานโมดูลจะมีการใช้งานแบบ Dot Notation หากเห็นการเขียนโมดูล math ในลักษณะนี้ เช่น `math.pi` ตัว `pi` เรียกว่าเป็นตัวแปรที่อยู่ในโมดูล `math` ซึ่งไม่ใช่ฟังก์ชัน แต่ถ้าเขียน `math.pow(2,2)` คือ สองยกกำลังสอง ลักษณะนี้จะเป็นการเรียกใช้ฟังก์ชันที่อยู่ในโมดูล ดังตัวอย่างต่อไปนี้

```
>>> math.pi
3.141592653589793
>>> math.pow(2,2)
4.0
```

การใช้ฟังก์ชันไม่จำเป็นต้องใช้ฟังก์ชันแบบฟังก์ชันเดียว ฟังก์ชันใช้ฟังก์ชันซ้อนกันได้ คือ การรวมฟังก์ชันหลายๆ อันซ้อนกัน เรียกต่อๆ กันไปได้ ดังตัวอย่างต่อไปนี้

```
>>> math.exp(math.log(3+2))
4.9999999999999
```

3.3 การสร้างฟังก์ชันในภาษา Python

การเขียนฟังก์ชันหรือสร้างฟังก์ชันขึ้นมาเองจะกระทำเพื่อต้องการให้โปรแกรมที่เขียนขึ้นมาทำงานเพื่อวัตถุประสงค์เฉพาะอย่าง ต้องใช้ `def` Statement ซึ่งย่อมาจาก Define เมื่อเขียนฟังก์ชันไว้ใน Source Code แล้วทำการ Run ฟังก์ชันนั้นจะถูก Define ไว้ในระบบแล้วถูกเรียกใช้ขึ้นมาได้เลย ด้วยการเรียกชื่อฟังก์ชันนั้นกี่ครั้งต่อกี่ครั้งก็ได้ ดังรูปแบบการเขียนต่อไปนี้ทั้งในแบบที่มีการคืนค่าและไม่มีการคืนค่า

```
def function_name(args...):
    # statements

def function_name(args...):
    # statements
    return value
```

ส่วนวิธีการเรียกใช้ฟังก์ชันทำได้โดย เรียกชื่อฟังก์ชันนั้นตามด้วยวงเล็บ () ซึ่งจะมี Arguments หรือไม่ก็แล้วแต่ฟังก์ชันที่กำหนดไว้ ดังตัวอย่างการเรียกใช้ฟังก์ชันที่สร้างขึ้นมาเองต่อไปนี้

```
def happy_birthday_song():
    print('Happy Birthday.')
    print('Happy Birthday to you.')
```

```
>>>  happy_birthday_song()
Happy Birthday.
Happy Birthday to you.
```

3.4 ขอบเขตของตัวแปร

ในวงเล็บ () เป็นการกำหนด Argument ของฟังก์ชัน ซึ่งจะกลายเป็นพารามิเตอร์ (Parameters) หรือตัวแปรที่ใช้ในฟังก์ชันนั้นๆ เท่านั้นและต้องประมวลผลให้เสร็จสิ้นในฟังก์ชัน หรือเรียกว่า Local Variables ตัวแปร Local นี้ จะไม่สามารถเรียกใช้ที่โปรแกรมหลักได้ และผู้เขียนโปรแกรมก็ไม่สามารถนำตัวแปรนี้ไปใช้ในฟังก์ชันอื่น ๆ ได้ ส่วนตัวแปร Global (Global Variables) จะประกาศไว้ในส่วนหลักของโปรแกรมที่เขียนขึ้น ฟังก์ชันย่อยและตัวโปรแกรมหลักสามารถเรียกใช้ตัวแปร Global ได้

```
def newage(age):
    a = 50 #local variable
    x = age + a #local variable
    return x

age = int(input('Enter your age: ')) #global variable
print('In 50 years from now, you will be ', newage(age), end='.')
```

```
Enter your age: 18
In 50 years from now, you will be  68.
>>>
```

ตัวแปรใดก็ตามที่จะใช้เป็น Local ให้ใส่คำว่า `global` ไปด้านหน้าตัวแปรที่ถูกเรียกใช้ในฟังก์ชัน อันที่จริงแล้วจะไม่เขียนคำว่า `global` ก็ได้หากชื่อตัวแปรไม่ซ้ำกันเลย

```
x = 5
def happy_birthday_song(name):
    global x
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday to ' , name)
    print(x)
happy_birthday_song('Mike')
```

```
>>>
Happy Birthday.
Happy Birthday.
Happy Birthday.
Happy Birthday to Mike
5
>>>
```

3.5 ฟังก์ชัน `return`

ฟังก์ชันทุกอันจะต้อง return คือสิ้นสุดการทำงาน แต่ได้เว้นไว้ในฐานที่เข้าใจ เมื่อโปรแกรมไล่การทำงานมาถึงจุด return โปรแกรมจะหยุดทำงานทันทีทั้งๆ ที่ยังมีคำสั่งอื่นตามมาหลังจาก return อีกก็ตาม ฟังก์ชัน `return` ซึ่งไม่ได้คืน ค่าอะไรออกมาเรียกว่า Void

```
x = 5
def happy_birthday_song(name):
    print('Happy Birthday.')
    return
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday to ' , name)
    print(x)
happy_birthday_song('Mike')
```

```
>>>
Happy Birthday.
>>>
```

3.6 การคืนค่าจากฟังก์ชัน

ฟังก์ชันสามารถ return ค่าได้ด้วย ดังเช่นตัวอย่างการหาพื้นที่สี่เหลี่ยม โดยกำหนดฟังก์ชันชื่อ `rectangle_area(width, height)` และฟังก์ชันมีพารามิเตอร์สองตัวสำหรับความกว้างและความยาวของสี่เหลี่ยม และฟังก์ชันทำการคืนค่า ผลลัพธ์ที่เป็นพื้นที่กลับไปด้วยคำสั่ง `return`

ตัวอย่างการหาพื้นที่สี่เหลี่ยม เป็นดังต่อไปนี้

```
def happy_birthday_song(name):
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday to ' , name)
    return name

def rectangle_area(width, height):
    return width * height  #

x = rectangle_area(4, 3)
print('The area of rectangle is', str(x))
```

ผลลัพธ์ของตัวอย่างการหาพื้นที่สี่เหลี่ยม เป็นดังต่อไปนี้

```
>>>
The area of rectangle is 12
>>>
```

อีกตัวอย่างของฟังก์ชันที่สร้างขึ้นมาเองโดยมีการใช้ return

```
def newage(age):
    x = age+50
    return x #

age = int(input('Enter your age: '))
print('In 50 years from now, you will be ', newage(age), end='.')
```

ผลลัพธ์จากตัวอย่างฟังก์ชันที่สร้างขึ้นมาเองโดยมีการใช้ return เป็นดังต่อไปนี้

```
>>>
Enter your age: 6
In 50 years from now, you will be  56.
>>>
```

3.7 การเขียนโปรแกรมเชิงฟังก์ชัน (Functional Programming)

การเขียนโปรแกรมเชิงฟังก์ชัน (Functional Programming) เป็นรูปแบบการเขียนโปรแกรมที่เก่าแก่ที่สุดและเป็นรูปแบบที่กลับมาได้รับความนิยมมากในปัจจุบัน คือสร้างฟังก์ชันแล้วให้ฟังก์ชันทำงานร่วมกันโดยไม่มี

การใช้ Global Variables เลย ฟังก์ชันหนึ่งทำงานส่งผลลัพธ์แก่อีกฟังก์ชันหนึ่งต่อๆ กันไปเรื่อยๆ ซึ่งภาษา Python สามารถใช้เขียนโปรแกรมแบบนี้ได้ ฟังก์ชันแล้ว return ค่าเป็นผลลัพธ์แก่อีกฟังก์ชันหนึ่งไปเรื่อยๆ

จากรูป `x=f(g(h(x)))` ทำงานเหมือนกันกับ `x = h(x)` แล้ว `x = g(x)` แล้ว `x = f(x)`

```
x = f(g(h(x)))
x = h(x)
x = g(x)
x = f(x)
```

3.8 แบบฝึกหัด

3.9 ฟังก์ชันที่เรียกตัวเอง (Recursion)

Recursion คือ การเรียกใช้ฟังก์ชันซ้อนฟังก์ชันนั้นๆ เอง หรือ ฟังก์ชันเรียกใช้ตัวมันเอง จากฟังก์ชัน `countdown()` ในตัวอย่าง เมื่อมีการเรียกฟังก์ชัน `countdown(5)` โปรแกรมจะทำงานลดหลั่นลงไปเรื่อยๆ คือตั้งแต่ 5, 4, 3, 2, 1 ตราบเท่าที่ n ยังมากกว่า 0 แต่เมื่อเงื่อนไขเป็นเท็จแล้ว โปรแกรมจะแสดงคำว่า Go! แทน ซึ่งมีลักษณะการเขียนดังนี้

```
def countdown(n):
    if x > 0:
        print(n)
        countdown(n-1)
    else:
        print('Go.')
```

ซึ่งจะได้ผลลัพธ์ดังนี้

```
>>> countdown(5)
5
4
3
2
1
Go.
```

1. ตั้งชื่อฟังก์ชัน **`hello`** เพื่อแสดงผลว่า สวัสดีคุณ
2. สร้างฟังก์ชันคำนวณพื้นที่ รับค่า ความกว้าง และ ความยาว
3. สร้างฟังก์ชันชื่อ **`maximal_2`** รับ arguments 2 ค่า และ return ค่าที่มากที่สุดออกมา
4. สร้างฟังก์ชันชื่อ **`maximal_3`** รับ arguments 3 ค่า และ return ค่าที่มากที่สุดออกมา
5. สร้างฟังก์ชันรับ arguments 3 ค่า และ return ผลคูณออกมา
6. สร้างฟังก์ชันรับค่าตัวเลขในหน่วยเมตรต่อวินาที แล้ว return ผลในหน่วยกิโลเมตรต่อชั่วโมง
7. สร้างฟังก์ชันหาผลต่างของรายรับกับรายจ่าย และส่งผลกลับมา

บทที่ 4

การใช้ประโยคสั่งทำงานวนซ้ำ

4.1 ฟังก์ชัน range()

ฟังก์ชัน `range()` คือ ระยะตั้งแต่เริ่มต้นถึงก่อนระยะสิ้นสุด มักจะใช้ในการควบคุมการทำงานของโปรแกรมเป็นจำนวนรอบ มีวิธีการเขียนดังนี้ `range(start, stop[,step])`: โดย start เป็นค่าเริ่มต้น stop เป็นตำแหน่งค่าระยะสิ้นสุด และ step คือ ระยะห่างการเพิ่มดังตัวอย่างต่อไปนี้

```
>>> for i in range(5): print(i, end=' ')
0 1 2 3 4
>>> for i in range(1,5): print(i, end=' ')
1 2 3 4
>>> for i in range(1,5,2): print(i, end=' ')
1 3
>>> for i in range(5,1,-2): print(i, end=' ')
5 3
>>>
```

4.2 คำสั่ง for

คำสั่ง **for** statement เป็นการทำงานซ้ำๆ ตามจำนวนครั้งที่ระบุไว้อย่างแน่นอน เช่น การใช้ **for** statement ร่วมกัน `range()` โดยมีรูปแบบการเขียนดังนี้

```
for var in sequence:
    # statements
```

ตัวอย่างการใช้ **for** statement เป็นดังนี้

```
for x in range(0,5): print(x, end=' ')
```

ซึ่งจะได้ผลลัพธ์ดังนี้

```
>>>
0 1 2 3 4
>>>
```

ตัวอย่างการใช้ **for** statement ใน Python shell จะได้ผลลัพธ์ดังนี้

```
>>> name = 'Jan'
>>> for char in name: print(char, end=' ')
J a n
>>> mylist = ['Jan', 200, 4.5]
>>> for item in mylist: print(item, end = ' ')
Jan 200 4.5
>>>
```

ตัวอย่างการใช้ **for** statement ที่มีการผสมของการใช้ **if** ในการกำหนดเงื่อนไขการตัดสินใจ เป็นดังนี้

```
message = input('Enter a message: ')
vowels = 'AEIOU'
counter = 0
for char in message:
    if char.upper() in vowels:
        counter += 1
print('Vowels: ', counter)
```

ซึ่งจะได้ผลลัพธ์ดังนี้

```
Enter a message: Jantawan
Vowels:  3
>>>
```

4.3 คำสั่ง `while`

while Statement เป็นคำสั่งให้โปรแกรมทำงานวนซ้ำในขณะที่เงื่อนไขของการวนซ้ำนั้นยังคงเป็นจริงอยู่ และเมื่อเงื่อนไขเป็นเท็จจะสิ้นสุดการทำงานวนซ้ำนั้นทันที ดังนั้นจึงต้องมีตัวควบคุมในการเพิ่มค่าไปเรื่อยๆ จนเงื่อนไขเป็นเท็จ

while Statement มีลักษณะการเขียน 3 รูปแบบ คือ แบบ Pre-Test แบบ Post-Test และแบบ Mid-Test

รูปแบบการเขียน **while** statement แบบ Pre-Test เป็นดังนี้คือ

```
while expression:
    # statements
```

และมีตัวอย่างการใช้ **while** statement เป็นดังนี้

```
x = 0
while x < 10:
    print(x)
    x = x + 1
```

ซึ่งได้ผลลัพธ์เช่นนี้

```
0
1
2
3
4
5
6
7
8
9
```

รูปแบบการเขียน **while** statement แบบ Post-Test เป็นดังนี้คือ

```
while True:
    # statements
    if Boolean_Expression: break
```

และมีตัวอย่างการใช้ **while** statement เป็นดังนี้

```
i = 10
while True:
    if i \% 8 == 0: print( i, "   8    ")
    if i \% 4 == 0: print( i, "   4    ")
    if i \% 2 == 0: print( i, "   2    ")
    i -= 2
    if i <= 0: break
```

ซึ่งได้ผลลัพธ์เช่นนี้

```
10    2
8     8
8     4
8     2
6     2
4     4
4     2
2     2
>>>
```

รูปแบบการเขียน **while** statement แบบ Mid-Test เป็นดังนี้คือ

```
while expression:
    # statements 1
    if Boolean_Expression: break
    # statements 2
```

และมีตัวอย่างการใช้ **while** statement เป็นดังนี้

```
x = 0
while True:
    print(x)
    if x == 10: break
    x = x +1
```

ซึ่งได้ผลลัพธ์เช่นนี้

```
0
1
2
3
4
5
6
7
8
9
10
```

4.4 แบบฝึกหัด

1. สร้างฟังก์ชันหาค่าของ $3^1 + 3^2 + 3^3 + 3^4 + 3^5$ โดยใช้ For loop
2. สร้างฟังก์ชันให้ผู้ใช้ป้อนค่า x แล้วนำค่า x มาคำนวณ $x^1 + x^2 + x^3 + x^4 + x^5$
3. สร้างฟังก์ชันให้ผู้ใช้ป้อนค่า n และให้แสดงเลขเริ่มที่ n โดยลดลงทีละหนึ่ง โดยใช้ while loop
4. สร้างฟังก์ชันให้ผู้ใช้ป้อนตัวเลข แล้วหาว่ามีเลขใดที่สามารถหารเลขที่ผู้ใช้ป้อนได้ลงตัว เช่น ผู้ใช้ป้อนตัวเลข 4 จะมี 1 2 4 ที่หารเลข 4 ลงตัว
5. สร้างฟังก์ชันคำนวณปีเกิด คศ. เป็น 12 ราศีปีนักษัตร
6. สร้างฟังก์ชันรับจำนวนเงินมาหนึ่งค่า แล้วแลกเปลี่ยนธนบัตร 100 บาท 50 บาท 20 บาท เหรียญ 10 บาท 5 บาท และ 1 บาท

บรรณานุกรม

Barry, P. (2016). *Head first python: A brain-friendly guide*. Sebastopol, CA, USA: O'Reilly Media.

Beazley, D., & Jones, B. K. (2013). *Python cookbook*. Sebastopol, CA, USA: O'Reilly Media.

Bouras, A. S. (2019). *Python and algorithmic thinking for the complete beginner (2nd edition): Learn to think like a programmer*. Independently published.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. Cambridge, MA, USA: The MIT Press.

Downey, A. B. (2015). *Think python: How to think like a computer scientist*. Sebastopol, CA, USA: O'Reilly Media.

Foundation, P. S. (2019, January). *Python.* Retrieved from `https://www.python.org/`

Guido, V. R. (2019, January). *Guido van rossum - personal home page.* Retrieved from `https://gvanrossum.github.io//help.html`

Lubanovic, B. (2015). *Introducing python: Modern computing in simple packages*. Sebastopol, CA, USA: O'Reilly Media.

Lutz, M. (2011). *Programming python: Powerful object-oriented programming*. Sebastopol, CA, USA: O'Reilly Media.

Lutz, M. (2013). *Learning python*. Sebastopol, CA, USA: O'Reilly Media.

Lutz, M. (2014). *Python pocket reference: Python in your pocket*. Sebastopol, CA, USA: O'Reilly Media.

Ramalho, L. (2015). *Fluent python: Clear, concise, and effective programming*. Sebastopol, CA, USA: O'Reilly Media.

Shuup. (2019, April). *25 of the most popular python and django websites.* Retrieved from `https://www.shuup.com/django/25-of-the-most-popular-python-and-django-websites/`

TIOBE. (2019, August). *The python programming language*. Retrieved from **https://www.tiobe.com/tiobe-index/python/**