Data Science Programming

DS – 270702

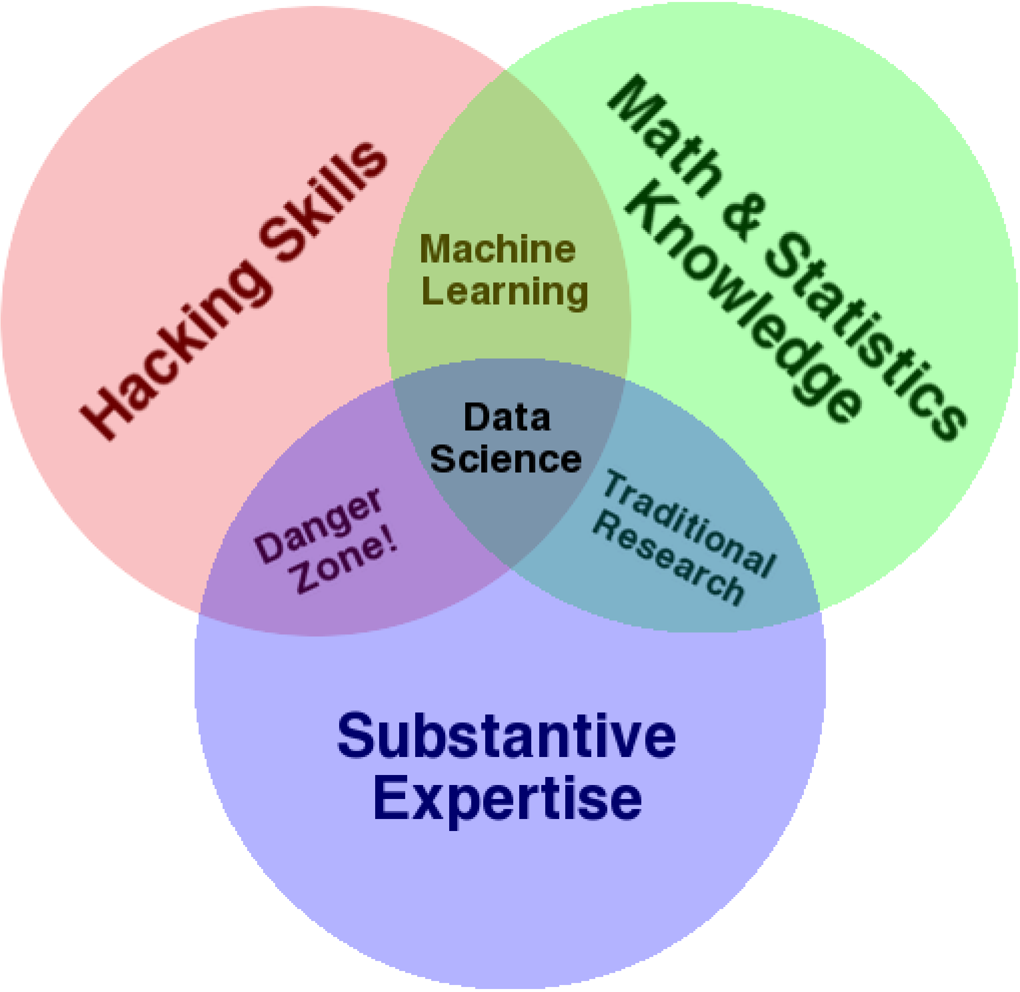
Part 2

Asst. Prof. **Paskorn Champrasert,** PhD.

paskorn@cmu.ac.th
www.OASYS-lab.com

OASYS Research Group,
Faculty of Engineering
Chiang Mai University

Data Science



https://jakevdp.github.io/PythonDataScienceHandbook/00.00-preface.html

Content

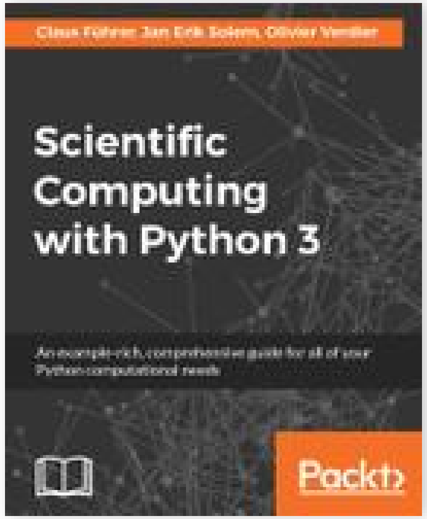
- Data Representation in Computer
- Program Flow
- Python Coding Style
- Basic Data Types
- Data type conversion
- Operation with Operator and Operand
- Input and Output

```python
"""
Plot a terrain model and a polar plot side by side.

"""

# Third party imports
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm
import matplotlib.colors
import mpl_toolkits.mplot3d  # Needed for 3D pylint: disa


# %% Plot final terrain model

# pylint: disable=no-member
plt.style.use('dark_background')

def generate_polar_plot():
    """Generate an example polar slice plot."""
    # Compute pie slices
    n_slices = 20
    theta = np.linspace(0.0, 2 * np.pi, n_slices, endpoin
    radii = 10 * np.random.rand(n_slices)
    width = np.pi / 4 * np.random.rand(n_slices)

    fig, ax = plt.subplots(figsize=(15, 6))
    fig.patch.set_facecolor('#395979')
    ax1 = plt.subplot(1, 2, 2, projection='polar')
    ax1.set_facecolor('#395979')
    bars = ax1.bar(theta, radii, width=width, bottom=0.0)

    # Use custom colors and opacity
    for radius, plot_bar in zip(radii, bars):
        plot_bar.set_facecolor(plt.cm.viridis(radius / 10
        plot_bar.set_alpha(0.5)


def generate_dem_plot():
    """Generate a 3D reprisentation of a terrain DEM."""
    dem_path = 'jacksboro_fault_dem.npz'
    with np.load(dem_path) as dem:
        z_data = dem['elevation']
        nrows, ncols = z_data.shape
```

Name	Type	Size	Value
data	Array of str128	(3, 3)	ndarray object of numpy module
df	DataFrame	(2, 2)	Column names: Col1, Col2
filename	str	1	/Users/Documents/spyder/spyder/tests/test_dont_use.py
i	Array of uint32	(10, 10)	[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]
li	list	5	['abcd', 745, 2.23, 'efgh', 70.2]
r	float	1	6.46567886443
t	tuple	5	('abcd', 745, 2.23, 'efgh', 70.2)
tinylist	list	2	[123, 'efgh']

Q08: Submit Google Colab. File (ipynb)

Q09: Using Spyder (submit test1.py)


Claus Führer Jan Erik Solem, Olivier Verdier
Scientific Computing with Python 3
An example-rich, comprehensive guide for all of your Python computational needs
Packt>

Scientific Computing with Python 3

★★★★★ 1 REVIEW

by Olivier Verdier, Jan Erik Solem, Claus Führer

Publisher: Packt Publishing

Release Date: December 2016

ISBN: 9781786463517

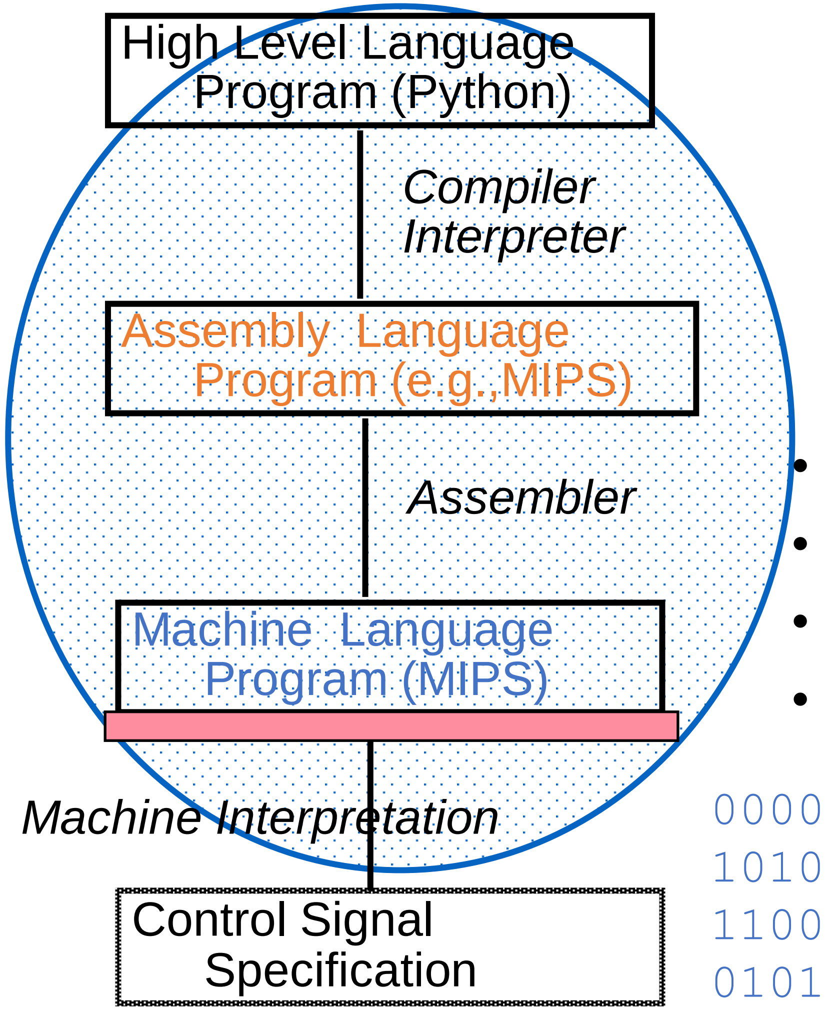
Topic: Scipy

Book Description

An example-rich, comprehensive guide for all of your Python computational needs

About This Book

- Your ultimate resource for getting up and running with Python numerical computations
- Explore numerical computing and mathematical libraries using Python 3.x code with SciPy and NumPy modules
- A hands-on guide to implementing mathematics with Python, with complete coverage of all the key concepts



High Level Language Program (Python)

Compiler
Interpreter

Assembly Language Program (e.g.,MIPS)

Assembler

Machine Language Program (MIPS)

Machine Interpretation

Control Signal Specification

```
temp = a
a = b
b = temp
```

```
• lw     $to,     0($2)
• lw     $t1,     4($2)
• sw     $t1,     0($2)
• sw     $t0,     4($2)
```

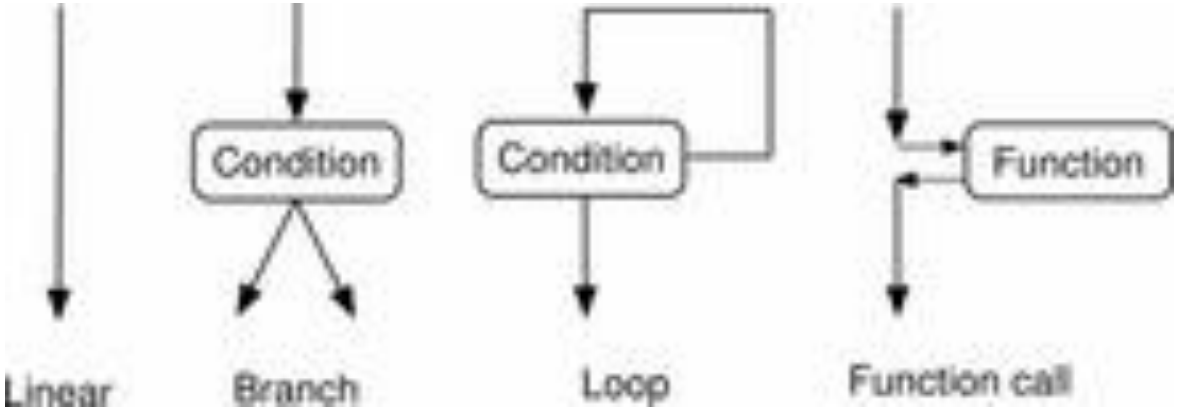
```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

ASCII TABLE

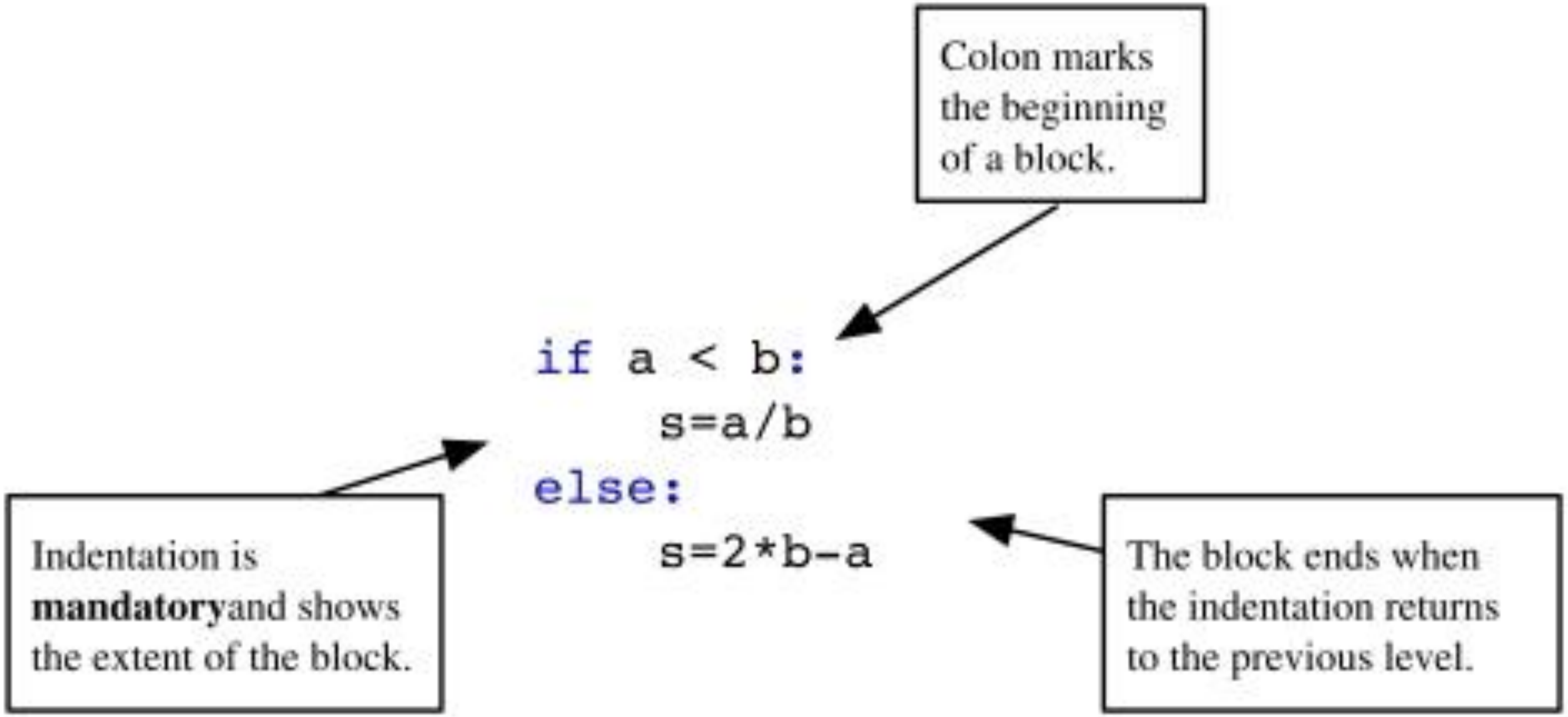
American Standard Code
for Information Interchange

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Program Flow



Python Style



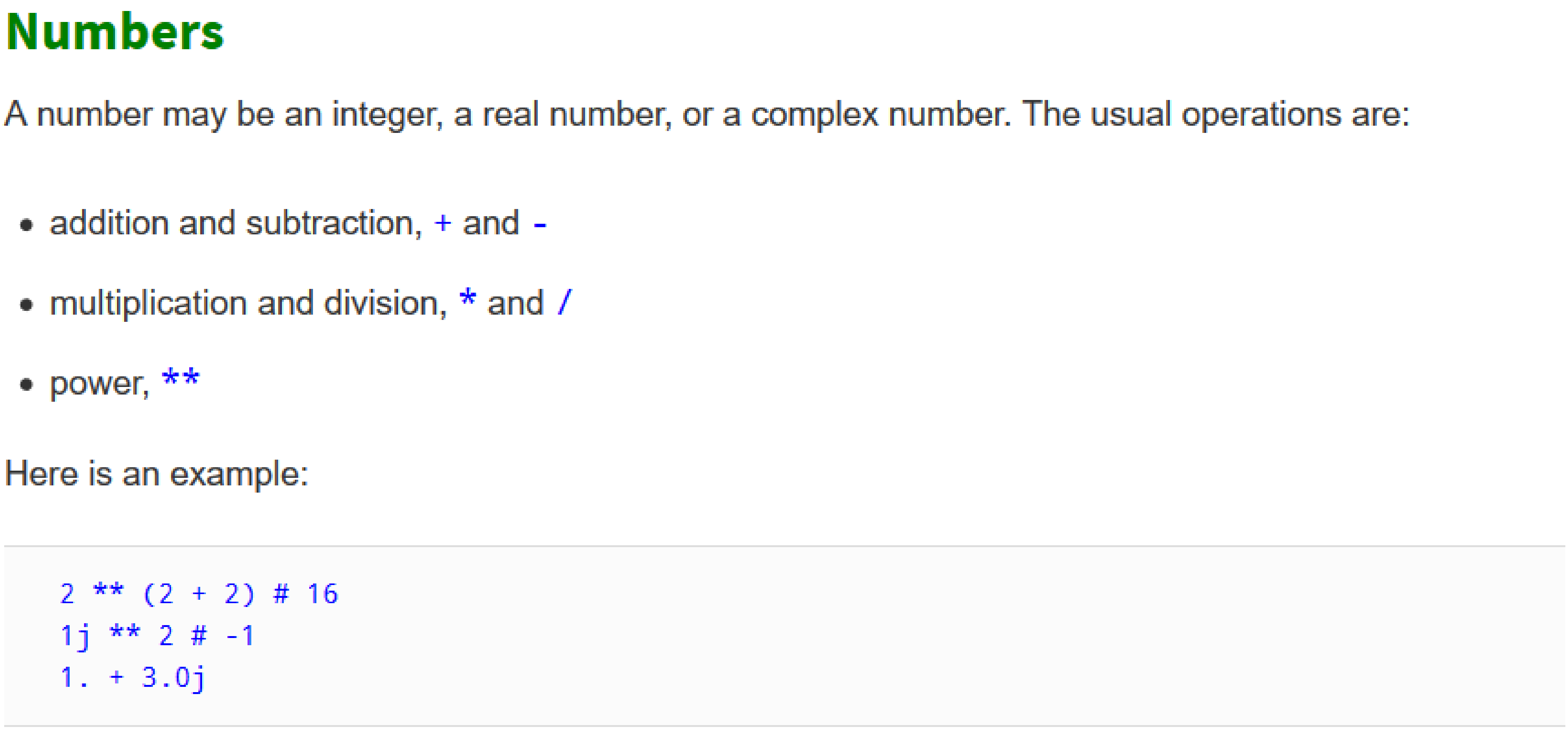
```
if a < b:
    s=a/b
else:
    s=2*b-a
```

Colon marks the beginning of a block.

Indentation is **mandatory**and shows the extent of the block.

The block ends when the indentation returns to the previous level.

Basic Data Types



Numbers

A number may be an integer, a real number, or a complex number. The usual operations are:

- addition and subtraction, + and -
- multiplication and division, * and /
- power, **

Here is an example:

```
2 ** (2 + 2) # 16
1j ** 2 # -1
1. + 3.0j
```

Strings

Strings are sequences of characters, enclosed by simple or double quotes:

```
'valid string'
"string with double quotes"
"you shouldn't forget comments"
'these are double quotes: ".." '
```

You can also use triple quotes for strings that have multiple lines:

```
"""This is
 a long,
 long string"""
```

Boolean expressions

A Boolean expression is an expression that may have the value `True` or `False`. Some common operators that yield conditional expressions are as follow:

- Equal, `==`
- Not equal, `!=`
- Less than, Less than or equal to, `<` , `<=`
- Greater than, Greater than or equal to, `>` , `>=`

```
2 >= 4  # False
2 < 3 < 4 # True
2 < 3 and 3 < 2 # False
2 != 3 < 4 or False # True
2 <= 2 and 2 >= 2 # True
not 2 == 3 # True
not False or True and False # True!
```

One combines different Boolean values with `or` and `and`. The keyword `not` , gives the logical negation of the expression that follows. Comparisons can be chained so that, for example, `x < y < z` is equivalent to `x < y and y < z`. The difference is that `y` is only evaluated once in the first example. In both cases, `z` is not evaluated at all when the first condition, `x < y`, evaluates to `False`:

Data Type Conversion

- When integer data is calculated with float data the result is conversed to float

- Data type conversion :

  int() and float()

  - `a = int(100.50)    # a=100`
  - `b = float(30/3)    # b=10.0`

```
>>> print(99+1.23)
100.23
>>> i = 42
>>> print(i)
42
>>> type(i)
int
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
float
>>> print(9/3)
3.0
>>> type(9/3)
float
```

String Conversions

- `c = float(’30.5’) # c=30.5`
- `d = int(‘100’)     # d=100`

- Converse number to string

  str()

Q08: Submit Google Colab. File

```
>>> string = '123.456'
>>> type(string)
                         1
>>> f = float(string)
>>> type(f)
                         2
>>> string + 1

                         3

>>> f + 1
                         4
>>> print(int(string))

                         5

>>> print(int('1234'))
                         6
>>> print(int('Hello'))

                         7

>>> s = str(f)
>>> print(s)
                         8
>>> type(s)
                         9
```

String Conversions

- c = float(’30.5’) # c=30.5
- d = int(‘100’)   # d=100

- Converse number to string

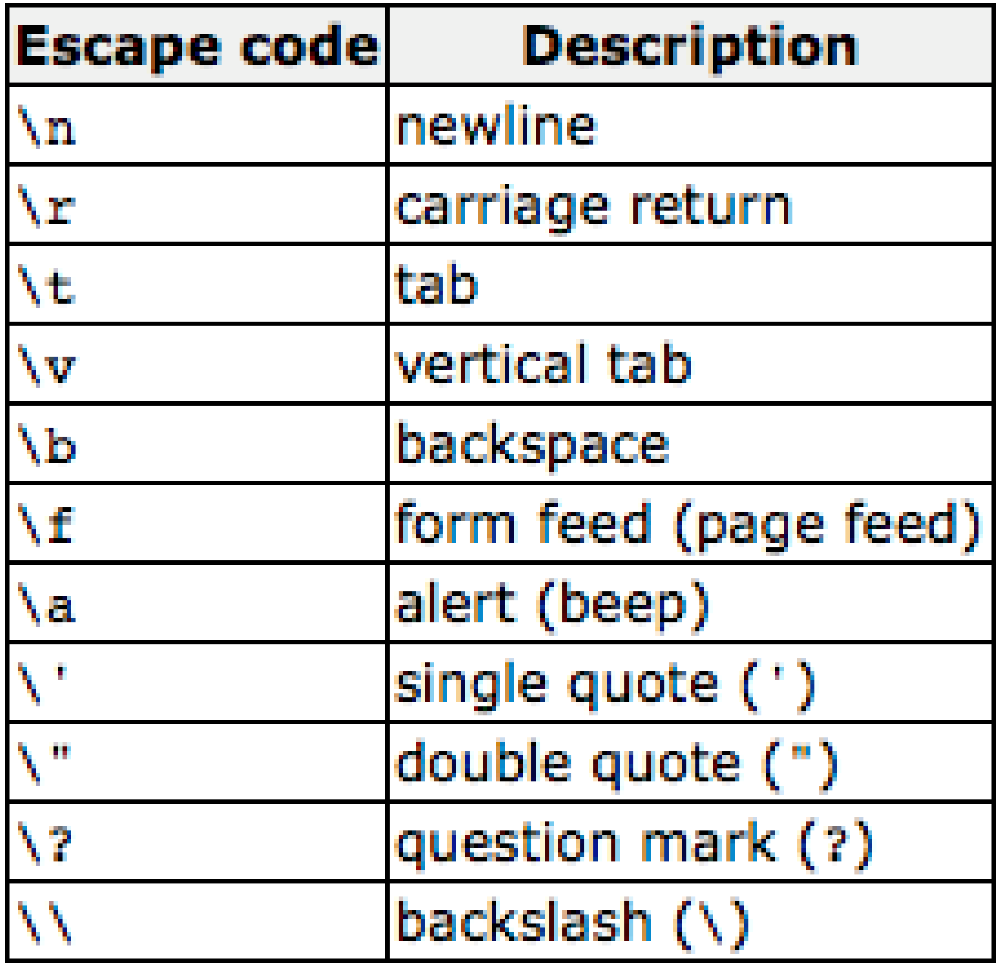
  str()

Q08: Submit Google Colab. File

```
>>> string = '123.456'
>>> type(string)
str
>>> f = float(string)
>>> type(f)
float
>>> string + 1
TypeError: can only concatenate str (not
"int") to str
>>> f + 1
124.456
>>> print(int(string))
ValueError: invalid literal for int() with
base 10: '123.456'
>>> print(int('1234'))
1234
>>> print(int('Hello'))
ValueError: invalid literal for int() with
base 10: 'Hello'
>>> s = str(f)
>>> print(s)
123.456
>>> type(s)
str
```

Special Characters

- Escape code



Escape code	Description
`\n`	newline
`\r`	carriage return
`\t`	tab
`\v`	vertical tab
`\b`	backspace
`\f`	form feed (page feed)
`\a`	alert (beep)
`\'`	single quote (')
`\"`	double quote (")
`\?`	question mark (?)
`\\`	backslash (\\)

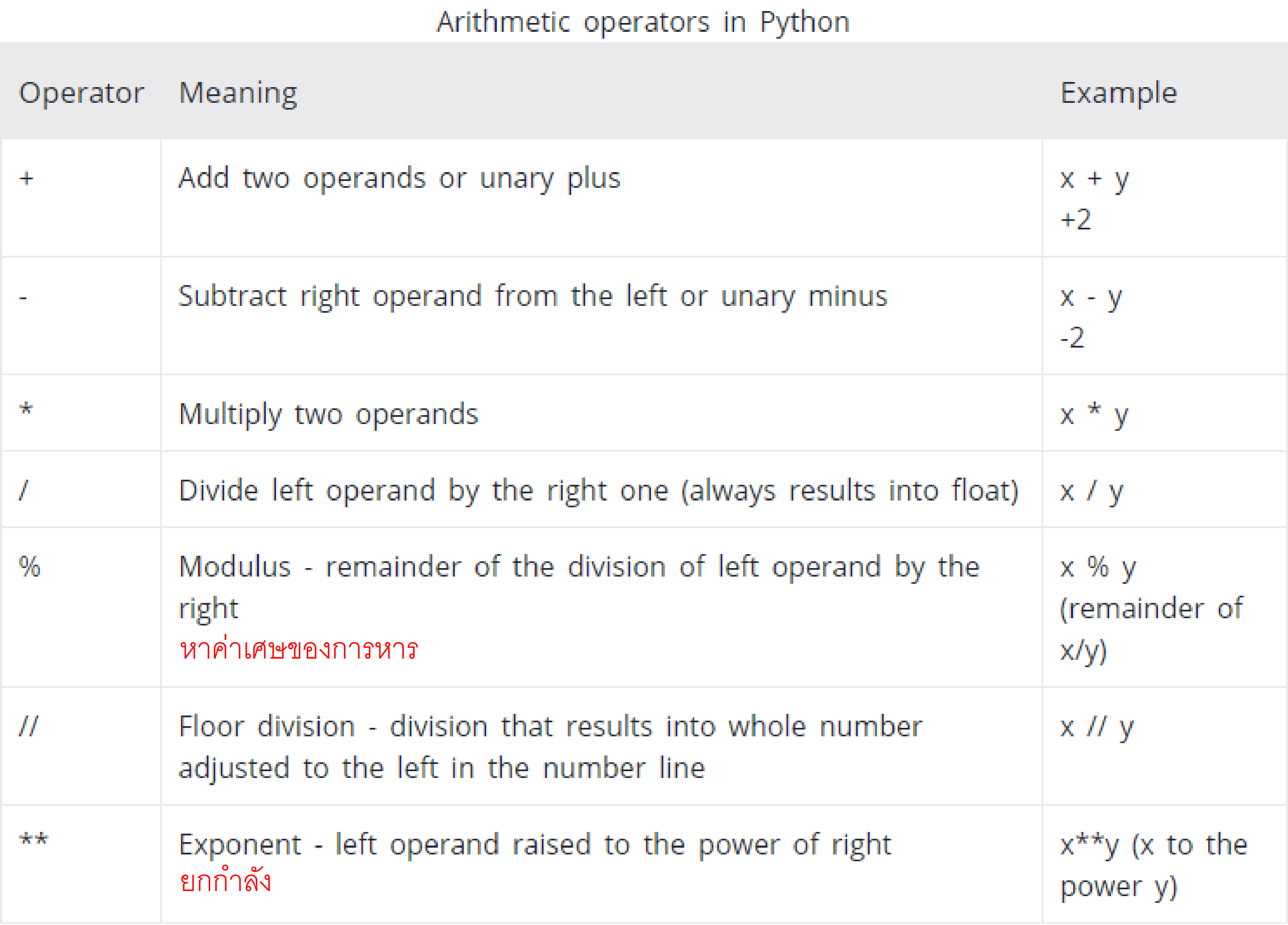
Operator and Operand

- Operator VS Operand

```
>>> 2 + 3
5
```

- Operator: +
- Operand: 2, 3
- Output : 5

Arithmetic Operators

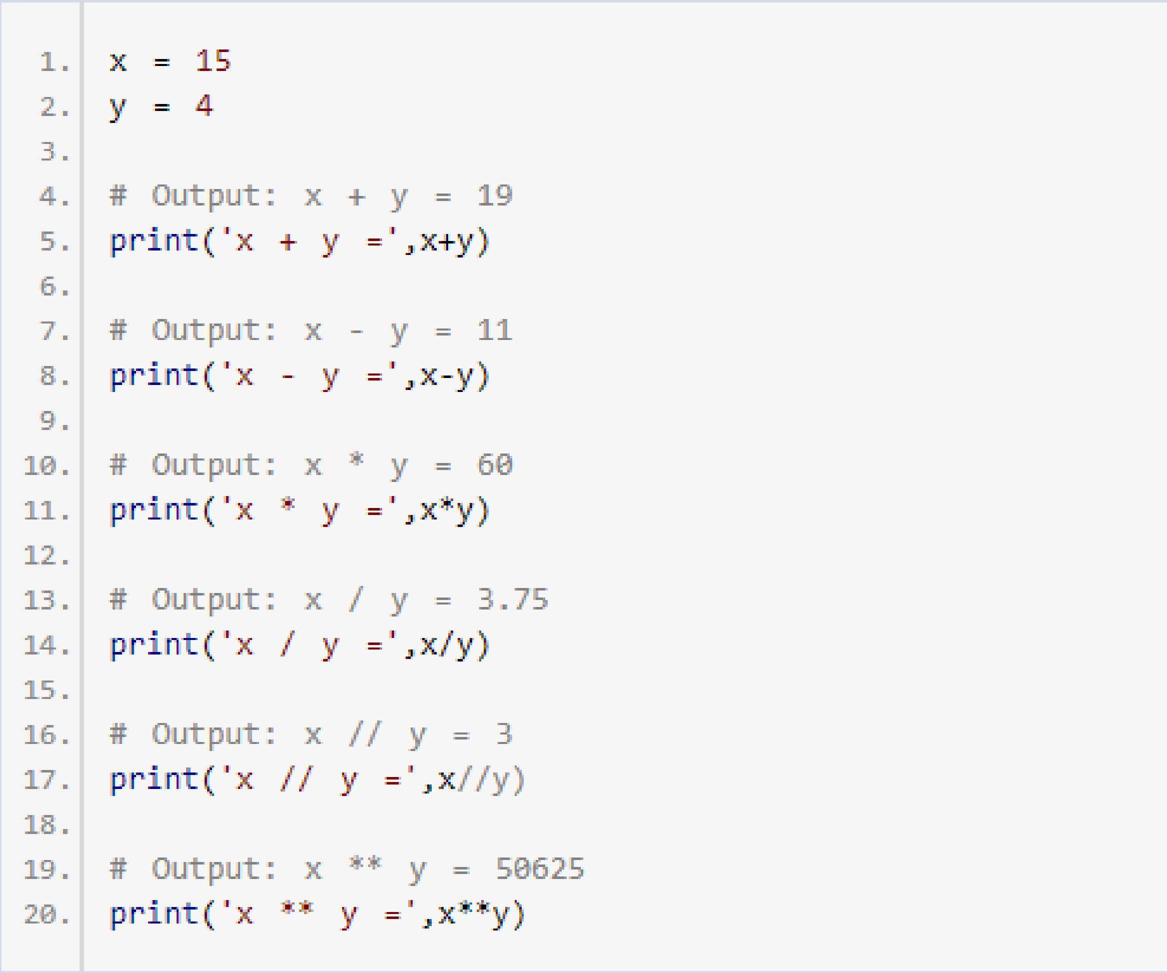


Arithmetic operators in Python

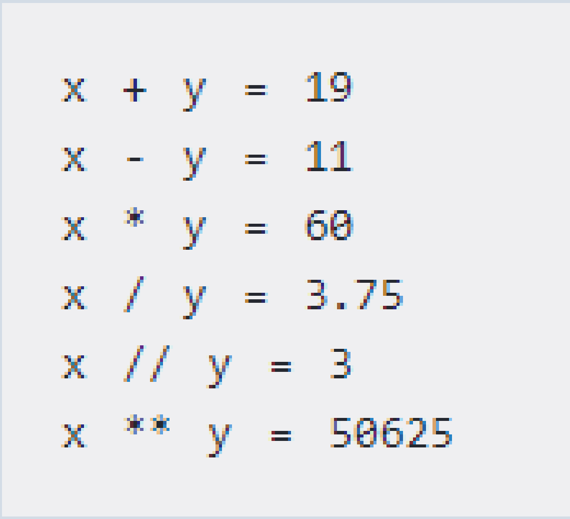
Operator	Meaning	Example
+	Add two operands or unary plus	x + y +2
-	Subtract right operand from the left or unary minus	x - y -2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right หาค่าเศษของการหาร	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right ยกกำลัง	x**y (x to the power y)

Arithmetic Operators

Example #1: Arithmetic operators in Python



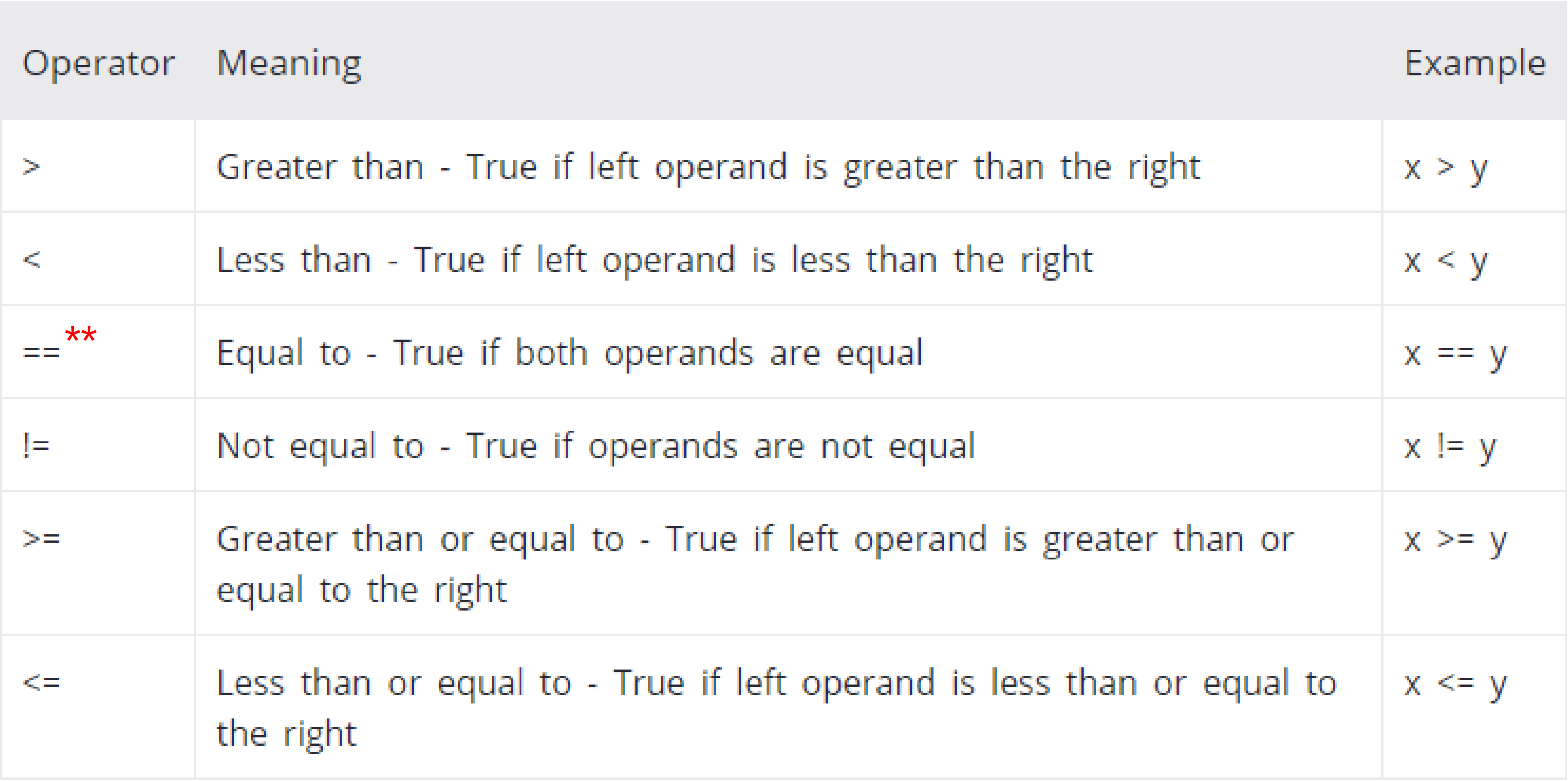
```python
x = 15
y = 4

# Output: x + y = 19
print('x + y =',x+y)

# Output: x - y = 11
print('x - y =',x-y)

# Output: x * y = 60
print('x * y =',x*y)

# Output: x / y = 3.75
print('x / y =',x/y)

# Output: x // y = 3
print('x // y =',x//y)

# Output: x ** y = 50625
print('x ** y =',x**y)
```



```
x + y = 19
x - y = 11
x * y = 60
x / y = 3.75
x // y = 3
x ** y = 50625
```

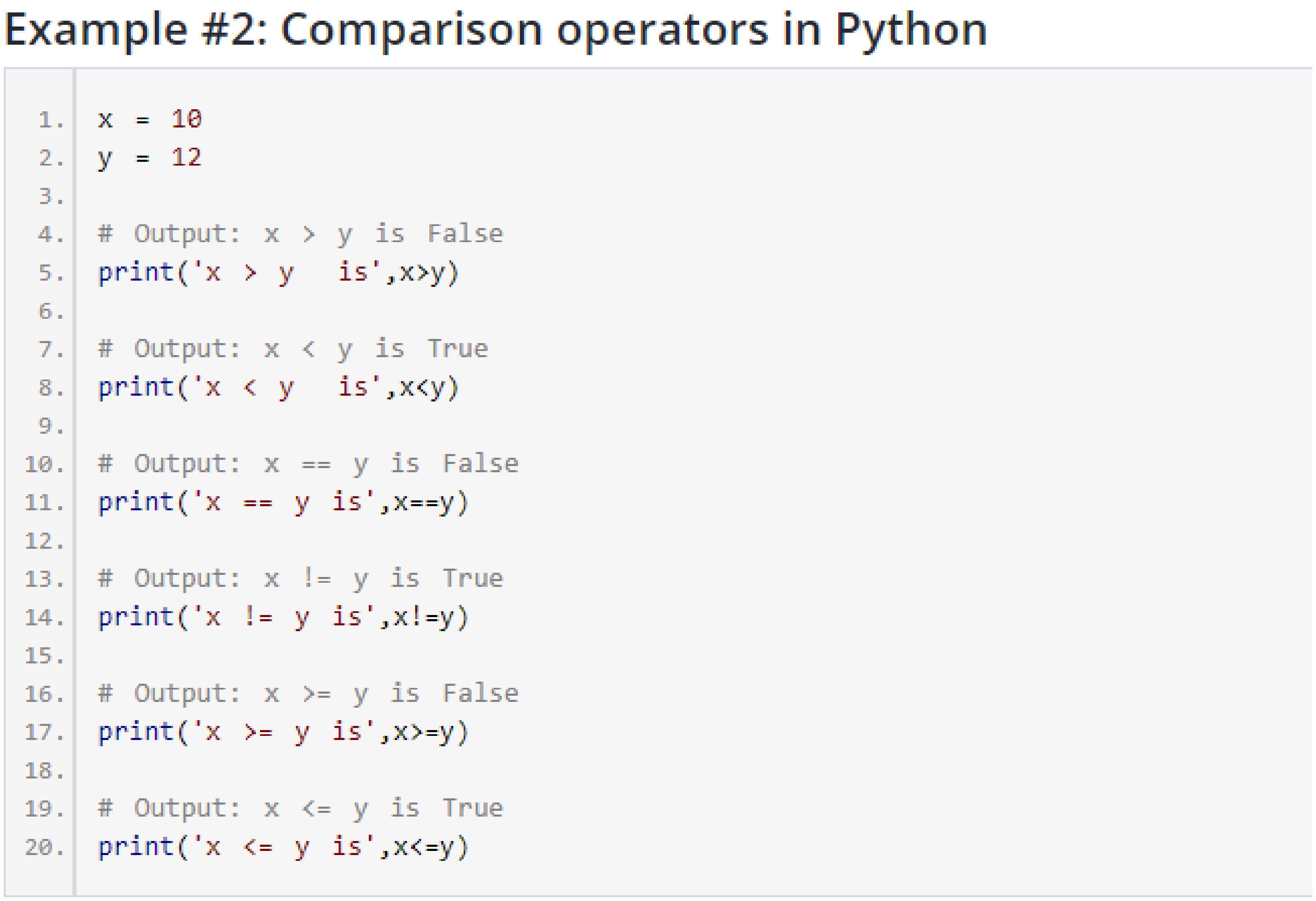
Comparison Operators

Comparision operators in Python (Result: True/False)



Operator	Meaning	Example
>	Greater than - True if left operand is greater than the right	x > y
<	Less than - True if left operand is less than the right	x < y
==**	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

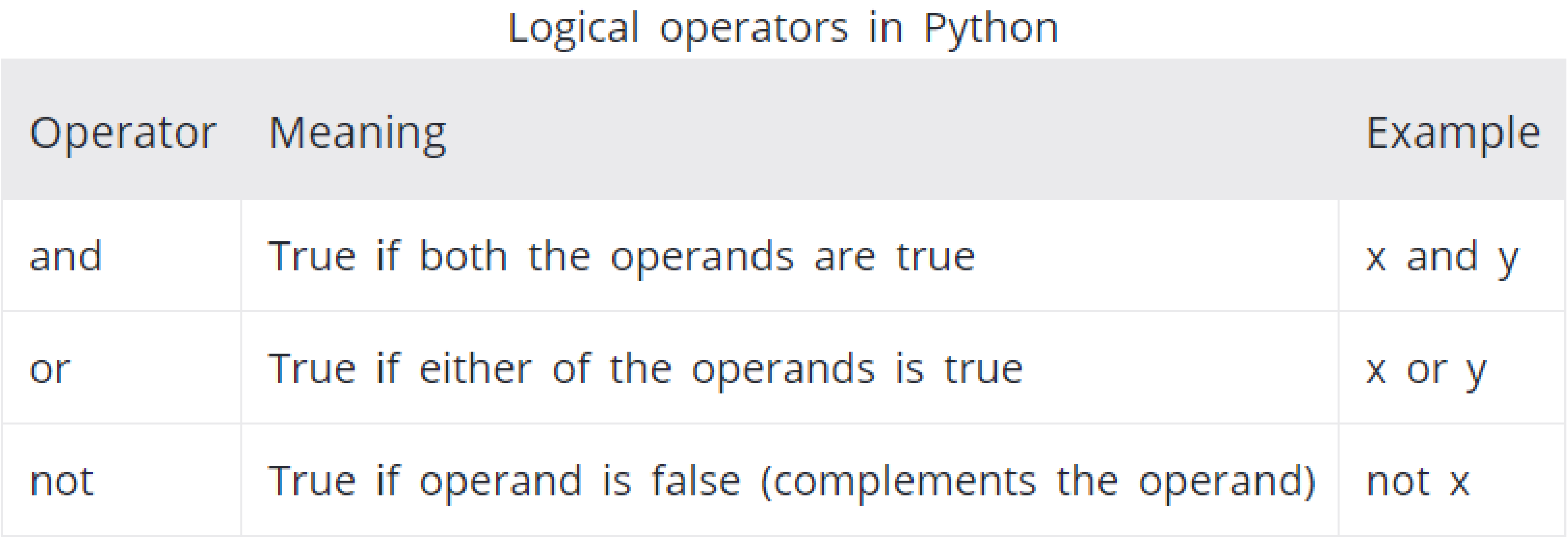
Comparison Operators



Example #2: Comparison operators in Python

```python
x = 10
y = 12

# Output: x > y is False
print('x > y  is',x>y)

# Output: x < y is True
print('x < y  is',x<y)

# Output: x == y is False
print('x == y is',x==y)

# Output: x != y is True
print('x != y is',x!=y)

# Output: x >= y is False
print('x >= y is',x>=y)

# Output: x <= y is True
print('x <= y is',x<=y)
```

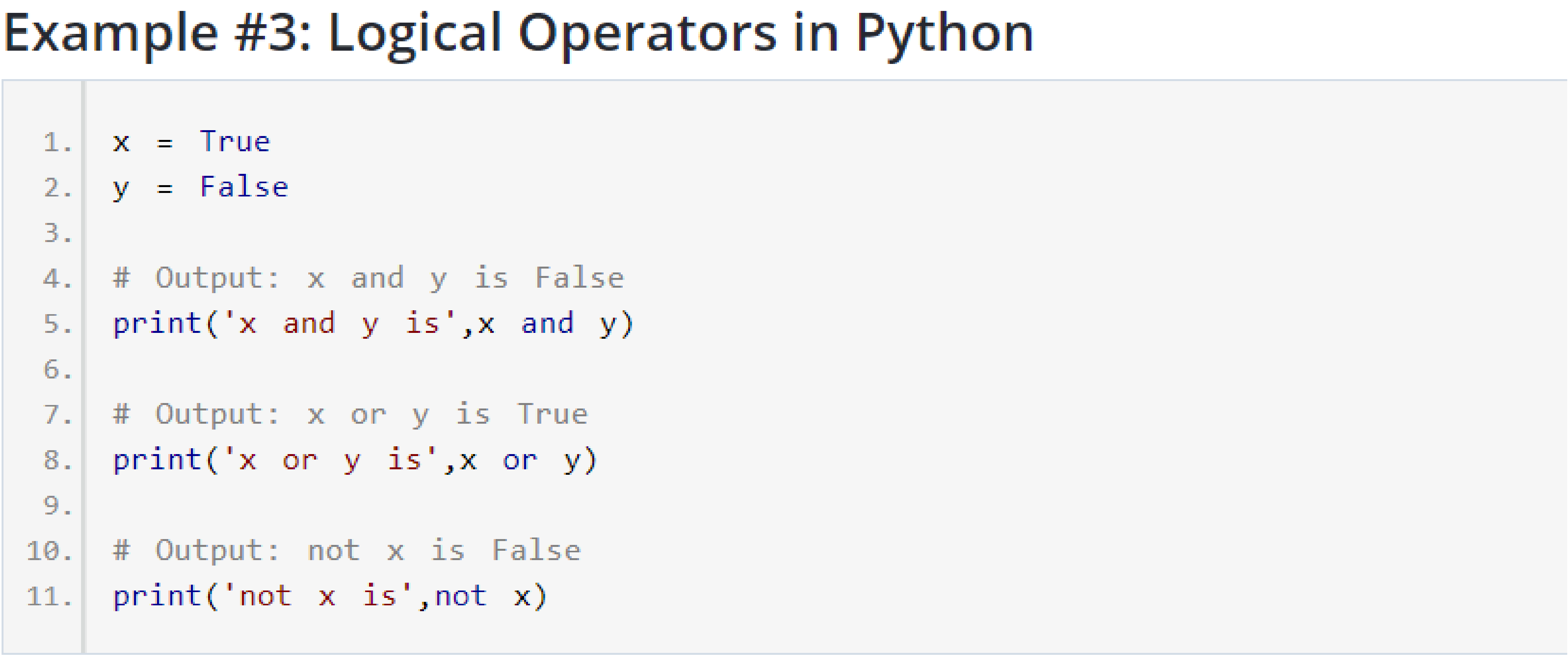
Logical Operators



Logical operators in Python

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

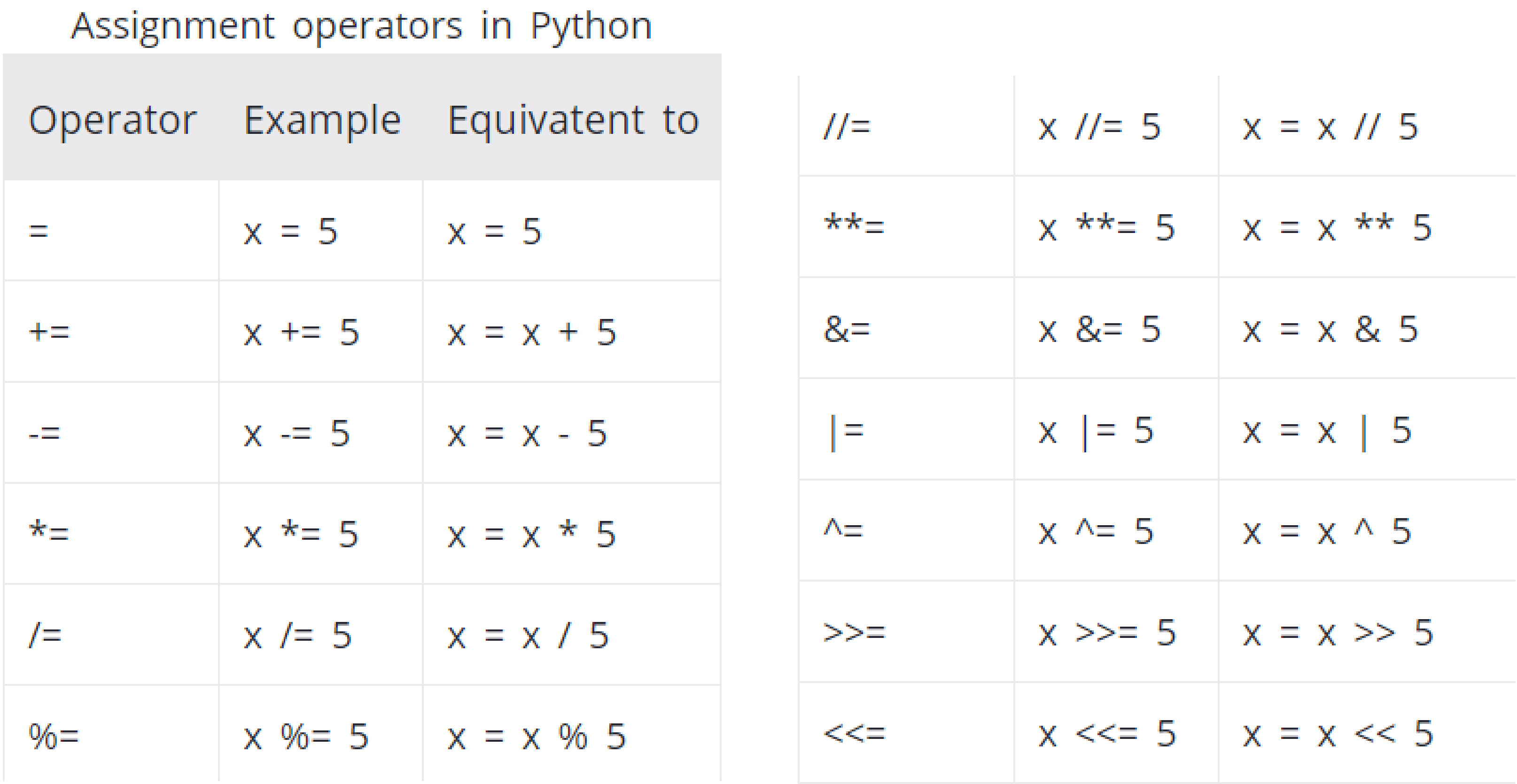
Logical Operators



Example #3: Logical Operators in Python

```python
x = True
y = False

# Output: x and y is False
print('x and y is',x and y)

# Output: x or y is True
print('x or y is',x or y)

# Output: not x is False
print('not x is',not x)
```

Assignment Operators



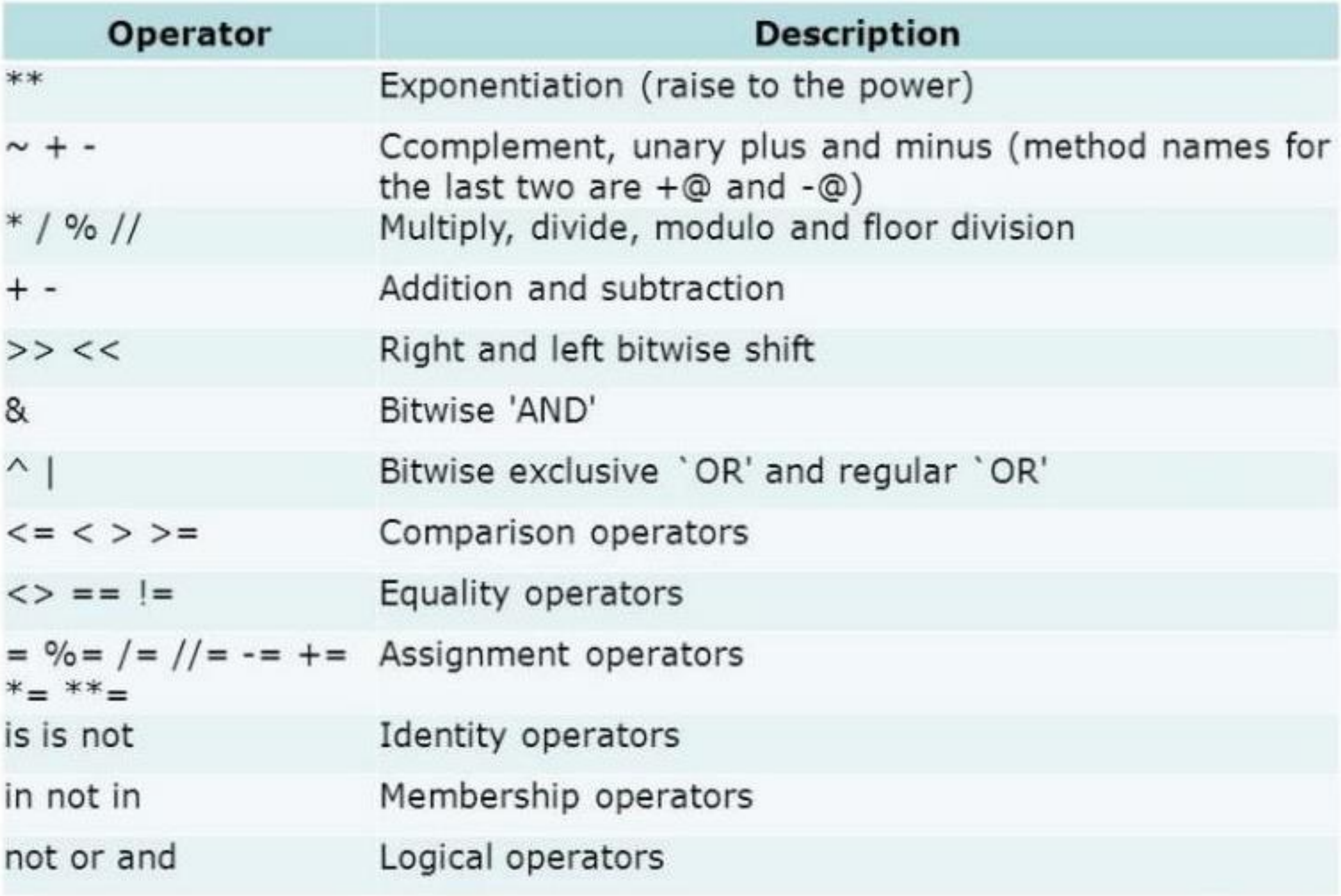
Assignment operators in Python

Operator	Example	Equivatent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
|=	x |= 5	x = x | 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

Python Operators Precedence

First

Later



Operator	Description
**	Exponentiation (raise to the power)
~ + -	Ccomplement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Example

Step 1: $y = \boxed{2 * 5} * 5 - 3 * 5 + 7$ (Leftmost multiplication)

2 * 5 is $\boxed{10}$

Step 2: $y = \boxed{10 * 5} - 3 * 5 + 7$ (Leftmost multiplication)

10 * 5 is $\boxed{50}$

Step 3: $y = 50 - \boxed{3 * 5} + 7$ (* before + -)

3 * 5 is $\boxed{15}$

Step 4: $y = \boxed{50 - 15} + 7$ (Leftmost + or -)

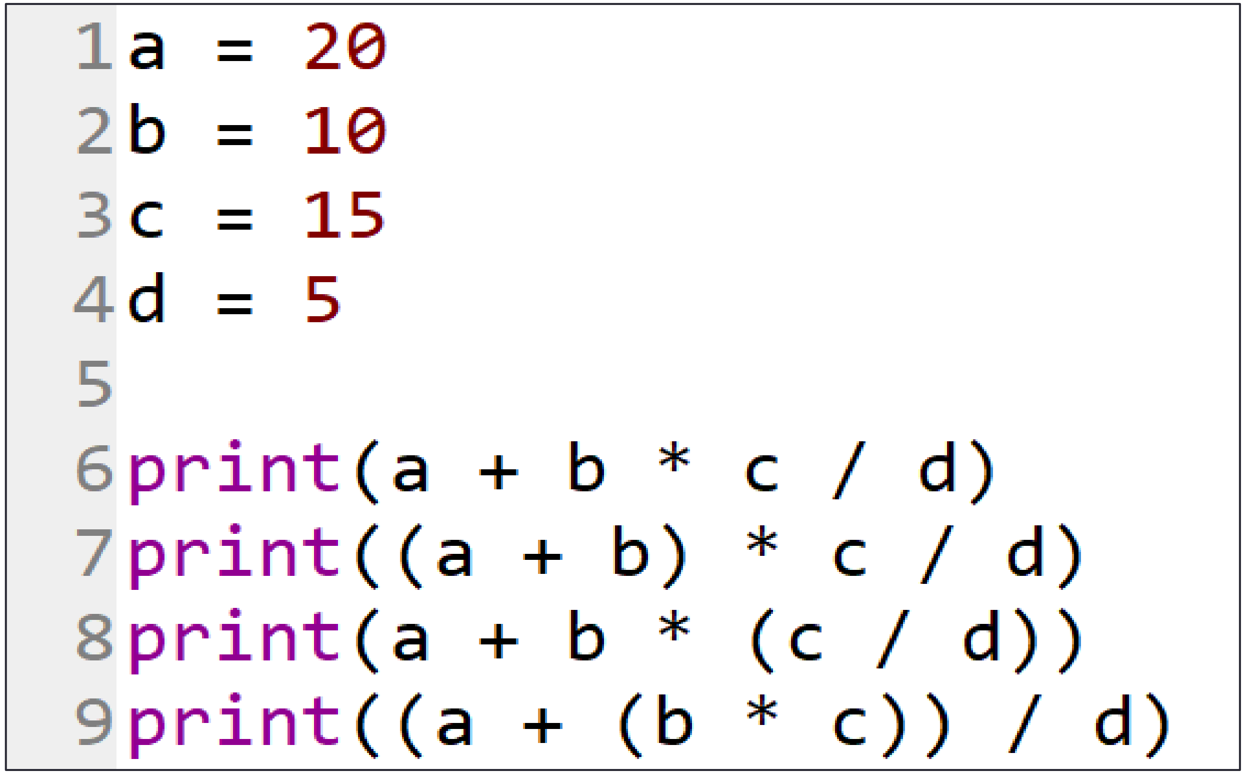
50 - 15 is $\boxed{35}$

Step 5: $y = 35 + 7$

35 + 7 is $\boxed{42}$

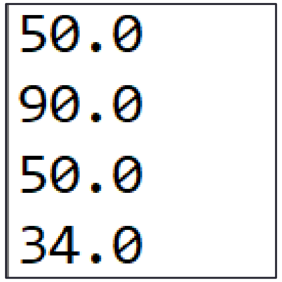
Step 6: $y = 42$

Operators Precedence



```
a = 20
b = 10
c = 15
d = 5

print(a + b * c / d)
print((a + b) * c / d)
print(a + b * (c / d))
print((a + (b * c)) / d)
```

Output:



```
50.0
90.0
50.0
34.0
```

More Math Operations with Python

- Trigonometric – `sin, cos, tan, asin, acos,` …
- Exponential and Logarithmic – `exp, log, log2, log10,` …
- Power – `pow, sqrt,` …
- Rounding – `ceil, floor`
- Others – `abs,` …
- Constant - `pi, e`

Example

```python
# importing built-in module math
import math

# using square root(sqrt) function contained
# in math module
print(math.sqrt(25))
# pi and e value
print(math.pi, math.e)
# 2 radians = 114.59 degreees
print(math.degrees(2))
# 60 degrees = 1.04 radians
print(math.radians(60))
# Sine of 2 radians
print(math.sin(2))
# Cosine of 0.5 radians
print(math.cos(0.5))
# 1 * 2 * 3 * 4 = 24
print(math.factorial(4))
# ceil
print(math.ceil(5.49))
```

```
5.0
3.141592653589793 2.718281828459045
114.59155902616465
1.0471975511965976
0.9092974268256817
0.8775825618903728
24
6
```

Input / Output

Output using print()

- `print` : Produces text output on the console.

- Syntax:

  `print ("`***Message***`“)`

  `print (`***Expression***`)`

  - Prints the given text message or expression value on the console, and moves the cursor down to the next line.

  `print (`***Item1***`,` ***Item2***`,` …`,` ***ItemN)***

  - Prints several messages and/or expressions on the same line.

- Examples:

```
print ("Hello, world!“)
age = 40
print ("You have", 60 - age, "years until retirement“)
```

Output:

```
Hello, world!
You have 20 years until retirement
```

input()

```
    a = input()
```

Data conversion for a:String to number

```
    a = int(input())      #  Integer
    a = float(input())   #  Float
```

input()

```
a = input(‘Please enter your name:’)
print(‘My name is’,a)
```

- Output

```
Please enter your name: John Doe
My name is John Doe
```

input() : Split

```
fname, lname = input(‘Enter yourname:’).split()
```

- Example:

```
Enter your name: Jurgen Klopp
```

- “Jurgen Klopp”
  - “Jurgen” => fname
  - “Klopp” => lname

But,

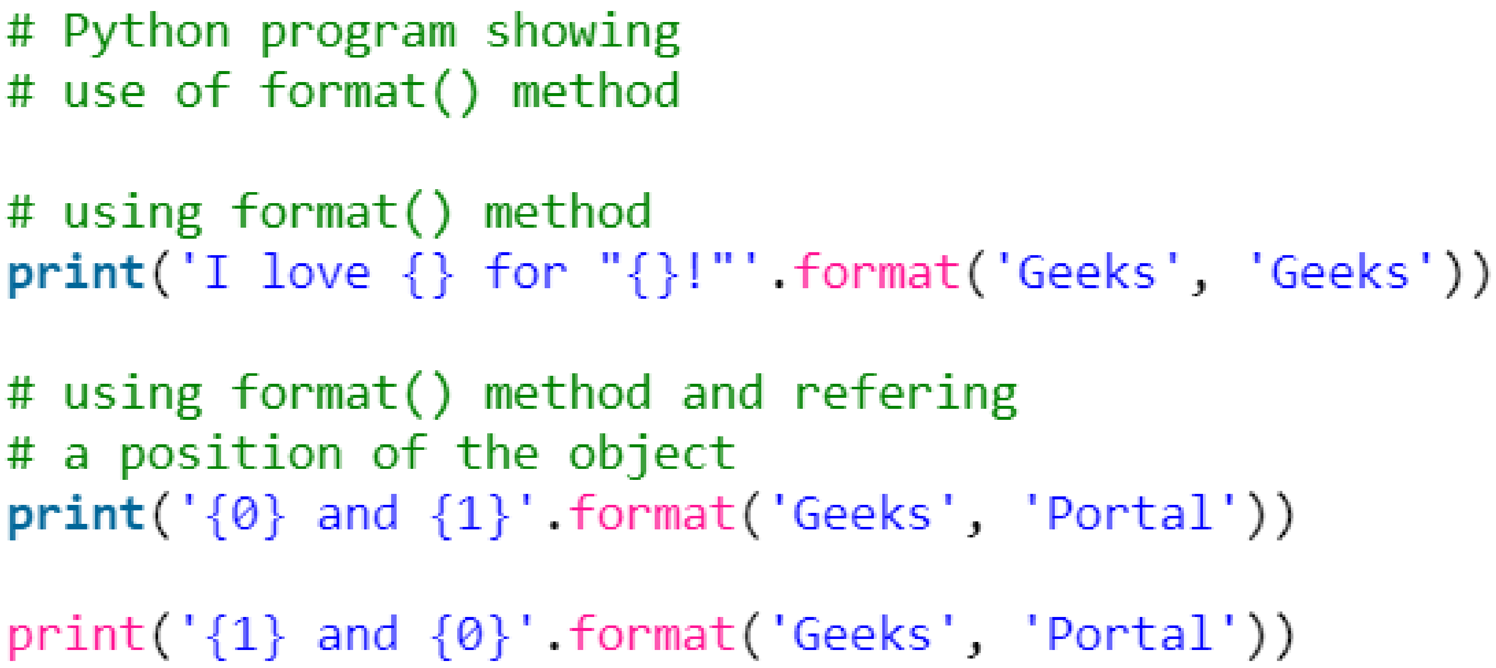
```
name = input(‘Enter your name:’)
```

“Jurgen Klopp” => name

Formatting output using `format()`

Q09 : Learning by Doing: What is the *format* function ?

Using Spyder to make the code and see the result



```
# Python program showing
# use of format() method

# using format() method
print('I love {} for "{}!"'.format('Geeks', 'Geeks'))

# using format() method and refering
# a position of the object
print('{0} and {1}'.format('Geeks', 'Portal'))

print('{1} and {0}'.format('Geeks', 'Portal'))
```

Data Science Programming

DS – 270702

Part 2B: Spyder IDE

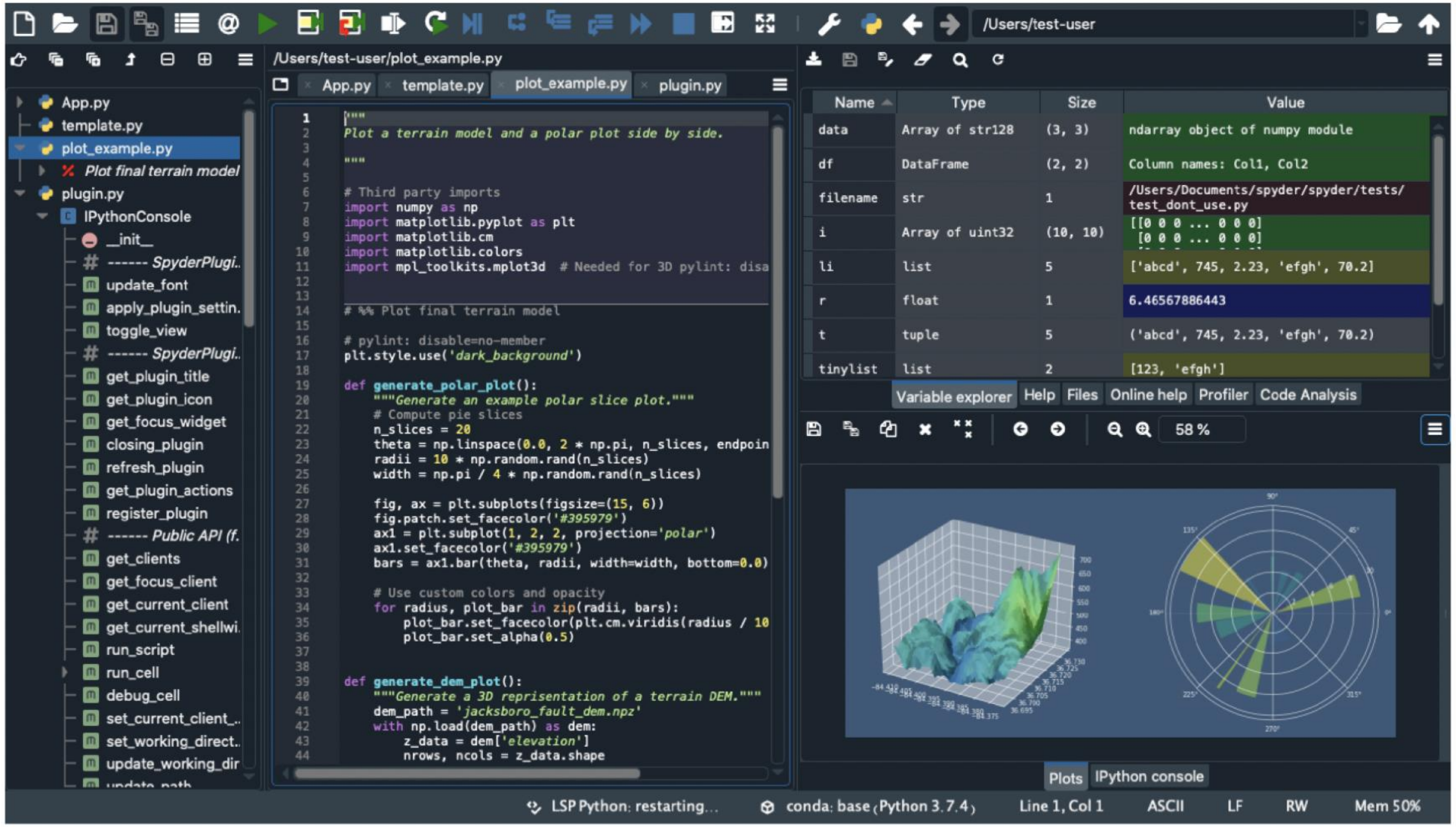
Asst. Prof. Paskorn Champrasert, PhD.

paskorn@cmu.ac.th
www.OASYS-lab.com

OASYS Research Group,
Faculty of Engineering
Chiang Mai University

SPYDER

The Scientific Python Development Environment



```python
"""
Plot a terrain model and a polar plot side by side.

"""

# Third party imports
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm
import matplotlib.colors
import mpl_toolkits.mplot3d  # Needed for 3D pylint: disa


# %% Plot final terrain model

# pylint: disable=no-member
plt.style.use('dark_background')

def generate_polar_plot():
    """Generate an example polar slice plot."""
    # Compute pie slices
    n_slices = 20
    theta = np.linspace(0.0, 2 * np.pi, n_slices, endpoin
    radii = 10 * np.random.rand(n_slices)
    width = np.pi / 4 * np.random.rand(n_slices)

    fig, ax = plt.subplots(figsize=(15, 6))
    fig.patch.set_facecolor('#395979')
    ax1 = plt.subplot(1, 2, 2, projection='polar')
    ax1.set_facecolor('#395979')
    bars = ax1.bar(theta, radii, width=width, bottom=0.0)

    # Use custom colors and opacity
    for radius, plot_bar in zip(radii, bars):
        plot_bar.set_facecolor(plt.cm.viridis(radius / 10
        plot_bar.set_alpha(0.5)


def generate_dem_plot():
    """Generate a 3D reprisentation of a terrain DEM."""
    dem_path = 'jacksboro_fault_dem.npz'
    with np.load(dem_path) as dem:
        z_data = dem['elevation']
        nrows, ncols = z_data.shape
```

Name	Type	Size	Value
data	Array of str128	(3, 3)	ndarray object of numpy module
df	DataFrame	(2, 2)	Column names: Col1, Col2
filename	str	1	/Users/Documents/spyder/spyder/tests/test_dont_use.py
i	Array of uint32	(10, 10)	[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]
li	list	5	['abcd', 745, 2.23, 'efgh', 70.2]
r	float	1	6.46567886443
t	tuple	5	('abcd', 745, 2.23, 'efgh', 70.2)
tinylist	list	2	[123, 'efgh']

Data Science Programming

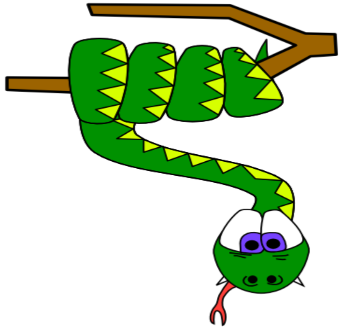
DS – 270702

Part 2c

Asst. Prof. Paskorn Champrasert, PhD.

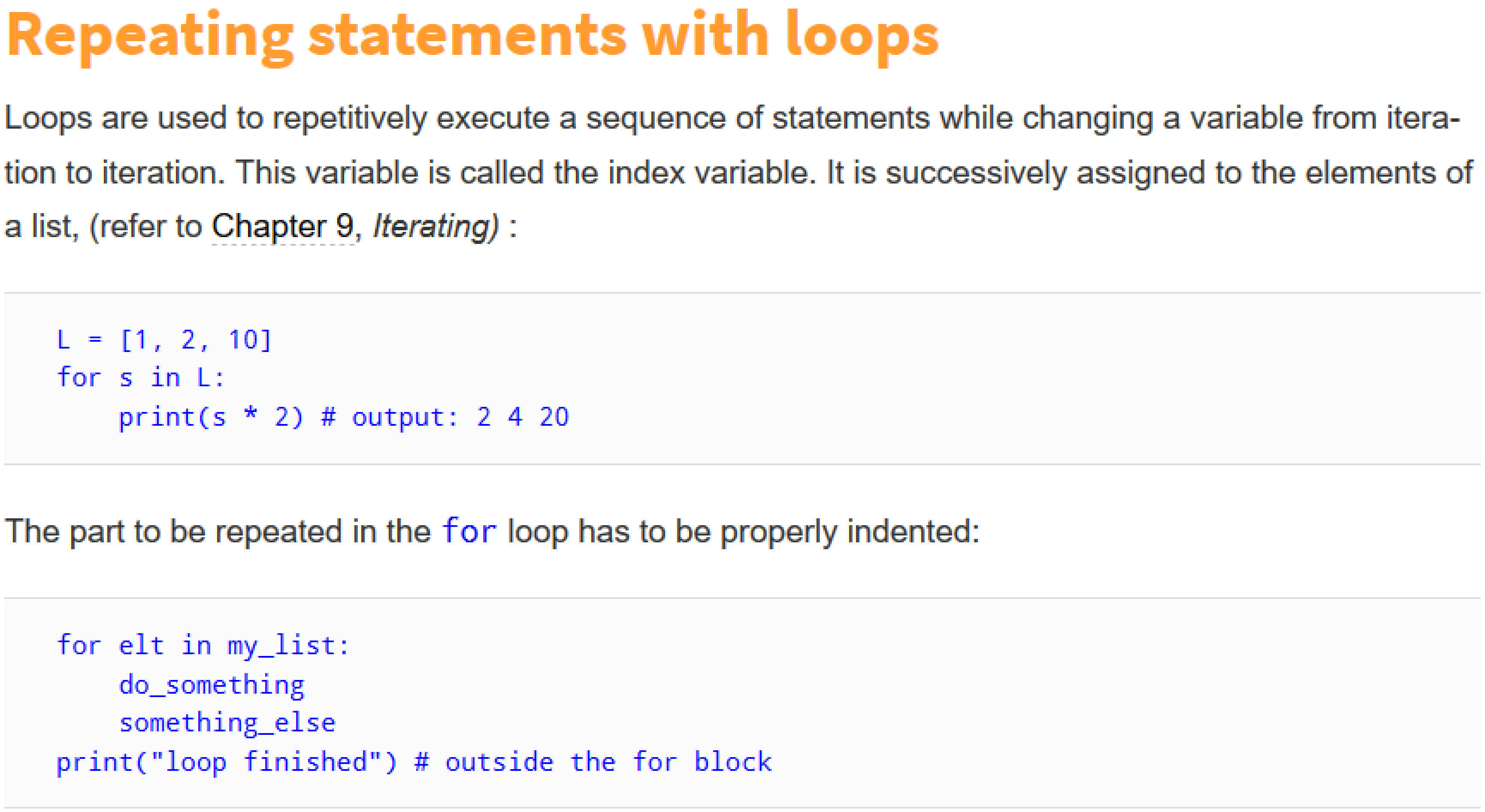
paskorn@cmu.ac.th
www.OASYS-lab.com

OASYS Research Group,
Faculty of Engineering
Chiang Mai University



Repetition (loops)
and Selection (if/else)

Loop



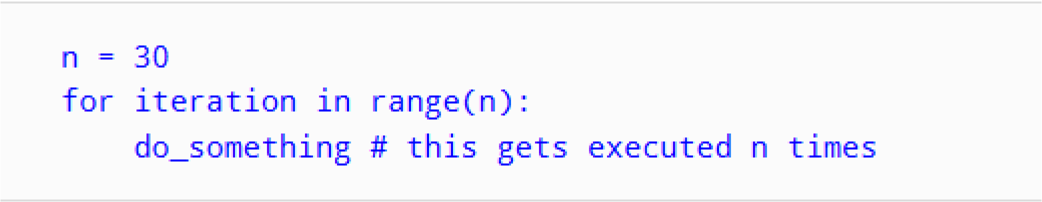
Repeating statements with loops

Loops are used to repetitively execute a sequence of statements while changing a variable from iteration to iteration. This variable is called the index variable. It is successively assigned to the elements of a list, (refer to Chapter 9, *Iterating)* :

```
L = [1, 2, 10]
for s in L:
    print(s * 2) # output: 2 4 20
```

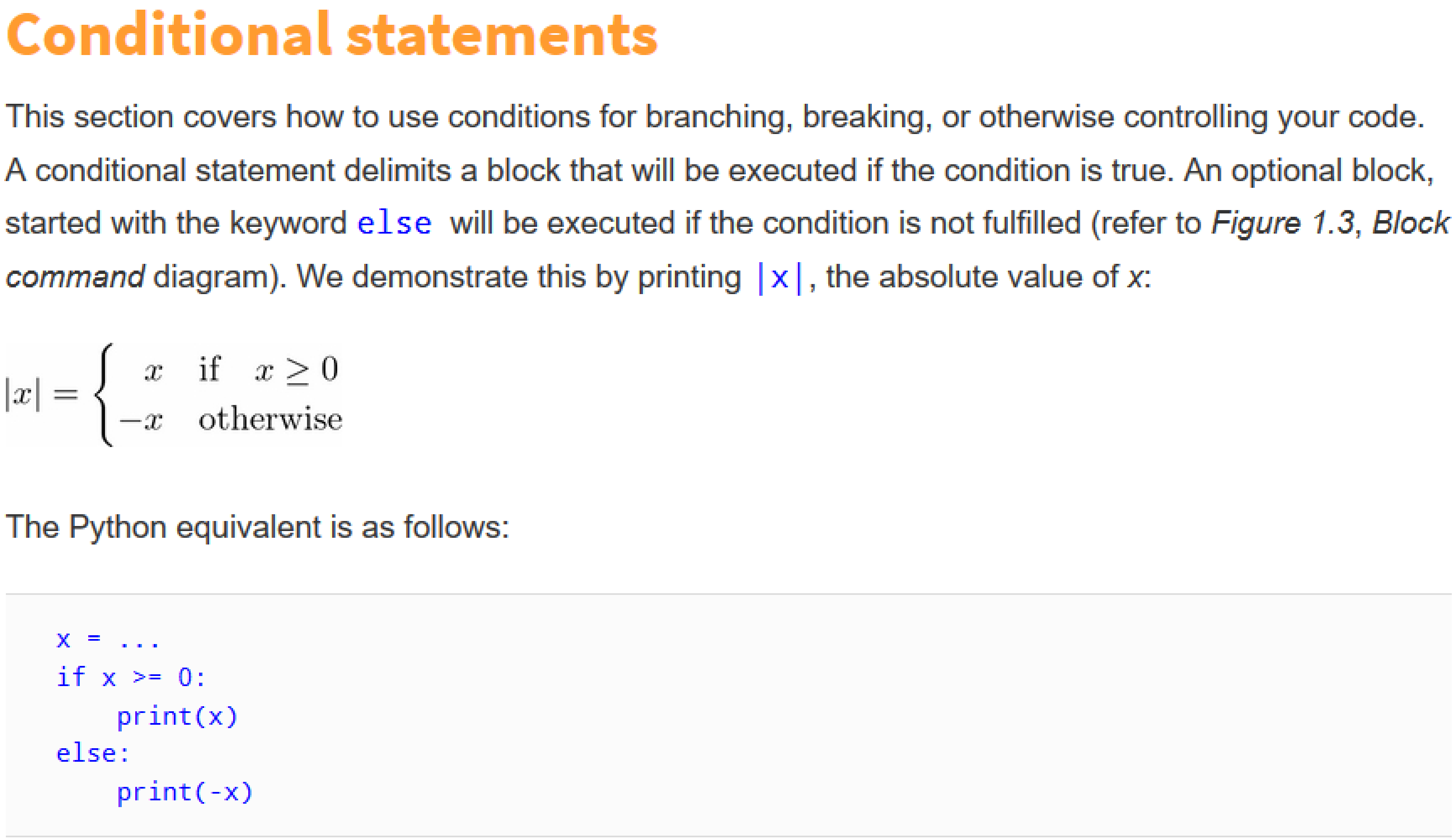
The part to be repeated in the `for` loop has to be properly indented:

```
for elt in my_list:
    do_something
    something_else
print("loop finished") # outside the for block
```



```
n = 30
for iteration in range(n):
    do_something # this gets executed n times
```

Condition



Conditional statements

This section covers how to use conditions for branching, breaking, or otherwise controlling your code. A conditional statement delimits a block that will be executed if the condition is true. An optional block, started with the keyword `else` will be executed if the condition is not fulfilled (refer to *Figure 1.3*, *Block command* diagram). We demonstrate this by printing `|x|`, the absolute value of *x*:

$$|x| = \begin{cases} x & \text{if} \quad x \geq 0 \\ -x & \text{otherwise} \end{cases}$$

The Python equivalent is as follows:

```
x = ...
if x >= 0:
    print(x)
else:
    print(-x)
```

Function

Encapsulating code with functions

Functions are useful for gathering similar pieces of code in one place. Consider the following mathematical function:

$$x \mapsto f(x) := 2x + 1$$

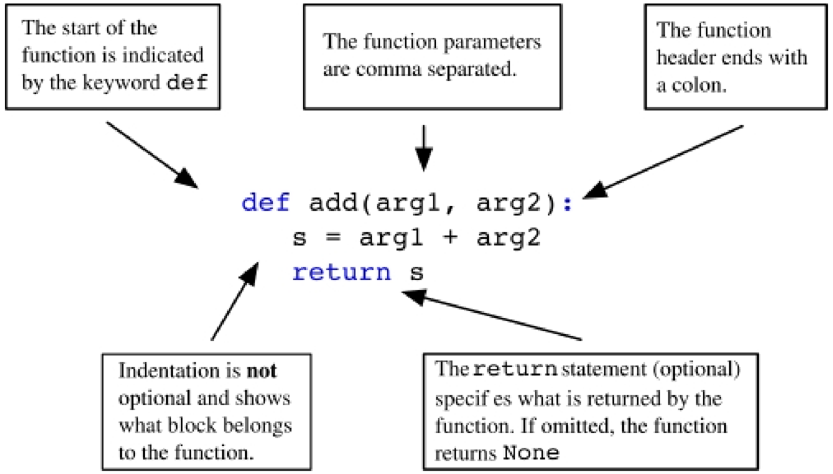
The Python equivalent is as follows:

```
def f(x):
    return 2*x + 1
```

Once the function is defined, it can be called using the following code:

```
f(2) # 5
f(1) # 3
```

- The keyword `def` tells Python we are defining a function.
- `f` is the name of the function.
- `x` is the argument, or input of the function.
- What is after `return` is called the output of the function.



```
def add(arg1, arg2):
    s = arg1 + arg2
    return s
```

The start of the function is indicated by the keyword `def`

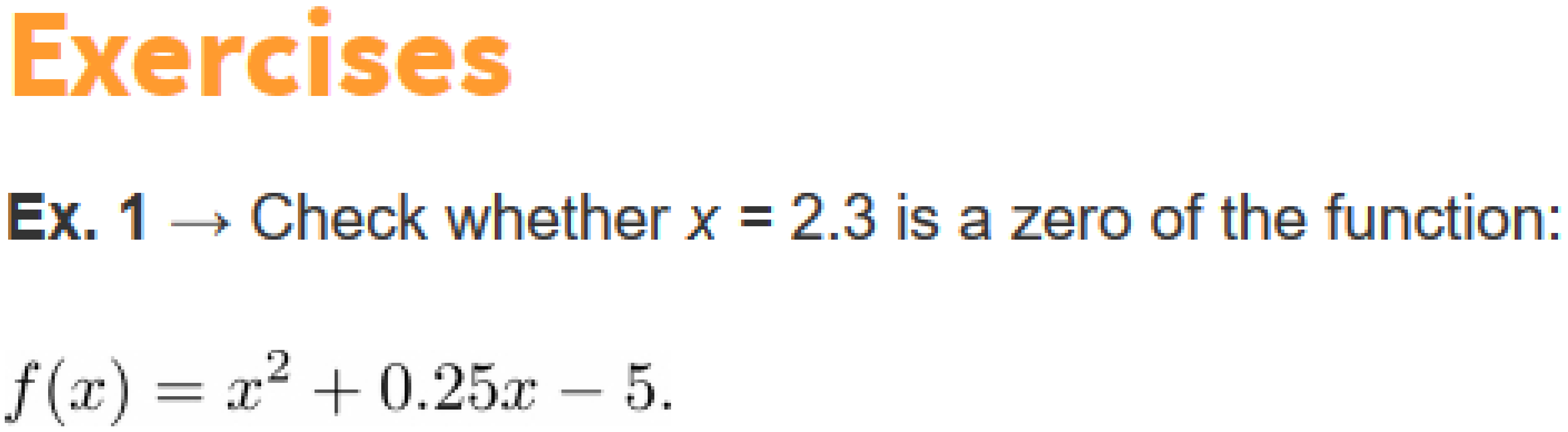
The function parameters are comma separated.

The function header ends with a colon.

Indentation is **not** optional and shows what block belongs to the function.

The `return` statement (optional) specifies what is returned by the function. If omitted, the function returns `None`

Quiz 10



Exercises

Ex. 1 → Check whether *x* = 2.3 is a zero of the function:

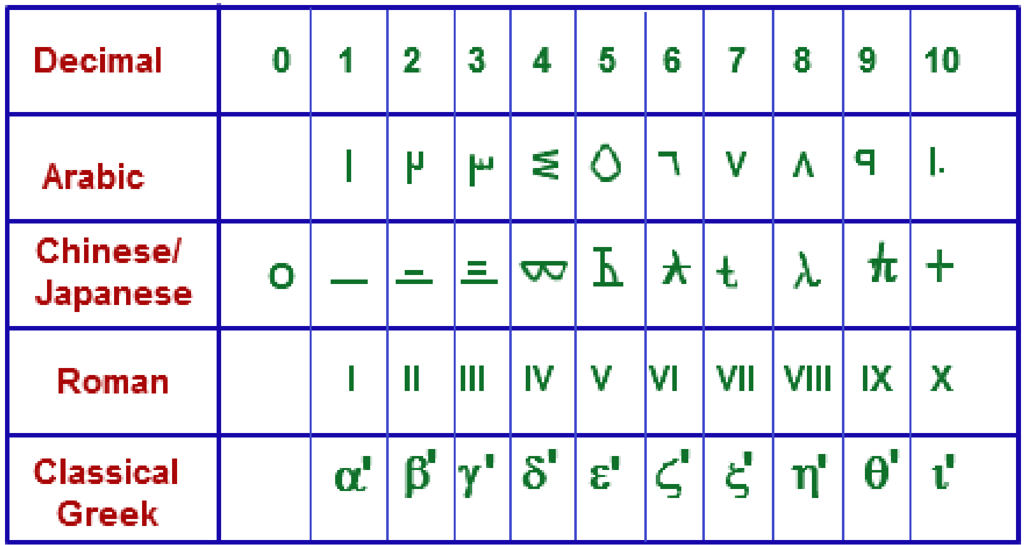
$$f(x) = x^2 + 0.25x - 5.$$

Quiz 10

1) create f(x) function
2) call function f(2.3) at the main program
3) check f(2.3) == 0 ?

Numeral System

ที่มา: https://www.youtube.com/watch?v=ldibzs0jB6I



Decimal	0	1	2	3	4	5	6	7	8	9	10
Arabic		١	٢	٣	٤	٥	٦	٧	٨	٩	١٠
Chinese/ Japanese	〇	一	二	三	四	五	六	七	八	九	十
Roman		I	II	III	IV	V	VI	VII	VIII	IX	X
Classical Greek		α'	β'	γ'	δ'	ε'	ς'	ζ'	η'	θ'	ι'

ที่มา : http://math.tutorvista.com

Decimal Number System

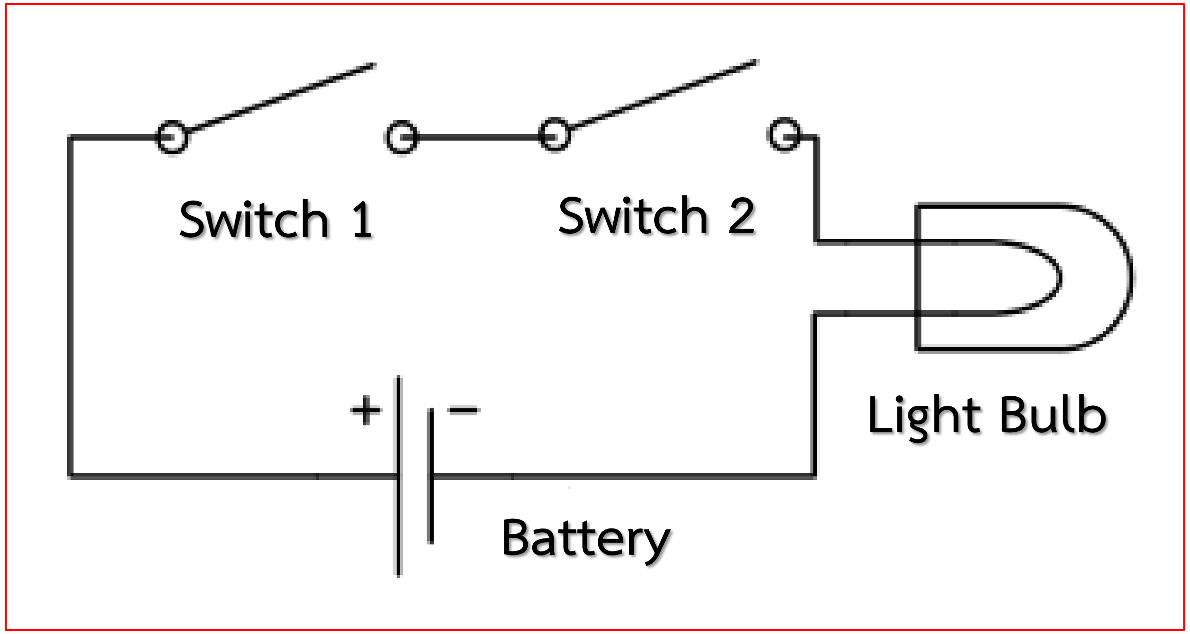
$$D = d_{n-1}d_{n-2}d_{n-3}\ldots d_3d_2d_1d_0.d_{-1}d_{-2}d_{-3}\ldots d_{-m}$$

$$\begin{aligned} D &= \sum_{i\in\mathbb{I}} 10^i d_i \\ &= 10^{n-1}d_{n-1} + 10^{n-2}d_{n-2} + 10^{n-3}d_{n-3}\ldots + 10^2d_2 + 10^1d_1 + 10^0d_0 \\ &\quad + 10^{-1}d_{-1} + 10^{-2}d_{-2} + 10^{-3}d_{-3}\ldots + 10^{-m}d_{-m} \end{aligned}$$

$$d_i \in \{0,1,2,3,4,5,6,7,8,9\}$$

$$\begin{aligned} &435.12_{10} \\ &= 4\times 10^2 + 3\times 10^1 + 5\times 10^0 + 1\times 10^{-1} + 2\times 10^{-}2 \\ &= 4\times 100 + 3\times 10 + 5\times 1 + 1\times \frac{1}{10} + 2\times \frac{1}{100} \\ &= 400 + 30 + 5 + \frac{1}{10} + \frac{2}{100} \end{aligned}$$

Binary & Switch



Switch 1	Switch 2	Dec	Light
0	0	0	0
0	1	1	0
1	0	2	0
1	1	3	1

LSB : Least Significant Bit

MSB : Most Significant Bit

Binary Number System

$$B = b_{n-1}b_{n-2}b_{n-3}\ldots b_3b_2b_1b_0.b_{-1}b_{-2}b_{-3}\ldots b_{-m}$$

$$B = \sum_{i\in\mathbb{I}} 2^i b_i$$

$$= 2^{n-1}b_{n-1} + 2^{n-2}b_{n-2} + 2^{n-3}b_{n-3}\ldots + 2^2b_2 + 2^1b_1 + 2^0b_0$$

$$+ 2^{-1}b_{-1} + 2^{-2}d_{-2} + 2^{-3}b_{-3}\ldots + 2^{-m}b_{-m}$$

$$b_i \in \{0,1\}$$

$$\begin{aligned}
&1101.11_2\\
&\qquad = 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2}\\
&\qquad = 8+4+0+1+\frac{1}{2}+\frac{1}{4}\\
&\qquad = 8+4+0+1+0.5+0.25\\
&\qquad = 13.75_{10}
\end{aligned}$$

Octal Number System

$$O = \varphi_{n-1}\varphi_{n-2}\varphi_{n-3}\cdots\varphi_3\varphi_2\varphi_1\varphi_0.\varphi_{-1}\varphi_{-2}\varphi_{-3}\cdots\varphi_{-m}$$

$$O = \sum_{i\in\mathbb{I}} 8^i\varphi_i$$

$$= 8^{n-1}\varphi_{n-1} + 8^{n-2}\varphi_{n-2} + 8^{n-3}\varphi_{n-3}\ldots + 8^2\varphi_2 + 8^1\varphi_1 + 8^0\varphi_0$$

$$+ 8^{-1}\varphi_{-1} + 8^{-2}\varphi_{-2} + 8^{-3}\varphi_{-3}\ldots + 8^{-m}\varphi_{-m}$$

$$\varphi_i \in \{0,1,2,3,4,5,6,7\}$$

412.24_8

$$= 4\times 8^2 + 1\times 8^1 + 2\times 8^0 + 2\times 8^{-1} + \overset{4}{\cancel{7}}\times 8^{-2}$$

$$= 256 + 8 + 2 + \frac{2}{8} + \frac{4}{64}$$

$$= 266.3125_{10}$$

Hexadecimal Number System

$$H = h_{n-1}h_{n-2}h_{n-3}...h_3h_2h_1h_0.h_{-1}h_{-2}h_{-3}...h_{-m}$$

$$H = \sum_{i\in\mathbb{I}} 16^i h_i$$

$$=16^{n-1}h_{n-1} + 16^{n-2}h_{n-2} + 16^{n-3}h_{n-3}... + 16^2h_2 + 16^1h_1 + 16^0h_0$$

$$+ 16^{-1}h_{-1} + 16^{-2}h_{-2} + 16^{-3}h_{-3}... + 16^{-m}h_{-m}$$

$$h_i \in \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$$

$AB.8_{16}$

$$= 10\times 16^1 + 11\times 16^0 + 8\times 16^{-1}$$

$$= 160 + 11 + \frac{8}{16}$$

$$= 171.5_{10}$$

Number System Conversion

13.25_{10} = ? octal number format

we known **$13.25_{10} = 1101.01_{2}$**

Number System Conversion

13.25_{10} = ? octal number format

we known **$13.25_{10} = 1101.01_2$**

$= 001\ 101\ .\ 010_2$

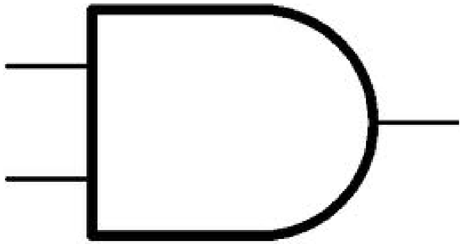
Number System Conversion

13.25_{10} = ? octal number format

we known **$13.25_{10} = 1101.01_2$**

$= 001\ 101\ .\ 010_2$

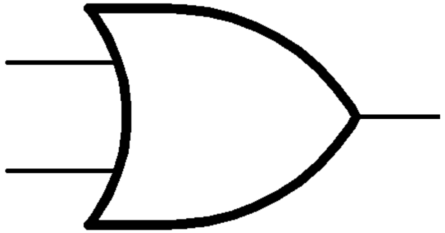
$= 1\ 5\ .\ 2_8$



x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

(a)

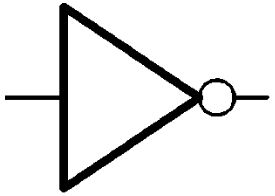
AND



x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

(b)

OR

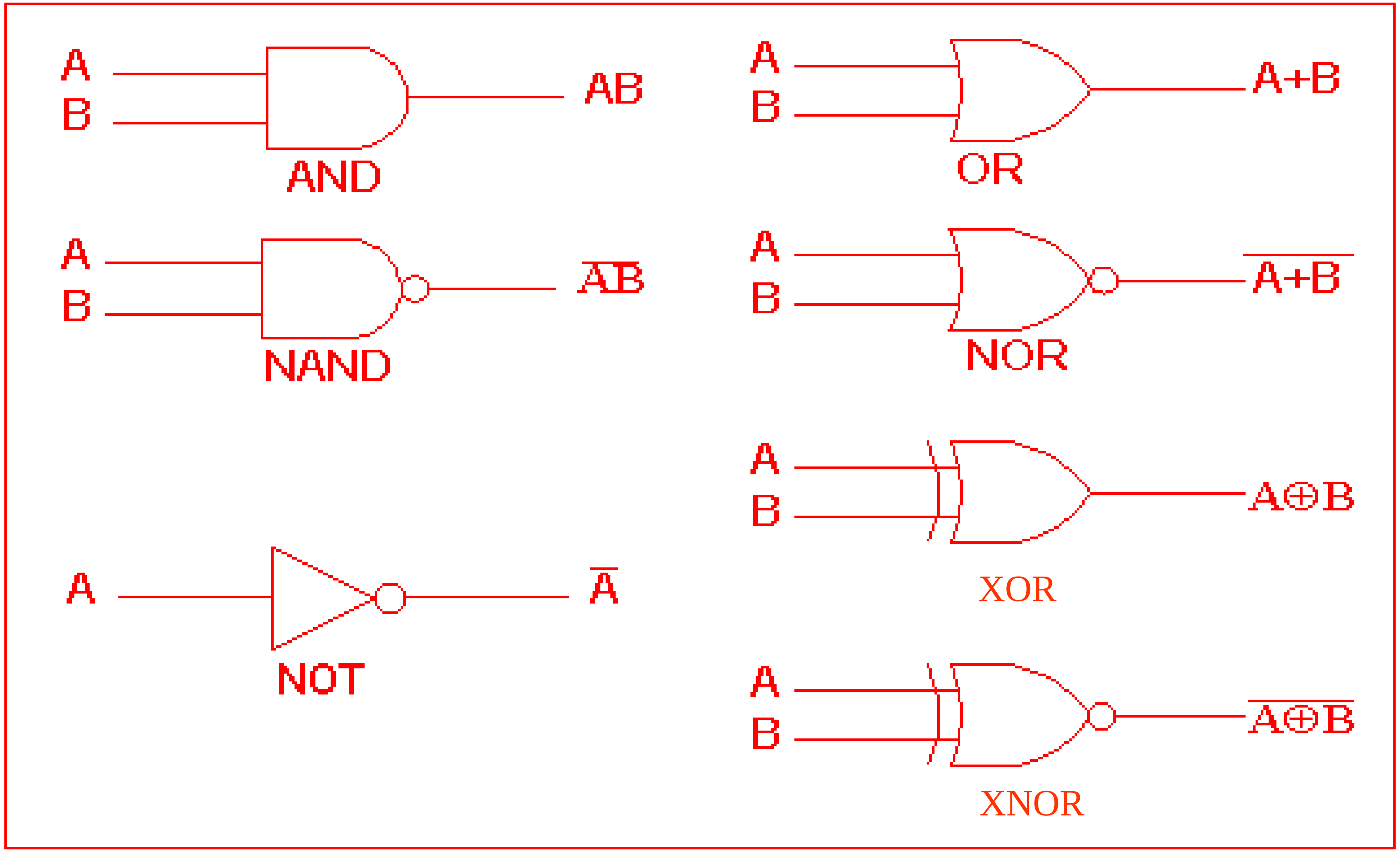


x	F
0	1
1	0

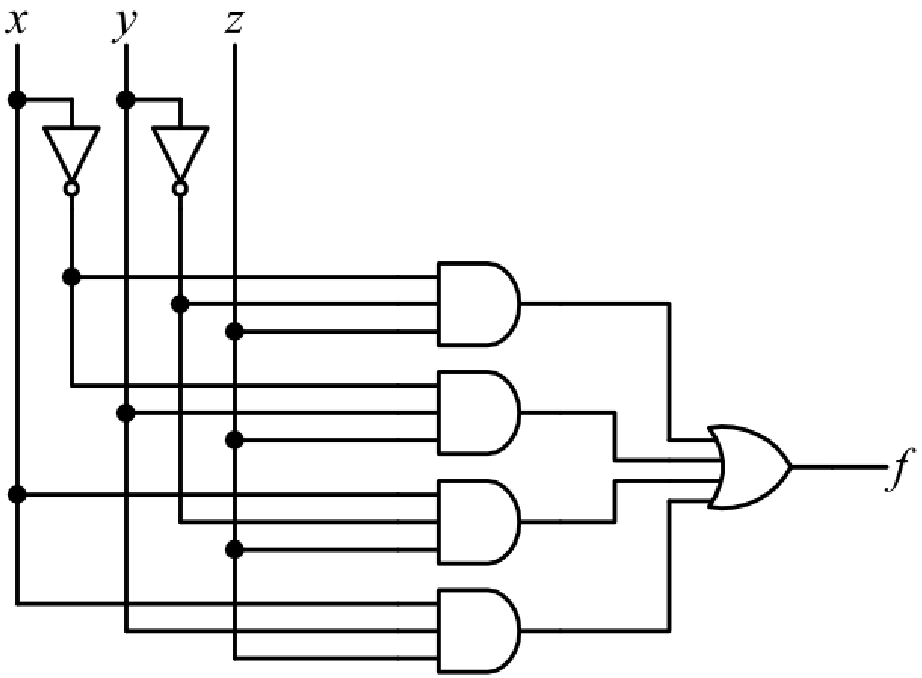
(c)

NOT

Logic Gate Symbols



Using a Truth Table



x	y	z	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Quiz 11

Quiz 9 : Half Adder

In a digital circuit, the number is only “0” or “1”.
Making half adder circuit that can find the result of the summation of two digits using logic gates
(AND, OR, NOT, NOR, NAND, XOR)

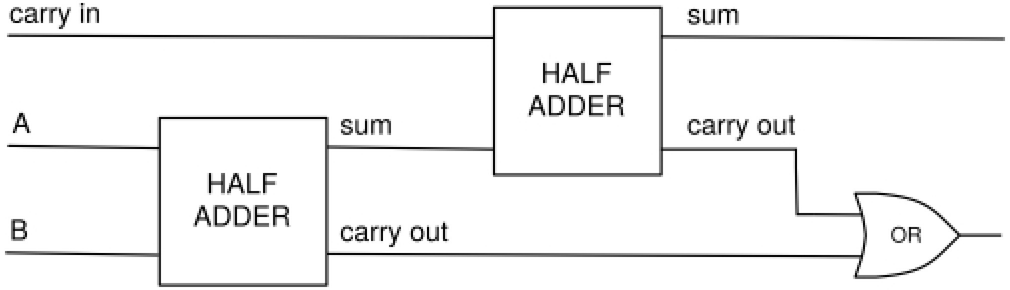
In Python code, create the logic gates as functions.
Run:

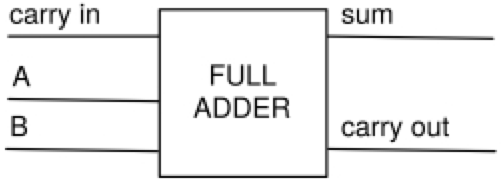
```
        Enter two digits:   0   1
        The result is       :  Sum = 1 , Carry = 0
```

Quiz 12

Quiz 12 : Full Adder

Add some code to create a full adder as see on this diagram.





Homework 02 :

- **Make it the GAME**

“ Take *x* sticks from the pile ”

You play as Player1

Python plays as Player2

There are N sticks in the pile.

Each player takes *x* sticks from the pile

The number of sticks (x) at each time you can take must be <=2

Loop until there are no stick in the pile

The player that take the last stick will lose.

Make the Python player smart or play randomly pick (1 or 2 sticks) from the pile.

Test Case :

- **Example:**

```
How many sticks (N) in the pile: 5
There are 5 sticks in the pile.
What is your name : Somchai

Somchai, how many sticks you will take (1 or 2): 2
There are 3 sticks in the pile.

I, smart computer, takes : 2
There is 1 stick in the pile

Somchai, how many sticks you will take (1 or 2): 1
Somchai, takes the last stick.


I, smart computer, win  !!!!
```

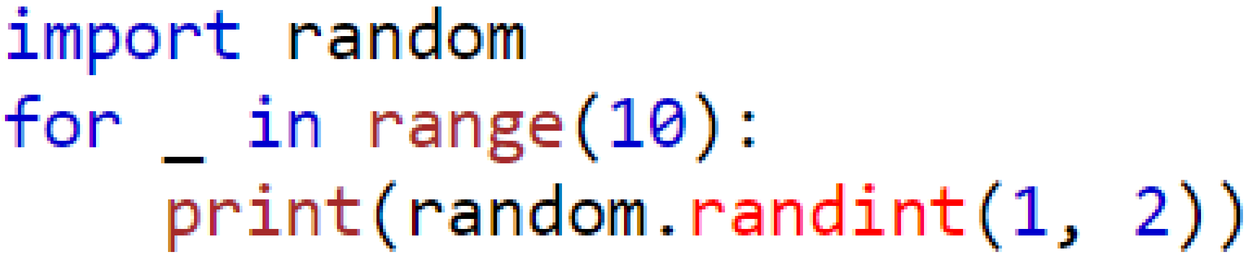
Test Case :

- **Example:**

```
How many sticks (N) in the pile: 5
There are 5 sticks in the pile.
What is your name : Somchai

Somchai, how many sticks you will take (1 or 2): 1
There are 4 sticks in the pile.

I, smart computer,  takes : 2
There are 2 sticks in the pile

Somchai, how many sticks you will take (1 or 2): 1
There is 1 stick in the pile

I, smart computer,  takes the last stick.

Somchai win  ( I, smart computer,  am sad T_T)
```

Hint:

- You may want to use this random number:



```python
import random
for _ in range(10):
    print(random.randint(1, 2))
```