



PYTHON

PROGRAMMING WITH ChatGPT



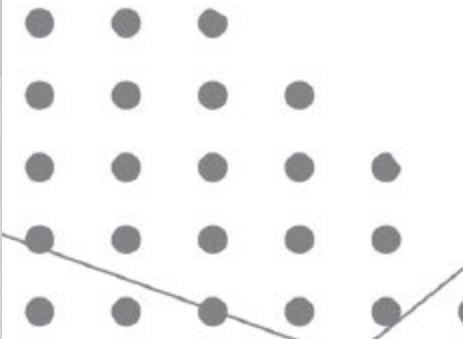
ดำเนินการสอนโดย
ขุน ชินประสาทศักดิ์



KRIN CHINPRASATSAK

DATASCIENTIST
INSTRUCTOR
SPEAKER

Experienced data scientist, instructor and speaker,
AI consultant for the public and private sectors,
CEO & Co-founder of MADEBYAI and QuantMetric



Introduction

Python Programming with ChatGPT

ภาษาพัฒนาโค้ด

**Fundamental
of Python
Programming**

**Working with
ChatGPT**

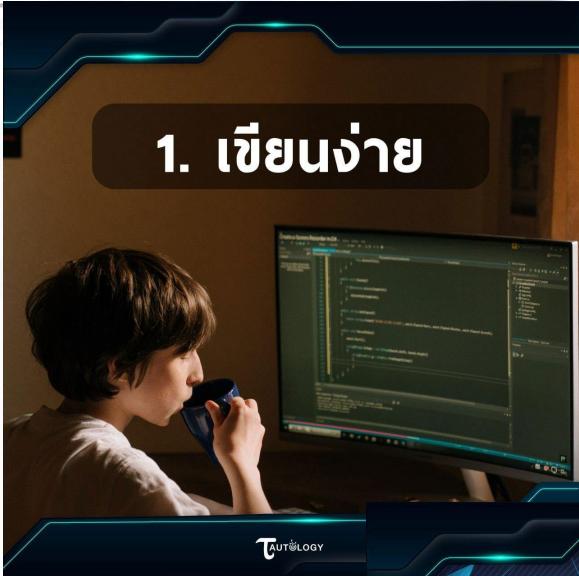
"All Level"



"Beginner to Expert"



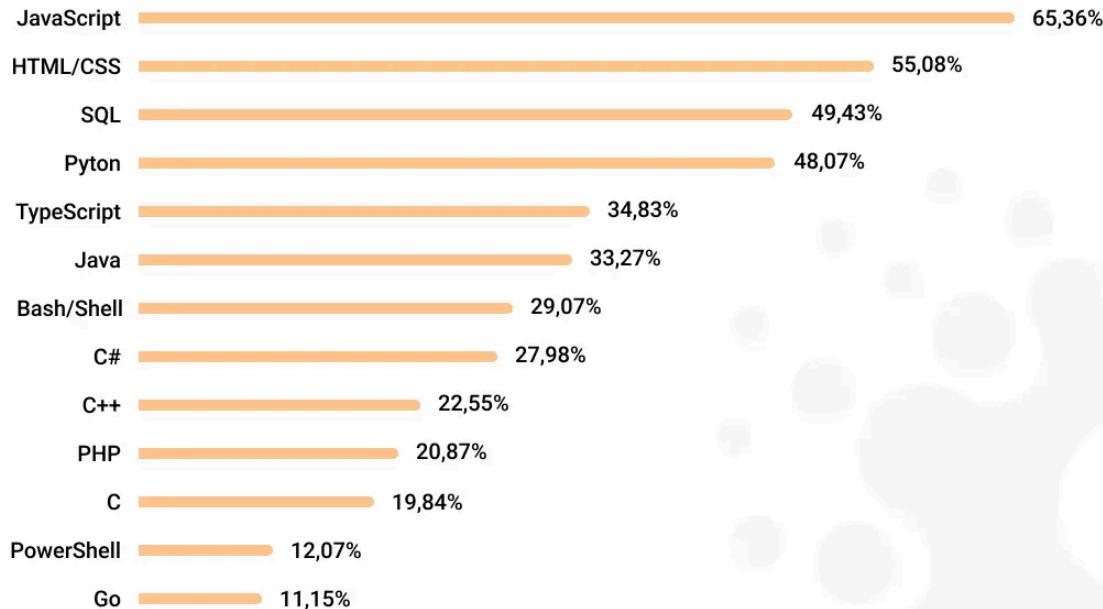
Why Python?



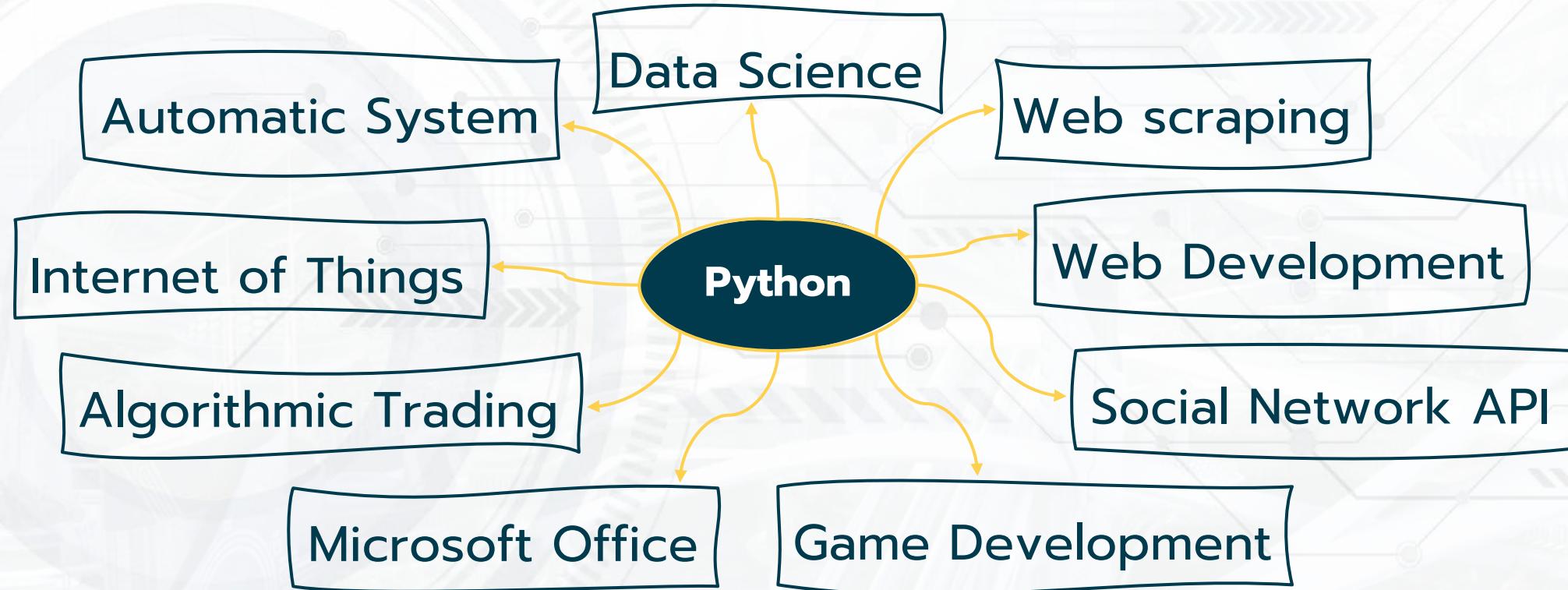
Ref: <https://www.facebook.com/tautologyai/posts/2987242824730932>

Why Python?

The most widely used programming languages worldwide



Ref: <https://www.softmii.com/blog/top-programming-languages-and-frameworks-for-software-development>



Facebook Group



Line Group



วิธีการติดตั้งใช้งาน Python



<https://www.youtube.com/watch?v=lbmyUcGF1i8&list=PL9T5PkG12iF4wd290QFLzs0u-lAnxD4vB&index=3>

วิธีการติดตั้งใช้งาน Python

colab



<https://www.youtube.com/watch?v=ZqPJObbIWyE&list=PL9T5PkG12iF4wd290QFLzs0u-lAnxD4vB&index=4>

เอกสารประกอบการสอน

Print Command

Fundamental of Python Programming

Print Command

1. Print String
2. Print Numeric
3. Print String + Numeric

1. Print String

1.1 Quote

1.2 Double Quote

1.3 Multiple Line

1.1 Quote

```
In [1]: ┌ 1 | print('Hello')
```

```
Hello
```

```
In [2]: ┌ 1 | print('I Love Python')
```

```
I Love Python
```

```
In [3]: ┌ 1 | print('100')
```

```
100
```

1.2 Double Quote

```
In [4]: 1 print("Hello")
```

Hello

```
In [5]: 1 print("I Love Python")
```

I Love Python

```
In [6]: 1 print("100")
```

100

1.3 Multiple Line

In [7]: ►

```
1 print(''  
2 Hello  
3 Hello Python  
4 I Love Python''' )
```

```
Hello  
Hello Python  
I Love Python
```

1.3 Multiple Line

In [8]:

```
1 print("""  
2 Hello  
3 Hello Python  
4 I Love Python""")
```

```
Hello  
Hello Python  
I Love Python
```

Print Command

- ✓ 1. Print String
- 2. Print Numeric
- 3. Print String + Numeric

2. Print Numeric

```
In [9]: print(1)
```

```
1
```

```
In [10]: print(1.5)
```

```
1.5
```

```
In [11]: print(3.50)
```

```
3.5
```

Print Command

- ✓ 1. Print String
- ✓ 2. Print Numeric
- 3. Print String + Numeric

3. Print String + Numeric

3.1 Comma

3.2 String Concatenation

3.3 F-strings

3.1 Comma

```
In [12]: print('I Love You', 3000)
```

I Love You 3000

```
In [13]: print('5 + 7 =', 5 + 7)
```

5 + 7 = 12

3.2 String Concatenation

```
In [14]: print('I Love You ' + str(3000))
```

```
I Love You 3000
```

```
In [15]: print('5 + 7 = ' + str(5 + 7))
```

```
5 + 7 = 12
```

3.3 F-strings

```
In [16]: print(f'I Love You {3000}')
```

```
I Love You 3000
```

```
In [17]: print(f'5 + 7 = {12}')
```

```
5 + 7 = 12
```

Print Command

- ✓ 1. Print String
- ✓ 2. Print Numeric
- ✓ 3. Print String + Numeric

Variable

Fundamental of Python Programming

Variable

1. Create & Assign
2. Create Multiple Variable
3. Identifier

1. Create & Assign

```
In [1]: 1firstname = 'John'
```

```
In [2]: 1lastname = 'Doe'
```

```
In [3]: 1age_full = 'I'm 32 years old.'
```

```
In [4]: 1print('My firstname is', firstname + '.')
2print('My lastname is', lastname + '.')
3print(age_full)
```

My firstname is John.
My lastname is Doe.
I'm 32 years old.

1. Create & Assign

```
In [5]: ┏ 1 a = 5
```

```
In [6]: ┏ 1 b = 7
```

```
In [7]: ┏ 1 print('a =', a)
         2 print('b =', b)
         3 print('a + b =', a + b)
```

```
a = 5
b = 7
a + b = 12
```

Variable

- ✓ 1. Create & Assign
- 2. Create Multiple Variable
- 3. Identifier

2. Create Multiple Variable

```
In [8]: ┏ 1 c = d = e = 5
```

```
In [9]: ┏ 1 print('c =', c)
          2 print('d =', d)
          3 print('e =', e)
```

```
c = 5
d = 5
e = 5
```

2. Create Multiple Variable

```
In [10]: ► 1 f = 6; g = 7; h = 8;
```

```
In [11]: ► 1 print('f =', f)
2 print('g =', g)
3 print('h =', h)
```

```
f = 6
g = 7
h = 8
```

Variable

- ✓ 1. Create & Assign
- ✓ 2. Create Multiple Variable
- 3. Identifier

3. Identifier

3.1 Character & Underscore First

3.2 Case Sensitive

3.3 No Special Character

3.4 Reversed Word

3.1 Character & Underscore First

```
In [12]: ┌ 1 a1 = 5
```

```
In [13]: ┌ 1 print('a1 =', a1)  
          a1 = 5
```

```
In [14]: ┌ 1 _a = 7
```

```
In [15]: ┌ 1 print('_a =', _a)  
          _a = 7
```

3.1 Character & Underscore First

```
In [16]: 1 1a = 5
```

```
File "<ipython-input-16-ce2af4c14e7e>", line 1
  1a = 5
  ^
SyntaxError: invalid syntax
```

```
In [17]: 1 3x = 7
```

```
File "<ipython-input-17-c7219bffa8d3>", line 1
  3x = 7
  ^
SyntaxError: invalid syntax
```

3.2 Case sensitive

```
In [18]: ➜ 1 a = 5  
2 b = 7
```

```
In [19]: ➜ 1 A = 10  
2 B = 14
```

```
In [20]: ➜ 1 print('a =', a)  
2 print('b =', b)  
3 print('A =', A)  
4 print('B =', B)
```

```
a = 5  
b = 7  
A = 10  
B = 14
```

Special Character

”

{ } [] ! @ # \$ % ^
& * () - = / | \ . < >

”

3.3 No Special Character

In [21]: ➜ 1 c\$d = 7

```
File "<ipython-input-21-81e2e827bf95>", line 1
  c$d = 7
  ^
SyntaxError: invalid syntax
```

3.3 No Special Character

```
In [22]: 1 c.d = 8
```

```
-----  
AttributeError                                Traceback (most recent call last)  
<ipython-input-22-56b67979db83> in <module>  
----> 1 c.d = 8
```

```
AttributeError: 'int' object has no attribute 'd'
```

3.3 No Special Character

In [23]: ➜ 1 c d = 9

```
File "<ipython-input-23-757563a21641>", line 1
  c d = 9
  ^
SyntaxError: invalid syntax
```

Reserved Word

and, assert, break, class, continue, def,
del, elif, else, except, exec, finally, for, from,
global, if, import, in, is, lambda, not, or,
pass, print, raise, return, try, while, yield

3.4 Reserved Word

```
In [24]: ┌ 1 | if = 5
          |
          File "<ipython-input-24-1a9bb92cc795>", line 1
            if = 5
            ^
SyntaxError: invalid syntax
```

3.4 Reserved Word

```
In [25]: ┶ 1 for = 10
```

```
File "<ipython-input-25-597287a2d762>", line 1
  for = 10
  ^

```

```
SyntaxError: invalid syntax
```

3.4 Reserved Word

```
In [26]: 1 del = 15
```

```
File "<ipython-input-26-f23a0aa76da8>", line 1
  del = 15
^
```

```
SyntaxError: invalid syntax
```

Variable

- ✓ 1. Create & Assign
- ✓ 2. Create Multiple Variable
- ✓ 3. Identifier

Data Structure

Fundamental of Python Programming

Data Structure

1. Basic Data Structure
2. Composite Data Structure

1. Basic Data Structure

1.1 Numeric

1.2 String

1.1 Numeric

1.1.1 Integer

1.1.2 Floating-Point Number

1.1.3 Boolean

1.1.4 Complex Number

1.1.1 Integer

```
In [1]: ► 1 a = 1
```

```
In [2]: ► 1 b = 2
```

```
In [3]: ► 1 type(a)
```

Out[3]: int

```
In [4]: ► 1 type(b)
```

Out[4]: int

1.1.2 Floating-Point Number

```
In [5]: ► 1 c = 0.5
```

```
In [6]: ► 1 d = 3/4
```

```
In [7]: ► 1 type(c)
```

Out[7]: float

```
In [8]: ► 1 type(d)
```

Out[8]: float

1.1.3 Boolean

```
In [9]: 1 t = True
```

```
In [10]: 1 f = False
```

```
In [11]: 1 type(t)
```

Out[11]: bool

```
In [12]: 1 type(f)
```

Out[12]: bool

1.1.4 Complex Number

```
In [13]: 1 z = 1 + 2j
```

```
In [14]: 1 z.real
```

Out[14]: 1.0

```
In [15]: 1 z.imag
```

Out[15]: 2.0

```
In [16]: 1 type(z)
```

Out[16]: complex

1. Basic Data Structure

1.1 Numeric 

1.2 String

1.2 String

1.2.1 Create String

1.2.2 Read String

1.2.3 String + replace

1.2.4 String + len

1.2.5 String + in

1.2.6 String + split

1.2.7 String + Concatenation

1.2.1 Create String

```
In [17]: 1 char1 = 'Hello Wold'
```

```
In [18]: 1 char2 = '0123456789'
```

```
In [19]: 1 type(char1)
```

```
Out[19]: str
```

```
In [20]: 1 type(char2)
```

```
Out[20]: str
```

1.2.2 Read String

```
In [21]: 1 char3 = '0123456789'
```

```
In [22]: 1 char3[0]
```

```
Out[22]: '0'
```

```
In [23]: 1 char3[5]
```

```
Out[23]: '5'
```

```
In [24]: 1 char3[-1]
```

```
Out[24]: '9'
```

```
In [25]: 1 char3[-2]
```

```
Out[25]: '8'
```

1.2.2 Read String

```
In [26]: 1 char3 = '0123456789'
```

```
In [27]: 1 char3[3:]
```

```
Out[27]: '3456789'
```

```
In [28]: 1 char3[:7]
```

```
Out[28]: '0123456'
```

```
In [29]: 1 char3[3:7]
```

```
Out[29]: '3456'
```

1.2.3 String + replace

```
In [30]: 1 char4 = 'abc'
```

```
In [31]: 1 char4.replace('a', 'A')
```

Out[31]: 'Abc'

```
In [32]: 1 char4 = 'abc'
```

```
In [33]: 1 char4.replace('bc', 'DE')
```

Out[33]: 'aDE'

1.2.4 String + len

```
In [34]: ➜ 1 char5 = '12345'
```

```
In [35]: ➜ 1 len(char5)
```

```
Out[35]: 5
```

1.2.5 String + in

```
In [36]: 1 char6 = 'Python'
```

```
In [37]: 1 'P' in char6
```

Out[37]: True

```
In [38]: 1 'p' in char6
```

Out[38]: False

1.2.6 String + split

```
In [39]: 1 char7 = 'one-two-three'
```

```
In [40]: 1 char7.split('-')
```

```
Out[40]: ['one', 'two', 'three']
```

```
In [41]: 1 char8 = 'I love coding'
```

```
In [42]: 1 char8.split(' ')
```

```
Out[42]: ['I', 'love', 'coding']
```

1.2.7 String + Concatenation

```
In [43]: ► 1 char9 = 'ab'  
          2 char10 = 'cd'
```

```
In [44]: ► 1 char9 + char10
```

Out[44]: 'abcd'

1.2.7 String + Concatenation

```
In [45]: 1 char11 = 'A'  
          2 char12 = 'B'  
          3 char13 = 'C'
```

```
In [46]: 1 char11 + char12 + char13
```

Out[46]: 'ABC'

1. Basic Data Structure

1.1 Numeric 

1.2 String 

Data Structure

- ✓ 1. Basic Data Structure
- 2. Composite Data Structure

2. Composite Data Structure

2.1 List

2.2 Tuple

2.3 Dictionary

2.4 Set

List

“CRUD + sort + len + in”

2.1 List

2.1.1 Create List

2.1.2 Read List

2.1.3 Update List

2.1.4 Delete List

2.1.5 List + sort

2.1.6 List + len

2.1.7 List + in

2.1.1 Create List

```
In [47]: 1 list1 = [1, 2, 3, 4, 5]
          2 list2 = ['a', 'b', 'c', 'd']
          3 list3 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [48]: 1 type(list1)
```

Out[48]: list

```
In [49]: 1 type(list2)
```

Out[49]: list

```
In [50]: 1 type(list3)
```

Out[50]: list

2.1.2 Read List

```
In [51]: █ 1 list4 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [52]: █ 1 list4[0]
```

Out[52]: 1

```
In [53]: █ 1 list4[-1]
```

Out[53]: 'c'

```
In [54]: █ 1 list4[-2]
```

Out[54]: 'b'

2.1.2 Read List

```
In [55]: █ 1 list5 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [56]: █ 1 list5[3:]
```

```
Out[56]: ['a', 'b', 'c']
```

```
In [57]: █ 1 list5[:5]
```

```
Out[57]: [1, 2, 3, 'a', 'b']
```

```
In [58]: █ 1 list5[3:5]
```

```
Out[58]: ['a', 'b']
```

2.1.3 Update List : Replace

```
In [59]: 1 list6 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [60]: 1 list6[0] = 0
```

```
In [61]: 1 list6
```

```
Out[61]: [0, 2, 3, 'a', 'b', 'c']
```

```
In [62]: 1 list6[-1] = 'x'
```

```
In [63]: 1 list6
```

```
Out[63]: [0, 2, 3, 'a', 'b', 'x']
```

2.1.3 Update List : append

```
In [64]: 1 list7 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [65]: 1 list7.append('d')
```

```
In [66]: 1 list7
```

```
Out[66]: [1, 2, 3, 'a', 'b', 'c', 'd']
```

```
In [67]: 1 list7.append('e')
```

```
In [68]: 1 list7
```

```
Out[68]: [1, 2, 3, 'a', 'b', 'c', 'd', 'e']
```

2.1.3 Update List : extend

```
In [69]: ► 1 list8 = [1, 2, 3]
          2 list9 = ['a', 'b', 'c']
```

```
In [70]: ► 1 list8.extend(list9)
```

```
In [71]: ► 1 list8
```

```
Out[71]: [1, 2, 3, 'a', 'b', 'c']
```

2.1.3 Update List : insert

```
In [72]: 1 list10 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [73]: 1 list10.insert(1, 'one')
```

```
In [74]: 1 list10
```

```
Out[74]: [1, 'one', 2, 3, 'a', 'b', 'c']
```

2.1.4 Delete List : del

```
In [75]: 1 list11 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [76]: 1 del list11[2]
```

```
In [77]: 1 list11
```

```
Out[77]: [1, 2, 'a', 'b', 'c']
```

2.1.4 Delete List : del

```
In [78]: 1 list11 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [79]: 1 del list11[-2]
```

```
In [80]: 1 list11
```

```
Out[80]: [1, 2, 3, 'a', 'c']
```

2.1.4 Delete List : remove

```
In [81]: 1 list12 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [82]: 1 list12.remove(1)
```

```
In [83]: 1 list12
```

```
Out[83]: [2, 3, 'a', 'b', 'c']
```

```
In [84]: 1 list12.remove('a')
```

```
In [85]: 1 list12
```

```
Out[85]: [2, 3, 'b', 'c']
```

2.1.4 Delete List : clear

```
In [86]: ➜ 1 list13 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [87]: ➜ 1 list13.clear()
```

```
In [88]: ➜ 1 list13
```

```
Out[88]: []
```

2.1.5 List + sort

```
In [89]: ┏━ 1 ┡ list14 = [1, 5, 4, 2, 3]
```

```
In [90]: ┏━ 1 ┡ list14.sort()
```

```
In [91]: ┏━ 1 ┡ list14
```

```
Out[91]: [1, 2, 3, 4, 5]
```

2.1.5 List + sort

```
In [92]: ┏ 1 list14 = [1, 5, 4, 2, 3]
```

```
In [93]: ┏ 1 list14.sort(reverse = True)
```

```
In [94]: ┏ 1 list14
```

```
Out[94]: [5, 4, 3, 2, 1]
```

2.1.5 List + sort

```
In [95]: ┏ 1 list14 = [1, 5, 4, 2, 3]
```

```
In [96]: ┏ 1 sorted_list14 = sorted(list14)
```

```
In [97]: ┏ 1 sorted_list14
```

Out[97]: [1, 2, 3, 4, 5]

2.1.5 List + sort

```
In [98]: 1 list14 = [1, 5, 4, 2, 3]
```

```
In [99]: 1 sorted_list14 = sorted(list14, reverse = True)
```

```
In [100]: 1 sorted_list14
```

```
Out[100]: [5, 4, 3, 2, 1]
```

2.1.7 List + len

```
In [101]: 1 list15 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [102]: 1 len(list15)
```

Out[102]: 6

```
In [103]: 1 list16 = ['a', 'b', 'c', 'd', 'e']
```

```
In [104]: 1 len(list16)
```

Out[104]: 5

2.1.5 List + in

```
In [105]: 1 list17 = [1, 2, 3, 'a', 'b', 'c']
```

```
In [106]: 1 'b' in list17
```

Out[106]: True

```
In [107]: 1 4 in list17
```

Out[107]: False

2. Composite Data Structure

2.1 List 

2.2 Tuple

2.3 Dictionary

2.4 Set

Tuple

“ CR~~XX~~ + len + in ”

2.2 Tuple

2.2.1 Create Tuple

2.2.2 Read Tuple

2.2.3 Tuple + len

2.2.4 Tuple + in

2.2.1 Create Tuple

```
In [108]: 1 tuple1 = (1, 2, 3, 4, 5)
           2 tuple2 = ('a', 'b', 'c', 'd')
           3 tuple3 = (1, 2, 3, 'a', 'b', 'c')
```

```
In [109]: 1 type(tuple1)
```

Out[109]: tuple

```
In [110]: 1 type(tuple2)
```

Out[110]: tuple

```
In [111]: 1 type(tuple3)
```

Out[111]: tuple

2.2.2 Read Tuple

```
In [112]: 1 tuple4 = (1, 2, 3, 'a', 'b', 'c')
```

```
In [113]: 1 tuple4[0]
```

Out[113]: 1

```
In [114]: 1 tuple4[-1]
```

Out[114]: 'c'

```
In [115]: 1 tuple4[-2]
```

Out[115]: 'b'

2.2.2 Read Tuple

```
In [116]: 1 tuple5 = (1, 2, 3, 'a', 'b', 'c')
```

```
In [117]: 1 tuple5[3:]
```

```
Out[117]: ('a', 'b', 'c')
```

```
In [118]: 1 tuple5[:5]
```

```
Out[118]: (1, 2, 3, 'a', 'b')
```

```
In [119]: 1 tuple5[3:5]
```

```
Out[119]: ('a', 'b')
```

2.2.3 Tuple + len

```
In [120]: 1 tuple6 = (1, 2, 3, 'a', 'b', 'c')
```

```
In [121]: 1 len(tuple6)
```

Out[121]: 6

```
In [122]: 1 tuple7 = ('a', 'b', 'c', 'd', 'e')
```

```
In [123]: 1 len(tuple7)
```

Out[123]: 5

2.2.4 Tuple + in

```
In [124]: 1 tuple8 = (1, 2, 3, 'a', 'b', 'c')
```

```
In [125]: 1 'b' in tuple8
```

Out[125]: True

```
In [126]: 1 4 in tuple8
```

Out[126]: False

2. Composite Data Structure

2.1 List 

2.2 Tuple 

2.3 Dictionary

2.4 Set

Dictionary

“CRUD + len + in”

2.3 Dictionary

2.1.1 Create Dictionary

2.1.2 Read Dictionary

2.1.3 Update Dictionary

2.1.4 Delete Dictionary

2.1.5 Dictionary + len

2.1.6 Dictionary + in

2.3.1 Create Dictionary

```
In [127]: ➜ 1 dict1 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}  
2 dict2 = {'fullname' : 'John Doe', 'hobby' : ['coding', 'study']}
```

```
In [128]: ➜ 1 type(dict1)
```

Out[128]: dict

```
In [129]: ➜ 1 type(dict2)
```

Out[129]: dict

2.3.2 Read Dictionary

```
In [130]: 1 dict3 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
```

```
In [131]: 1 dict3['firstname']
```

```
Out[131]: 'John'
```

```
In [132]: 1 dict3['age']
```

```
Out[132]: 32
```

2.3.3 Update Dictionary : Replace

```
In [133]: 1 dict4 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
```

```
In [134]: 1 dict4['firstname'] = 'Mario'
```

```
In [135]: 1 dict4
```

```
Out[135]: {'firstname': 'Mario', 'lastname': 'Doe', 'age': 32}
```

2.3.3 Update Dictionary : Add

```
In [136]: 1 dict5 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}  
In [137]: 1 dict5['blood_group'] = 'O'  
In [138]: 1 dict5  
Out[138]: {'firstname': 'John', 'lastname': 'Doe', 'age': 32, 'blood_group': 'O'}
```

2.3.3 Update Dictionary : update

```
In [139]: 1 dict6 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
```

```
In [140]: 1 dict6.update({'blood_group' : 'O', 'height' : 180})
```

```
In [141]: 1 dict6
```

```
Out[141]: {'firstname': 'John',
            'lastname': 'Doe',
            'age': 32,
            'blood_group': 'O',
            'height': 180}
```

2.3.4 Delete Dictionary : del

```
In [142]: 1 dict7 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
```

```
In [143]: 1 del dict7['lastname']
```

```
In [144]: 1 dict7
```

```
Out[144]: {'firstname': 'John', 'age': 32}
```

2.3.4 Delete Dictionary : clear

```
In [145]: 1 dict8 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
```

```
In [146]: 1 dict8.clear()
```

```
In [147]: 1 dict8
```

```
Out[147]: {}
```

2.3.5 Dictionary + len

```
In [148]: 1 dict9 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
```

```
In [149]: 1 len(dict9)
```

```
Out[149]: 3
```

```
In [150]: 1 len(dict9.keys())
```

```
Out[150]: 3
```

```
In [151]: 1 len(dict9.values())
```

```
Out[151]: 3
```

2.3.6 Dictionary + in

```
In [152]: 1 dict10 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
```

```
In [153]: 1 'firstname' in dict10
```

Out[153]: True

```
In [154]: 1 'firstname' in dict10.keys()
```

Out[154]: True

```
In [155]: 1 'John' in dict10.values()
```

Out[155]: True

2.3.6 Dictionary + in

```
In [156]: 1 dict10 = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
```

```
In [157]: 1 'weight' in dict10
```

Out[157]: False

```
In [158]: 1 'weight' in dict10.keys()
```

Out[158]: False

```
In [159]: 1 70 in dict10.values()
```

Out[159]: False

2. Composite Data Structure

2.1 List 

2.2 Tuple 

2.3 Dictionary 

2.4 Set

Set

“ CRUD + len + in ”

Set

- + union
- + intersection
- + difference
- + Symmetric Difference

2.4 Set

2.4.1 Create Set

2.4.2 Read Set

2.4.3 Update Set

2.4.4 Delete Set

2.4.5 Set + len

2.4.6 Set + in

2.4 Set

2.4.7 Set + Union

2.4.8 Set + Intersection

2.4.9 Set + Difference

2.4.10 Set + Symmetric Difference

2.4.1 Create Set

```
In [160]: 1 set1 = {1, 2, 3, 4, 5}  
          2 set2 = {1, 2, 3, 'a', 'b', 'c'}
```

```
In [161]: 1 type(set1)
```

Out[161]: set

```
In [162]: 1 type(set2)
```

Out[162]: set

2.4.2 Read Set

```
In [163]: ➜ 1 set3 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [164]: ➜ 1 for x in set3:  
          2     print(x)
```

```
d  
b  
c  
e  
a
```

2.4.3 Update Set : add

```
In [165]: 1 set4 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [166]: 1 set4.add(1)
```

```
In [167]: 1 set4
```

```
Out[167]: {1, 'a', 'b', 'c', 'd', 'e'}
```

2.4.3 Update Set : update

```
In [168]: 1 set5 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [169]: 1 set5.update({1, 2})
```

```
In [170]: 1 set5
```

```
Out[170]: {1, 2, 'a', 'b', 'c', 'd', 'e'}
```

2.4.4 Delete Set : remove

```
In [171]: 1 set6 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [172]: 1 set6.remove('a')
```

```
In [173]: 1 set6
```

```
Out[173]: {'b', 'c', 'd', 'e'}
```

```
In [174]: 1 set6.remove('c')
```

```
In [175]: 1 set6
```

```
Out[175]: {'b', 'd', 'e'}
```

2.4.4 Delete Set : clear

```
In [176]: 1 set7 = {'a', 'b', 'c', 'd', 'e'}
```

```
In [177]: 1 set7.clear()
```

```
In [178]: 1 set7
```

```
Out[178]: set()
```

2.4.5 Set + len

```
In [179]: 1 set8 = {1, 2, 3, 4, 5}
```

```
In [180]: 1 len(set8)
```

Out[180]: 5

2.4.6 Set + in

```
In [181]: 1 set9 = {1, 2, 3, 4, 5}
```

```
In [182]: 1 1 in set9
```

Out[182]: True

```
In [183]: 1 'a' in set9
```

Out[183]: False

2.4.7 Set + Union

```
In [184]: ➔ 1 setA = {1, 2, 3, 4, 5}  
          2 setB = {3, 4, 5, 6, 7}
```

```
In [185]: ➔ 1 setA | setB
```

Out[185]: {1, 2, 3, 4, 5, 6, 7}

2.4.8 Set + Intersection

```
In [186]: 1 setA = {1, 2, 3, 4, 5}  
          2 setB = {3, 4, 5, 6, 7}
```

```
In [187]: 1 setA & setB
```

Out[187]: {3, 4, 5}

2.4.9 Set + Difference

```
In [188]: ➔ 1 setA = {1, 2, 3, 4, 5}  
          2 setB = {3, 4, 5, 6, 7}
```

```
In [189]: ➔ 1 setA - setB
```

Out[189]: {1, 2}

```
In [190]: ➔ 1 setB - setA
```

Out[190]: {6, 7}

2.4.10 Set + Symmetric Difference

```
In [191]: ➔ 1 setA = {1, 2, 3, 4, 5}  
2 setB = {3, 4, 5, 6, 7}
```

```
In [192]: ➔ 1 setA ^ setB
```

Out[192]: {1, 2, 6, 7}

2. Composite Data Structure

2.1 List 

2.2 Tuple 

2.3 Dictionary 

2.4 Set 

Data Structure

- ✓ 1. Basic Data Structure
- ✓ 2. Composite Data Structure

Conclusion

	List	Tuple	Dictionary	Set
Create	<code>listA=[]</code> <code>listB=[1, 2, 3]</code>	<code>tupleA=()</code> <code>tupleB=(1, 2, 3)</code>	<code>dictA={}</code> <code>dictB={'key':value}</code>	<code>setA=set()</code> <code>setB={1, 2, 3}</code>
Read	<code>listB[start]</code> <code>listB[start:end]</code>	<code>tupleB[start]</code> <code>tupleB[start:end]</code>	<code>dictB['key']</code>	<code>for b in setB:</code> <code>print(b)</code>

Conclusion

	List	Tuple	Dictionary	Set
Update	Replace ✓ append ✓ insert ✓ extend ✓ add ✗ update ✗	✗	✓ ✗ ✗ ✗ ✗ ✗	✗ ✗ ✗ ✗ ✗ ✓ ✓
Access	Indexing ✓ Slicing ✓ Looping ✓ Iteration ✓ Membership ✗	Indexing ✗ Slicing ✗ Looping ✗ Iteration ✗ Membership ✗	Indexing ✗ Slicing ✗ Looping ✗ Iteration ✗ Membership ✗	Indexing ✗ Slicing ✗ Looping ✗ Iteration ✗ Membership ✗
Iteration	✓	✗	✓	✓

Conclusion

		List	Tuple	Dictionary	Set
Delete	del	✓		✓	✗
	remove	✓	✗	✗	✓
	clear	✓		✓	✓
	len	len(listB)	len(tupleB)	len(dictB)	len(setB)
	in	'a' in listB	'a' in tupleB	'keyA' in dictB	'a' in setB

Conclusion

Set

Union (|)

Intersection (&)

Difference (-)

Symmetric Difference
(^)

Input

Fundamental of Python Programming

Input

1. Input String
2. Input Numeric

1. Input String

```
In [1]: name = input('Please input your name : ')
print(f'your name is {name}.')
```

```
Please input your name : Krin
your name is Krin.
```

Input

- ✓ 1. Input String
- 2. Input Numeric

2. Input Numeric

```
In [2]: age = int(input('Please input your age : '))
print(f'you are {age} years old.')
```

```
Please input your age : 18
you are 18 years old.
```

```
In [3]: height = float(input('Please input your height (meters) : '))
print(f'you are {height} meters tall.')
```

```
Please input your height (meters) : 1.8
you are 1.8 meters tall.
```

Input

- ✓ 1. Input String
- ✓ 2. Input Numeric

Operator

Fundamental of Python Programming

Operator

$$y = x^2 + 2x + 1$$

- $y, x, 1$ คือ ตัวถูกดำเนินการ (Operand)
- $=, +$ คือ ตัวดำเนินการ (Operator)
- $y = x^2 + 2x + 1$ ก็งหมด คือ นิพจน์ (expression)

Operator

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Membership Operators
5. Operator Precedence

1. Arithmetic Operators

1.1 บวก (+)

1.2 ลบ (-)

1.3 คูณ (*)

1.4 หาร (/)

1.5 ยกกำลัง (**)

1.6 เศษจารการหาร (%)

1.7 ผลหารปัดเศษลง (//)

1.1 บวก (+)

```
In [1]: ► 1 | a = 5  
          2 | b = 10  
          3 | c = 7
```

```
In [2]: ► 1 | a + b
```

Out[2]: 15

```
In [3]: ► 1 | print(b + c)
```

17

1.2 au (-)

```
In [4]: 1 a = 5  
2 b = 10  
3 c = 7
```

```
In [5]: 1 a - b
```

Out[5]: -5

```
In [6]: 1 b - c
```

Out[6]: 3

1.3 គុណ (*)

```
In [7]: ► 1 a = 5  
2 b = 10  
3 c = 7
```

```
In [8]: ► 1 a * b
```

Out[8]: 50

```
In [9]: ► 1 b * c
```

Out[9]: 70

1.4 မျှေး (/)

```
In [10]: 1 a = 5  
          2 b = 10  
          3 c = 7
```

```
In [11]: 1 a/b
```

Out[11]: 0.5

```
In [12]: 1 b/c
```

Out[12]: 1.4285714285714286

1.5 ยกกำลัง (**)

```
In [13]: ► 1 | a = 2  
          2 | b = 3  
          3 | c = 4
```

```
In [14]: ► 1 | a**b
```

Out[14]: 8

```
In [15]: ► 1 | b**c
```

Out[15]: 81

1.6 ເສຫາກາරໂຮດ (%)

```
In [16]: 1 a = 5  
          2 b = 10  
          3 c = 7
```

```
In [17]: 1 a%b
```

Out[17]: 5

```
In [18]: 1 b%c
```

Out[18]: 3

1.7 ພັກຫາກັບເຄືອຂອງ (//)

```
In [19]: █ 1 a = 5  
2 b = 10  
3 c = 7
```

```
In [20]: █ 1 a//b
```

Out[20]: 0

```
In [21]: █ 1 b//c
```

Out[21]: 1

Operator

- ✓ 1. Arithmetic Operators
- 2. Comparison Operators
- 3. Logical Operators
- 4. Membership Operators
- 5. Operator Precedence

2. Comparison Operators

2.1 ตรวจสอบความเท่ากัน (==)

2.2 ตรวจสอบความไม่เท่ากัน (!=)

2.3 ตรวจสอบความมากกว่า (>)

2.4 ตรวจสอบความน้อยกว่า (<)

2.5 ตรวจสอบความมากกว่าหรือเท่ากับ (>=)

2.6 ตรวจสอบความน้อยกว่าหรือเท่ากับ (<=)

2.1 ตรวจสอบความเท่ากัน (==)

```
In [22]: 1 a = 5  
          2 b = 'name'
```

```
In [23]: 1 a == 5
```

Out[23]: True

```
In [24]: 1 a == 7
```

Out[24]: False

```
In [25]: 1 b == 'name'
```

Out[25]: True

```
In [26]: 1 b == 'fullname'
```

Out[26]: False

2.2 ตรวจสอบความไม่เท่ากัน (\neq)

```
In [27]: ┌─┐ 1 a = 5  
      2 b = 'name'
```

```
In [28]: ┌─┐ 1 a != 5
```

Out[28]: False

```
In [29]: ┌─┐ 1 a != 7
```

Out[29]: True

```
In [30]: ┌─┐ 1 b != 'name'
```

Out[30]: False

```
In [31]: ┌─┐ 1 b != 'fullname'
```

Out[31]: True

2.3 តារាងសែបគារមាត្រក់ (>)

In [32]: `1 a = 5`

In [33]: `1 a > 4`

Out[33]: `True`

In [34]: `1 a > 5`

Out[34]: `False`

In [35]: `1 a > 6`

Out[35]: `False`

2.4 ตรวจสอบความน้อยกว่า (<)

```
In [36]: 1 a = 5
```

```
In [37]: 1 a < 4
```

Out[37]: False

```
In [38]: 1 a < 5
```

Out[38]: False

```
In [39]: 1 a < 6
```

Out[39]: True

2.5 ตรวจสอบความมากกว่าหรือเท่ากับ (\geq)

```
In [40]: 1 a = 5
```

```
In [41]: 1 a >= 4
```

Out[41]: True

```
In [42]: 1 a >= 5
```

Out[42]: True

```
In [43]: 1 a >= 6
```

Out[43]: False

2.6 ตรวจสอบความน้อยกว่าหรือเท่ากับ (\leq)

In [44]:  1 | a = 5

In [45]:  1 | a \leq 4

Out[45]: False

In [46]:  1 | a \leq 5

Out[46]: True

In [47]:  1 | a \leq 6

Out[47]: True

Operator

- ✓ 1. Arithmetic Operators
- ✓ 2. Comparison Operators
- 3. Logical Operators
- 4. Membership Operators
- 5. Operator Precedence

3. Logical Operators

3.1 and

3.2 or

3.3 not

3.1 and

p	q	p and q
True	True	True
True	False	False
False	True	False
False	False	False

3.1 and

```
In [48]: ➤ 1 t = True  
2 f = False
```

```
In [49]: ➤ 1 t and f
```

Out[49]: False

3.2 or

p	q	p or q
True	True	True
True	False	True
False	True	True
False	False	False

3.2 or

```
In [50]: 1 t = True  
          2 f = False
```

```
In [51]: 1 t or f
```

Out[51]: True

3.3 not

p	not p
True	False
False	True

3.3 not

```
In [52]: ➜ 1 t = True  
          2 f = False
```

```
In [53]: ➜ 1 not t
```

Out[53]: False

```
In [54]: ➜ 1 not f
```

Out[54]: True

Operator

- ✓ 1. Arithmetic Operators
- ✓ 2. Comparison Operators
- ✓ 3. Logical Operators
- 4. Membership Operators
- 5. Operator Precedence

4. Membership Operators

4.1 in

4.2 not in

4.1 in

```
In [55]: ➜ 1 listA = [1, 2, 3, 4]
          2 tupleA = ('a', 'b', 'c', 'd')
```

```
In [56]: ➜ 1 1 in listA
```

Out[56]: True

```
In [57]: ➜ 1 1 in tupleA
```

Out[57]: False

4.1 in

```
In [58]: ➜ 1 dictA = {'name' : 'John', 'age' : 32}  
2 setA = {'Sushi', 'Salmon'}
```

```
In [59]: ➜ 1 'name' in dictA
```

Out[59]: True

```
In [60]: ➜ 1 'Egg' in setA
```

Out[60]: False

4.2 not in

```
In [61]: ➜ 1 listA = [1, 2, 3, 4]
          2 tupleA = ('a', 'b', 'c', 'd')
```

```
In [62]: ➜ 1 1 not in listA
```

Out[62]: False

```
In [63]: ➜ 1 1 not in tupleA
```

Out[63]: True

4.2 not in

```
In [64]: 1 dictA = {'name' : 'John', 'age' : 32}  
         2 setA = {'Sushi', 'Salmon'}
```

```
In [65]: 1 'name' not in dictA
```

Out[65]: False

```
In [66]: 1 'Egg' not in setA
```

Out[66]: True

Operator

- ✓ 1. Arithmetic Operators
- ✓ 2. Comparison Operators
- ✓ 3. Logical Operators
- ✓ 4. Membership Operators
- 5. Operator Precedence

5. Operators Precedence

()

*** . / . % . //**

+ . -

<= . < . > . >=

== . !=

=

Operator

- ✓ 1. Arithmetic Operators
- ✓ 2. Comparison Operators
- ✓ 3. Logical Operators
- ✓ 4. Membership Operators
- ✓ 5. Operator Precedence

Flowchart

Fundamental of Python Programming

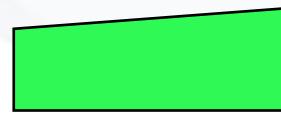
Flowchart



เริ่ม, จบ



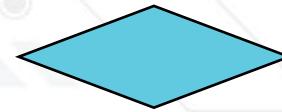
การประกาศตัวแปร
การคำนวณ



รับอินพุต



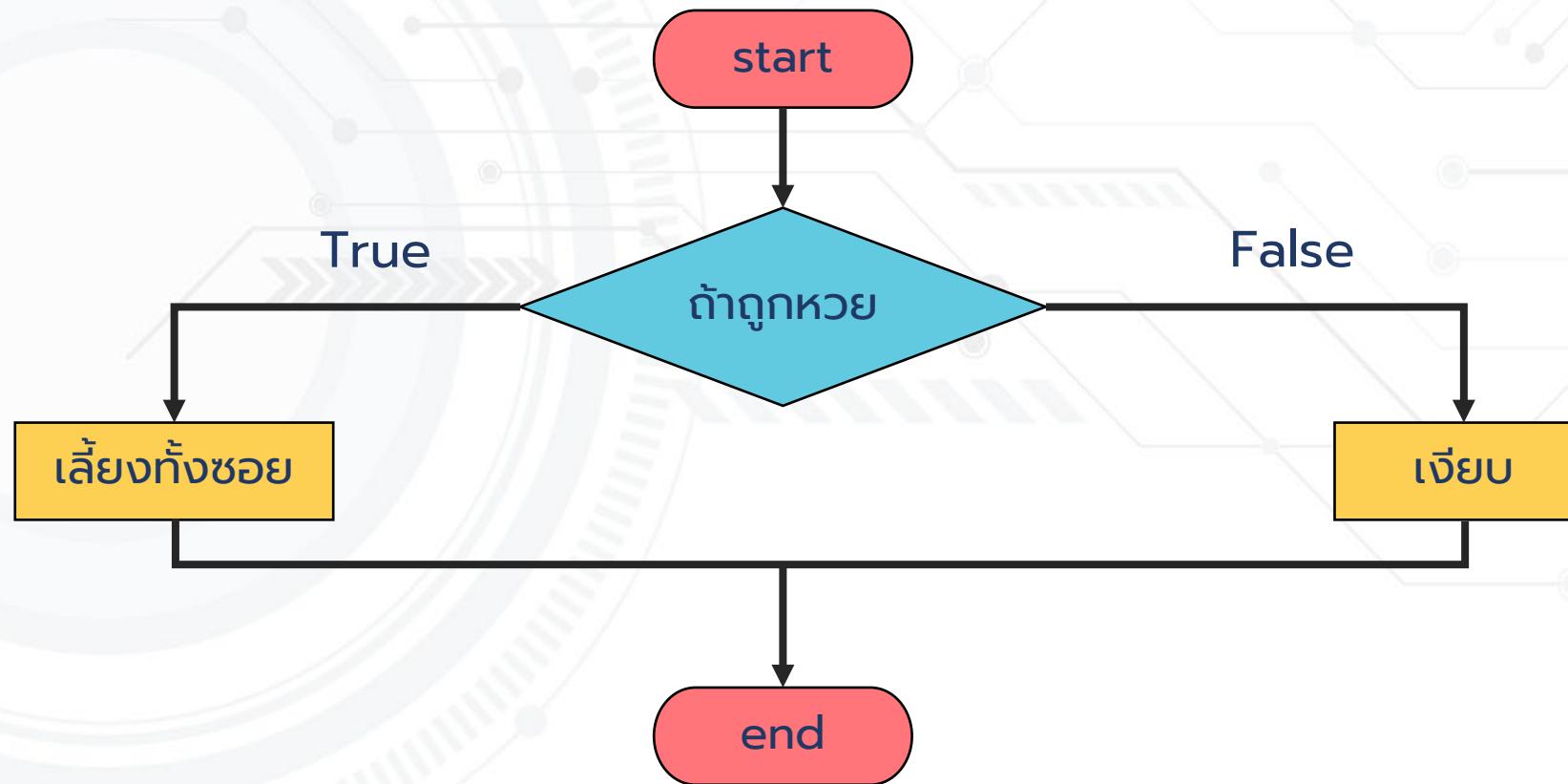
พิมพ



เงื่อนไข
(if-else)

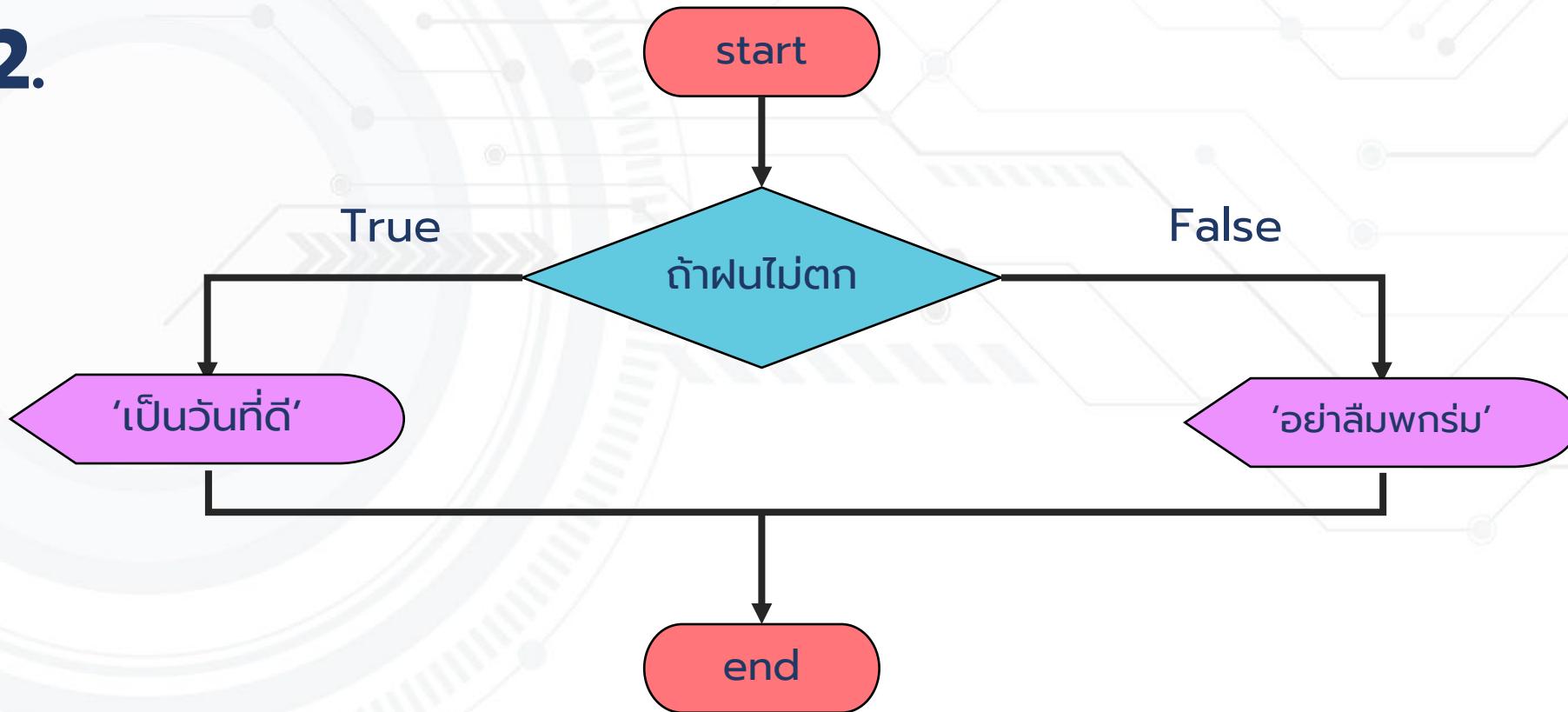
Flowchart

Ex 1.



Flowchart

Ex 2.



If-Else

Fundamental of Python Programming

If-Else

1. If
2. If-Else
3. If-Elif
4. If-Elif-Else

1. If

In [1]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'John':
4     print('Hi John')
5 print('Nice to meet you')
```

Please insert your name: John

Hi John
Nice to meet you

1. If

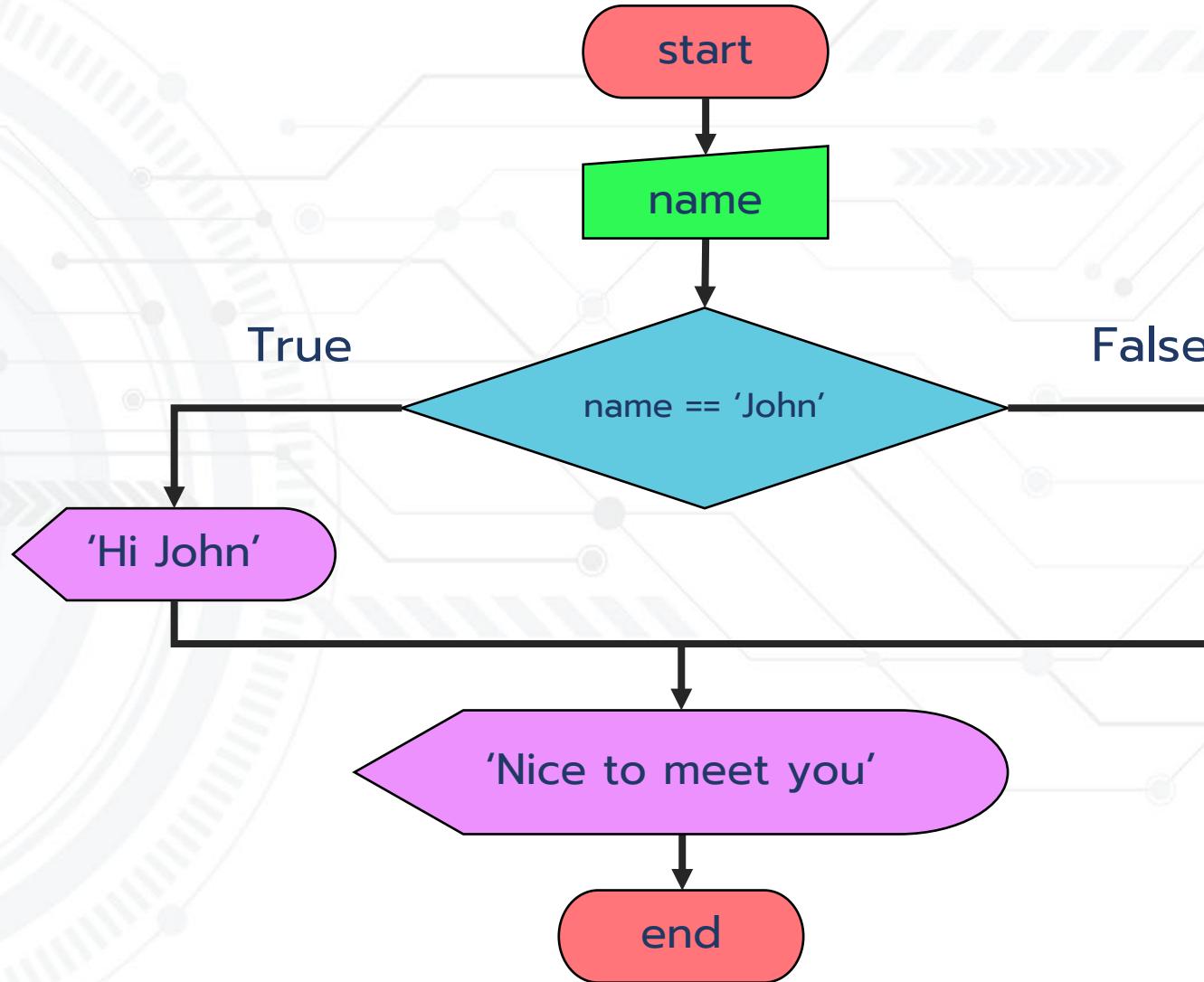
In [2]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'John':
4     print('Hi John')
5 print('Nice to meet you')
```

Please insert your name: Jane

Nice to meet you

1. If



If-Else

- ✓ 1. If
- 2. If-Else
- 3. If-Elif
- 4. If-Elif-Else

2. If-Else

In [3]: ►

```
1 score = int(input('Please insert score: '))
2 if score >= 50:
3     print('Pass')
4 else:
5     print('Fail')
```

```
Please insert score: 50
Pass
```

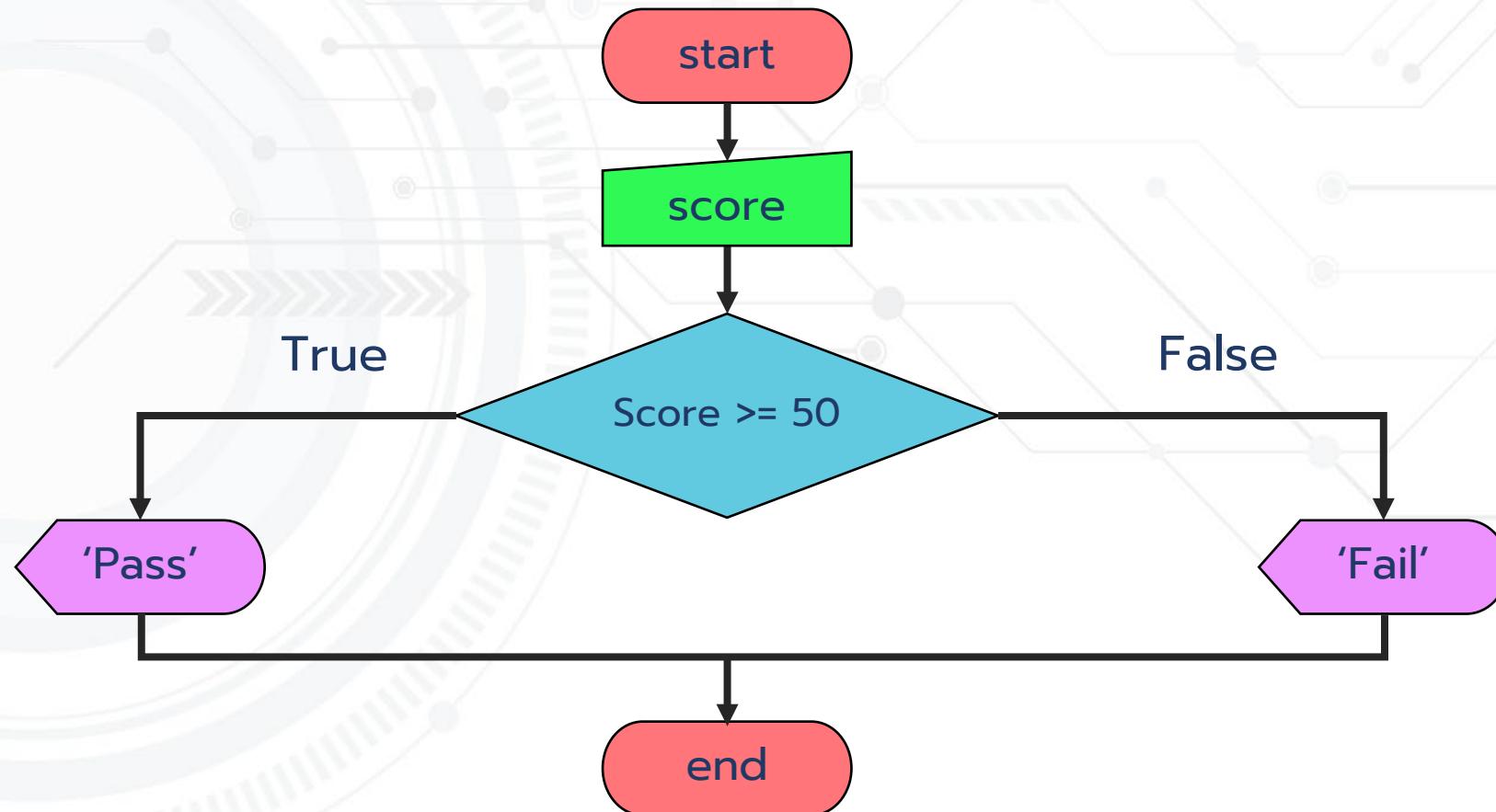
2. If-Else

In [4]: ►

```
1 score = int(input('Please insert score: '))
2 if score >= 50:
3     print('Pass')
4 else:
5     print('Fail')
```

Please insert score: 49
Fail

2. If-Else



If-Else

- ✓ 1. If
- ✓ 2. If-Else
- 3. If-Elif
- 4. If-Elif-Else

3. If-Elif

In [5]: ►

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Alan Turing':
4     print('Oh God !')
5 elif name == 'Isac Newton':
6     print('OMG !')
7 print('Glad to see you')
```

Please insert your name: Alan Turing

Oh God !

Glad to see you

3. If-Elif

In [6]: ►

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Alan Turing':
4     print('Oh God !')
5 elif name == 'Isac Newton':
6     print('OMG !')
7 print('Glad to see you')
```

Please insert your name: Isac Newton

OMG !

Glad to see you

3. If-Elif

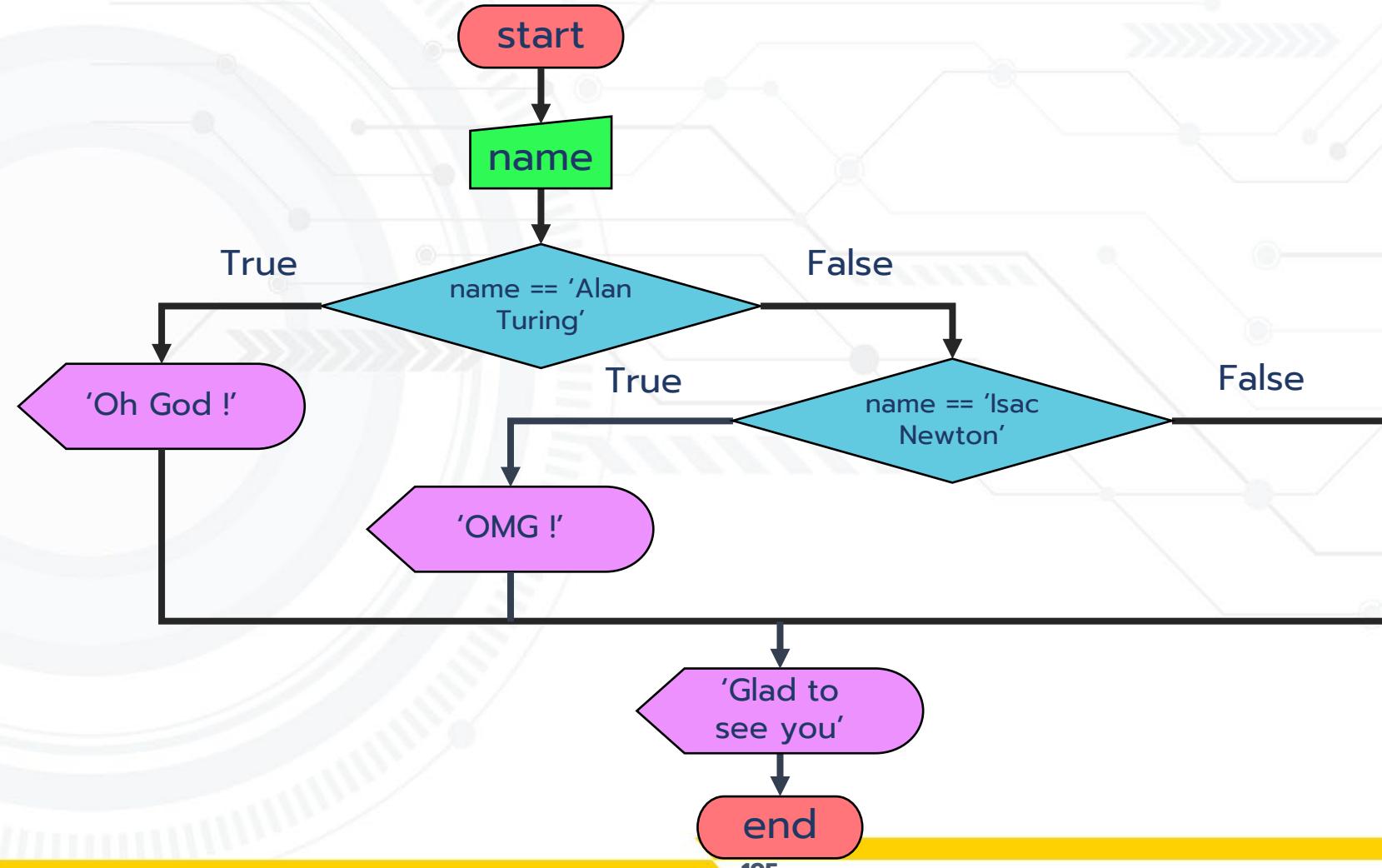
In [7]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Alan Turing':
4     print('Oh God !')
5 elif name == 'Isac Newton':
6     print('OMG !')
7 print('Glad to see you')
```

Please insert your name: Yoda

Glad to see you

3. If-Elif



If-Else

- ✓ 1. If
- ✓ 2. If-Else
- ✓ 3. If-Elif
- 4. If-Elif-Else

4. If-Elif-Else

In [8]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Gift

I miss you

4. If-Elif-Else

In [9]:

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Cherry

I love you

4. If-Elif-Else

In [10]: ►

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Wine

I need you

4. If-Elif-Else

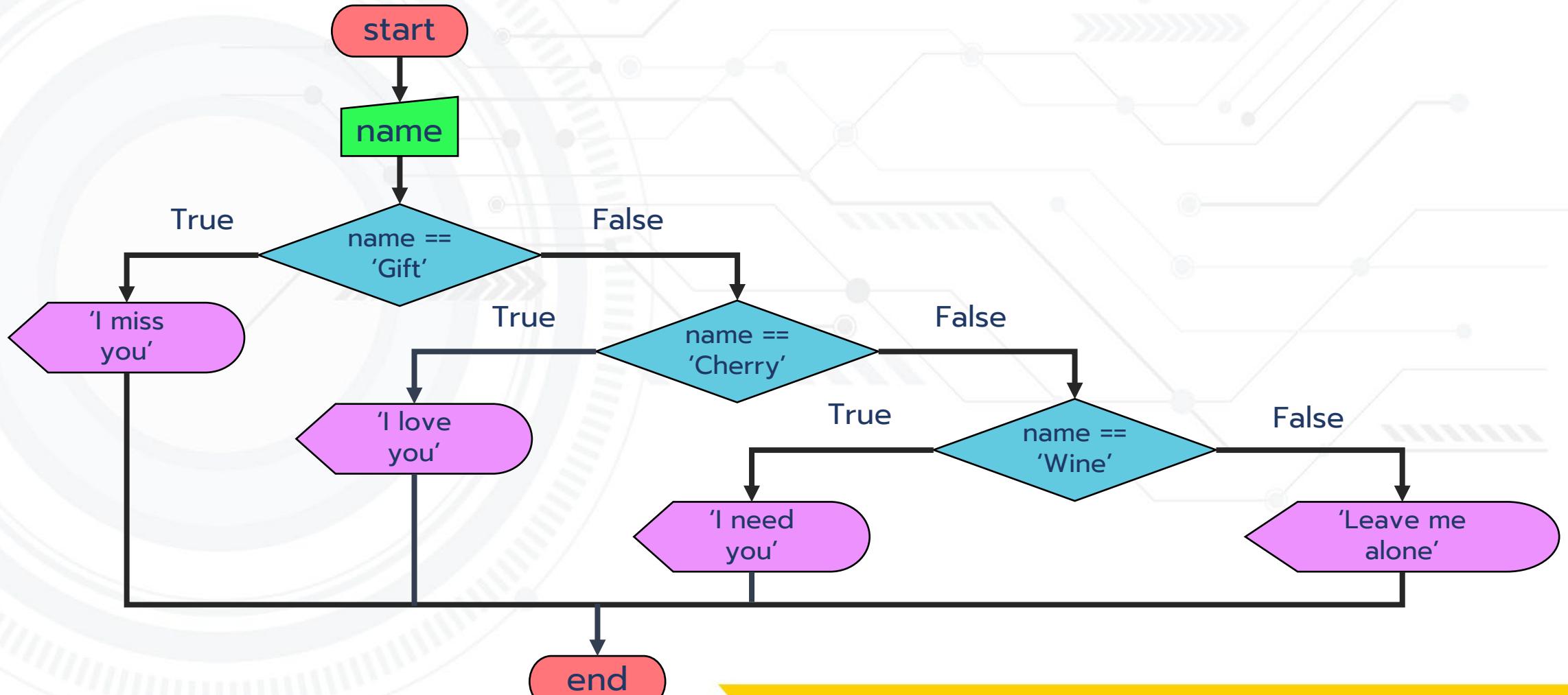
In [11]: ►

```
1 name = input('Please insert your name: ')
2 print()
3 if name == 'Gift':
4     print('I miss you')
5 elif name == 'Cherry':
6     print('I love you')
7 elif name == 'Wine':
8     print('I need you')
9 else:
10    print('Leave me alone')
```

Please insert your name: Longan

Leave me alone

4. If-Elif-Else



If-Else

- ✓ 1. If
- ✓ 2. If-Else
- ✓ 3. If-Elif
- ✓ 4. If-Elif-Else

For & While Loop

Fundamental of Python Programming

For & While Loop

1. For
2. While
3. Loop & Statement

1. For

1.1 Standard For

1.2 For + String

1.3 For + List

1.4 For + Tuple

1.5 For + Dictionary

1.6 For + Set

1.1 Standard For

1.1.1 for i in range (end)

1.1.2 for i in range (start, end)

1.1.3 for i in range (start, end, step)

1.1.1 for i in range (end)

```
In [1]: ┏ 1 for i in range(5):  
  └ 2   print(i)
```

```
0  
1  
2  
3  
4
```

1.1.2 for i in range (start, end)

```
In [2]: ➜ 1 for i in range(5, 11):  
 2     print(i)
```

```
5  
6  
7  
8  
9  
10
```

1.1.3 for i in range (start, end, step)

```
In [3]: 1 for i in range(10, 20, 2):  
2     print(i)
```

```
10  
12  
14  
16  
18
```

1. For

1.1 Standard For 

1.2 For + String

1.3 For + List

1.4 For + Tuple

1.5 For + Dictionary

1.6 For + Set

1.2 For + String

In [4]: ►

```
1 char = 'Python'  
2 n = len(char)  
3 for i in range(n):  
4     print(char[i])
```

P
y
t
h
o
n

1.2 For + String

In [5]: ➔

```
1 char = 'Python'  
2 for i in range(len(char)):  
3     print(char[i])
```

P
y
t
h
o
n

1.2 For + String

In [6]: ➔

```
1 char = 'Python'  
2 for c in char:  
3     print(c)
```

P
y
t
h
o
n

1.2 For + String

In [7]:

```
1 char = 'Python'  
2 for i,c in enumerate(char):  
3     print(i, c)
```

```
0 P  
1 y  
2 t  
3 h  
4 o  
5 n
```

1. For

1.1 Standard For 

1.2 For + String 

1.3 For + List

1.4 For + Tuple

1.5 For + Dictionary

1.6 For + Set

1.3 For + List

In [8]: ➔

```
1 listx = ['a', 'b', 'c', 'd', 'e']
2 n = len(listx)
3 for i in range(n):
4     print(listx[i])
```

a
b
c
d
e

1.3 For + List

In [9]: ➔

```
1 listx = ['a', 'b', 'c', 'd', 'e']
2 for i in range(len(listx)):
3     print(listx[i])
```

a
b
c
d
e

1.3 For + List

```
In [10]: ┏━
 1 listx = ['a', 'b', 'c', 'd', 'e']
 2 for x in listx:
 3     print(x)
```

```
a
b
c
d
e
```

1.3 For + List

```
In [11]: ➜ 1 listx = ['a', 'b', 'c', 'd', 'e']  
2 for i, x in enumerate(listx):  
3     print(i, x)
```

```
0 a  
1 b  
2 c  
3 d  
4 e
```

1. For

1.1 Standard For 

1.2 For + String 

1.3 For + List 

1.4 For + Tuple

1.5 For + Dictionary

1.6 For + Set

1.4 For + Tuple

```
In [12]: ► 1 tuplex = ('a', 'b', 'c', 'd', 'e')
  2 n = len(tuplex)
  3 for i in range(n):
  4     print(tuplex[i])
```

```
a
b
c
d
e
```

1.4 For + Tuple

In [13]: ➔

```
1 tuplex = ('a', 'b', 'c', 'd', 'e')
2 for i in range(len(tuplex)):
3     print(tuplex[i])
```

a
b
c
d
e

1.4 For + Tuple

```
In [14]: ► 1 tuplex = ('a', 'b', 'c', 'd', 'e')
          2 for x in tuplex:
          3     print(x)
```

```
a
b
c
d
e
```

1.4 For + Tuple

```
In [15]: ➜ 1 tuplex = ('a', 'b', 'c', 'd', 'e')
          2 for i, x in enumerate(tuplex):
          3     print(i, x)
```

```
0 a
1 b
2 c
3 d
4 e
```

1. For

1.1 Standard For 

1.2 For + String 

1.3 For + List 

1.4 For + Tuple 

1.5 For + Dictionary

1.6 For + Set

1.5 For + Dictionary

```
In [16]: ➜ 1 dictx = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
2 for key in dictx:
3     print(key)
```

```
firstname
lastname
age
```

1.5 For + Dictionary

```
In [17]: ► 1 dictx = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
      2 for key in dictx.keys():
      3     print(key)

firstname
lastname
age
```

1.5 For + Dictionary

```
In [18]: ➜ 1 dictx = {'firstname' : 'John', 'lastname' : 'Doe', 'age' : 32}
2 for value in dictx.values():
3     print(value)
```

```
John
Doe
32
```

1. For

- 1.1 Standard For 
- 1.2 For + String 
- 1.3 For + List 
- 1.4 For + Tuple 
- 1.5 For + Dictionary 
- 1.6 For + Set

1.6 For + Set

```
In [19]: ► 1 setx = {1, 2, 3, 'a', 'b', 'c'}  
2 for x in setx:  
3     print(x)
```

```
b  
1  
2  
3  
a  
c
```

1. For

- 1.1 Standard For 
- 1.2 For + String 
- 1.3 For + List 
- 1.4 For + Tuple 
- 1.5 For + Dictionary 
- 1.6 For + Set 

For & While Loop

- ✓ 1. For
- 2. While
- 3. Loop & Statement

2. While

In [20]:

```
1 i = 1
2 while i <= 5:
3     print(i)
4     i = i + 1
```

```
1
2
3
4
5
```

2. While

In [21]:

```
1 i = 5
2 while i <= 15:
3     print(i)
4     i = i + 2
```

```
5
7
9
11
13
15
```

2. While

In [22]:

```
1 i = 0
2 while True:
3     print(i)
4     if i == 5:
5         break
6     i = i + 1
```

```
0
1
2
3
4
5
```

For & While Loop

- ✓ 1. For
- ✓ 2. While
- 3. Loop & Statement

3. Loop & Statement

3.1 break

3.2 continue

3.3 pass

3.1 break

```
In [23]: ➜ 1 for i in range(100000):  
 2     print(i)  
 3     if i == 5:  
 4         break
```

```
0  
1  
2  
3  
4  
5
```

3.2 continue

In [24]:

```
1 for i in range(10):
2     if i%2 == 0:
3         continue
4         print('hello')
5     else:
6         print(i)
```

```
1
3
5
7
9
```

3.3 pass

In [25]:

```
1 for i in range(10):
2     if i%2 == 0:
3         pass
4         print('hello')
5     else:
6         print(i)
```

```
hello
1
hello
3
hello
5
hello
7
hello
9
```

3.3 pass

In [26]:

```
1 v_list = [120, 70, 90]
2 for v in v_list:
3     if v >= 120:
4         pass
5     else:
6         print('500 baht finepage x')
```

500 baht finepage x
500 baht finepage x

For & While Loop

- ✓ 1. For
- ✓ 2. While
- ✓ 3. Loop & Statement

Function

Fundamental of Python Programming

ทำไมต้องใช้ Function

Ex. `list1 = [1, 3, 4, 7, 11]`
`list2 = [4, 3, 2, 1, 0]`

- ➔ อยากรอรวมของ `list1`
- ➔ อยากรอรวมของ `list2`

ทำไมต้องใช้ Function

Ex.

```
sum1 = 0
for i in range(len(list1)):
    sum1 = sum1+list1[i]
sum2
for i in range(len(list2)):
    sum2 = sum2+list2[i]
```

ทำไมต้องใช้ Function

Ex.

list1 = [1, 3, 4, 7, 11]

⋮

listN = [7, 5, 12, 4, 0]

➔ อยากรอรวมของ list1

⋮

➔ อยากรอรวมของ listN

ทำไมต้องใช้ Function

ช้าช่อน !

จะเขียนทำไมหลายรอบ !

ทำไมต้องใช้ Function

Ex.

```
def sum_list(list):  
    sumx = 0  
    for i in range(len(list)):  
        sumx = sumx + list[i]  
    return sumx
```

ทำไมต้องใช้ Function

Ex.

sum1 = sum_list(list1)

sum2 = sum_list(list2)

⋮

sumN = sum_list(listN)

Function

1. With Parameter
2. Without Parameter

1. With Parameter

```
In [1]: ► 1 def sum_list(listA):  
2     sumx = 0  
3     for i in range(len(listA)):  
4         sumx = sumx + listA[i]  
5     return sumx
```

```
In [2]: ► 1 list1 = [1, 2, 3, 4, 5]  
2 list2 = [6, 7, 8, 9, 10]
```

```
In [3]: ► 1 sum_list(list1)
```

Out[3]: 15

```
In [4]: ► 1 sum_list(list2)
```

Out[4]: 40

1. With Parameter

```
In [5]: ┏ 1 def plus(a, b):  
  ┏ 2   c = a + b  
  ┏ 3   return c
```

```
In [6]: ┏ 1 plus(1, 3)
```

Out[6]: 4

```
In [7]: ┏ 1 def minus(a, b):  
  ┏ 2   d = a - b  
  ┏ 3   return d
```

```
In [8]: ┏ 1 minus(1, 3)
```

Out[8]: -2

1. With Parameter

```
In [9]: ► 1 def plus_minus(a, b, c):  
 2     c = a + b + c  
 3     d = a - b - c  
 4     return c, d
```

```
In [10]: ► 1 result1, result2 = plus_minus(1, 3, 5)
```

```
In [11]: ► 1 result1
```

Out[11]: 9

```
In [12]: ► 1 result2
```

Out[12]: -11

Function

- ✓ 1. With Parameter
- 2. Without Parameter

2. Without Parameter

In [13]: ►

```
1 def tree():
2     print('''
3         ---1---
4         --121--
5         -12321-
6         1234321'''')
```

In [14]: ►

```
1 tree()
```

```
---1---
--121--
-12321-
1234321
```

2. Without Parameter

In [15]:

```
1 def PYTHON():
2     print('
3     --XXXXXX--XX----XX--XXXXXXXX--XX----XX----XXXXXX----XX----XX--
4     --XX----XX--XX----XX----XX----XX----XX----XX----XX----XXX----XX--
5     --XX----XX--XX--XX----XX----XX----XX----XX----XX----XX----XXXX--XX--
6     --XXXXXX----XXXX----XX----XXXXXXXX--XX----XX--XX--XX--XX--XX--XX--
7     --XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXX--'
8     --XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXX--'
9     --XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXXXX----XX----XX--')
```

In [16]:

```
1 PYTHON()
```

```
--XXXXXX--XX----XX--XXXXXXXX--XX----XX----XXXXXX----XX----XX--
--XX----XX--XX----XX----XX----XX----XX----XX----XX----XX----XXX----XX--
--XX----XX--XX--XX----XX----XX----XX----XX----XX----XX----XX----XXXX--XX--
--XXXXXX----XXXX----XX----XXXXXXXX--XX----XX--XX--XX--XX--XX--XX--XX--
--XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXX--'
--XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXX--'
--XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XX----XXXXXX----XX----XX--'
```

Function

- ✓ 1. With Parameter
- ✓ 2. Without Parameter

OOP

Fundamental of Python Programming

ทำความรู้จัก OOP

- 1. OOP เป็นเทคนิคการเขียนโปรแกรมแบบหนึ่ง ที่มุ่งทุกอย่างเป็น Object**
- 2. การเขียน Code แบบ OOP ทำให้ง่าย & เป็นระเบียบ ในการ พัฒนา/ปรับปรุง**

OOP

1. Attribute
2. Method

1. Attribute

```
In [1]: ┏ 1 class Warrior:  
      2     def __init__(self, HP, ATK, DEF):  
      3         self.HP = HP  
      4         self.ATK = ATK  
      5         self.DEF = DEF
```

```
In [2]: ┏ 1 warrior1 = Warrior(100, 45, 25)
```

```
In [3]: ┏ 1 print("warrior1's HP =", warrior1.HP)  
      2 print("warrior1's ATK =", warrior1.ATK)  
      3 print("warrior1's DEF =", warrior1.DEF)
```

```
warrior1's HP = 100  
warrior1's ATK = 45  
warrior1's DEF = 25
```

1. Attribute

```
In [4]: ┏ 1 warrior2 = Warrior(100, 35, 30)
```

```
In [5]: ┏ 1 print("warrior2's HP =", warrior2.HP)
      2 print("warrior2's HP =", warrior2.ATK)
      3 print("warrior2's HP =", warrior2.DEF)
```

```
warrior2's HP = 100
warrior2's HP = 35
warrior2's HP = 30
```

1. Attribute

In [6]:

```
1 class Car:  
2     def __init__(self, brand, model, color):  
3         self.brand = brand  
4         self.model = model  
5         self.color = color
```

In [7]:

```
1 car1 = Car('Toyota', 'Camry', 'Black')
```

In [8]:

```
1 print("car1's brand =", car1.brand)  
2 print("car1's model =", car1.model)  
3 print("car1's color =", car1.color)
```

```
car1's brand = Toyota  
car1's model = Camry  
car1's color = Black
```

1. Attribute

```
In [9]: ► 1 car2 = Car('Honda', 'Civic', 'White')
```

```
In [10]: ► 1 print("car2's brand =", car2.brand)
2 print("car2's model =", car2.model)
3 print("car2's color =", car2.color)
```

```
car2's brand = Honda
car2's model = Civic
car2's color = White
```

OOP

- ✓ 1. Attribute
- 2. Method

2. Method

In [11]: ►

```
1 class Warrior:
2     def __init__(self, HP, ATK, DEF):
3         self.HP = HP
4         self.ATK = ATK
5         self.DEF = DEF
6
7     def training(self):
8         self.HP = self.HP + 5
9         self.ATK = self.ATK + 10
10        self.DEF = self.DEF + 5
```

In [12]: ►

```
1 warrior1 = Warrior(100, 45, 25)
2 print('HP = %d, ATK = %d, DEF = %d' %(warrior1.HP, warrior1.ATK, warrior1.DEF))
3 warrior1.training()
4 print('HP = %d, ATK = %d, DEF = %d' %(warrior1.HP, warrior1.ATK, warrior1.DEF))
```

```
HP = 100, ATK = 45, DEF = 25
HP = 105, ATK = 55, DEF = 30
```

2. Method

In [13]: ►

```
1 class Car:
2     def __init__(self, brand, model, color):
3         self.brand = brand
4         self.model = model
5         self.color = color
6
7     def set_color(self, color):
8         self.color = color
```

In [14]: ►

```
1 car1 = Car('Toyota', 'Camry', 'Black')
2 print('old color = %s' %car1.color)
3 car1.set_color('Red')
4 print('new color = %s' %car1.color)
```

```
old color = Black
new color = Red
```

OOP

- ✓ 1. Attribute
- ✓ 2. Method

File Management

Fundamental of Python Programming

File Management

1. **txt**
2. **Path**

1. txt

1.1 write (w)

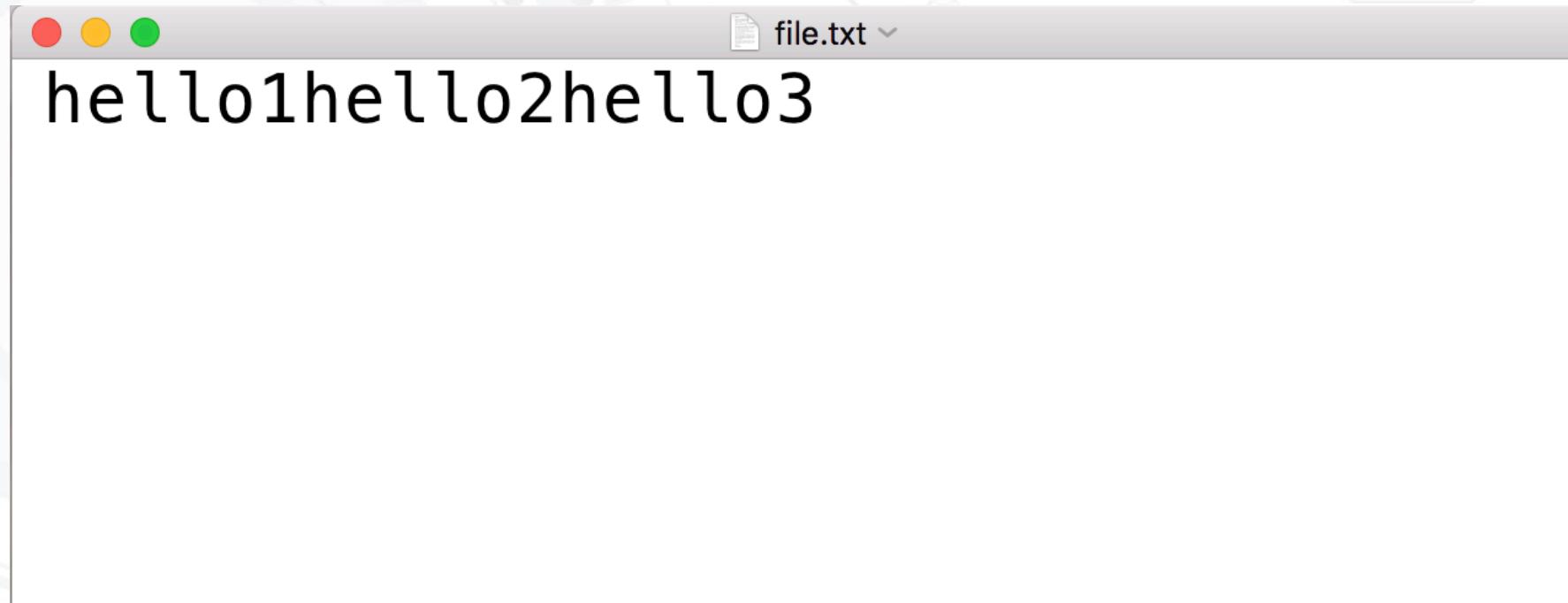
1.2 read (r)

1.3 append (a)

1.1 write (w)

```
In [1]: file = open('file.txt', 'w', encoding='utf8')
file.write('hello1')
file.write('hello2')
file.write('hello3')
file.close()
```

1.1 write (w)



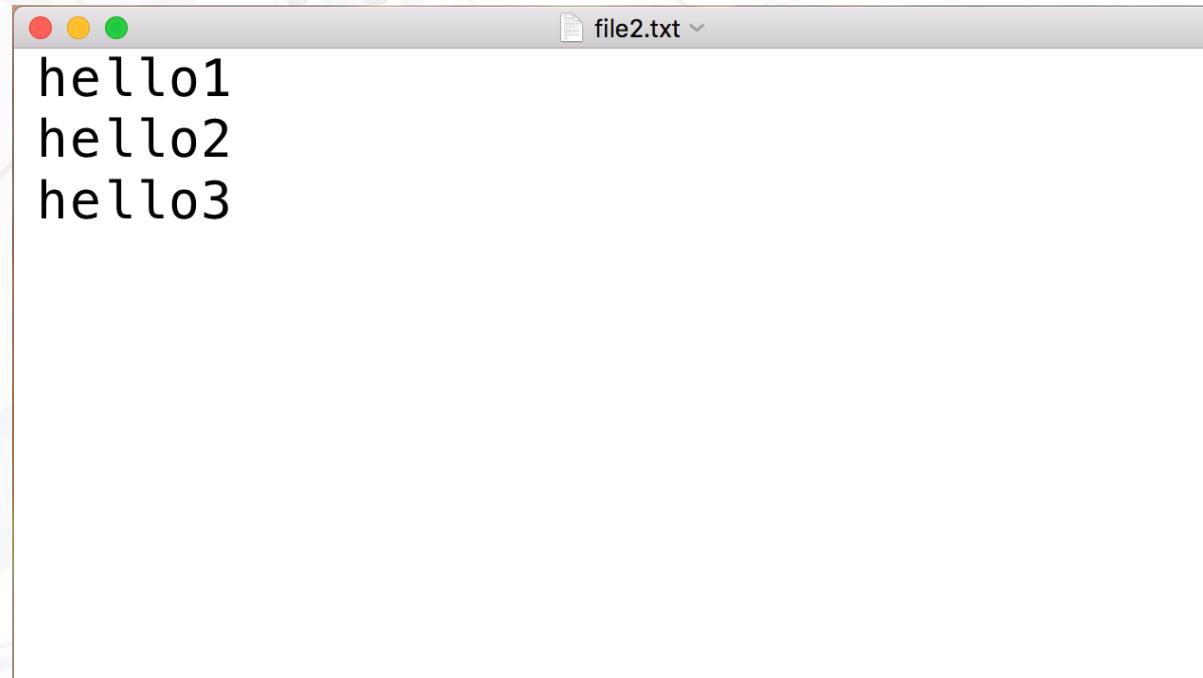
1.1 write (w)

```
In [2]: file2 = open('file2.txt', 'w', encoding='utf8')
file2.write('hello1' + '\n')
file2.write('hello2' + '\n')
file2.write('hello3' + '\n')
file2.close()
```

1.1 write (w)

```
In [3]: file2 = open('file2.txt', 'w', encoding='utf8')
file2.write('hello1\n')
file2.write('hello2\n')
file2.write('hello3\n')
file2.close()
```

1.1 write (w)



1.1 write (w)

```
In [6]: file3 = open('file3.txt', 'w', encoding='utf8')
file3.write(1)
file3.close()
```

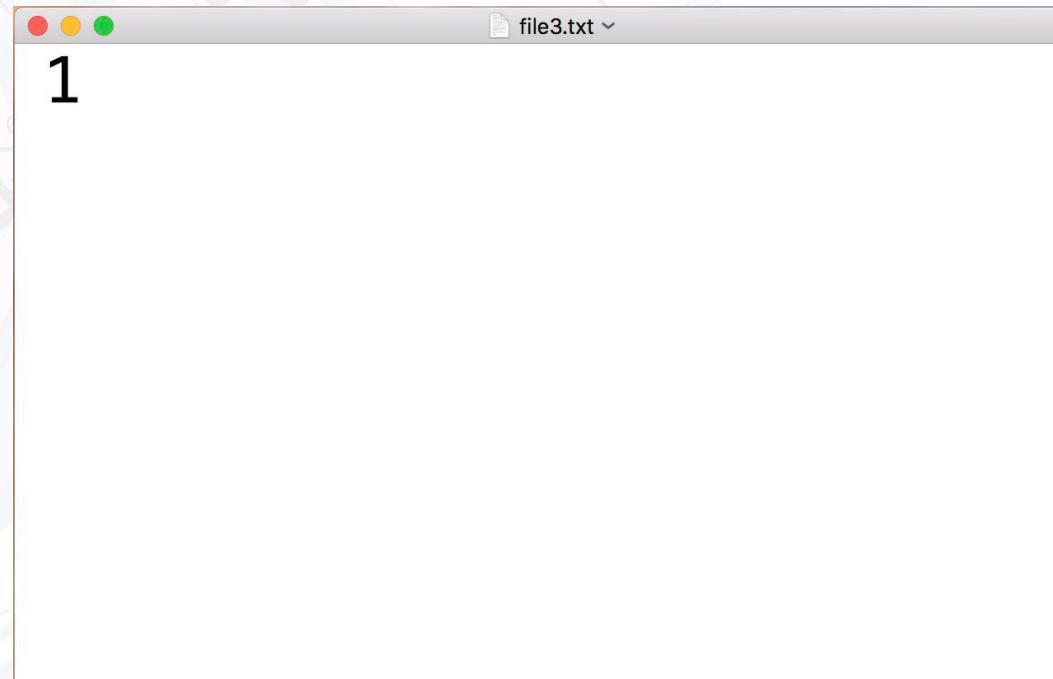
```
-----
--> TypeError
t)
<ipython-input-6-9b573a3a285f> in <module>()
      1 file3 = open('file3.txt', 'w', encoding='utf8')
----> 2 file3.write(1)
      3 file3.close()
```

```
TypeError: write() argument must be str, not int
```

1.1 write (w)

```
In [7]: file3 = open('file3.txt', 'w', encoding='utf8')
file3.write('1')
file3.close()
```

1.1 write (w)



1.1 write (w)

```
In [8]: file4 = open('file4.txt', 'w', encoding='utf8')
file4.write('1')
file4.write('2')
file4.write('3')
file4.close()
```

1.1 write (w)



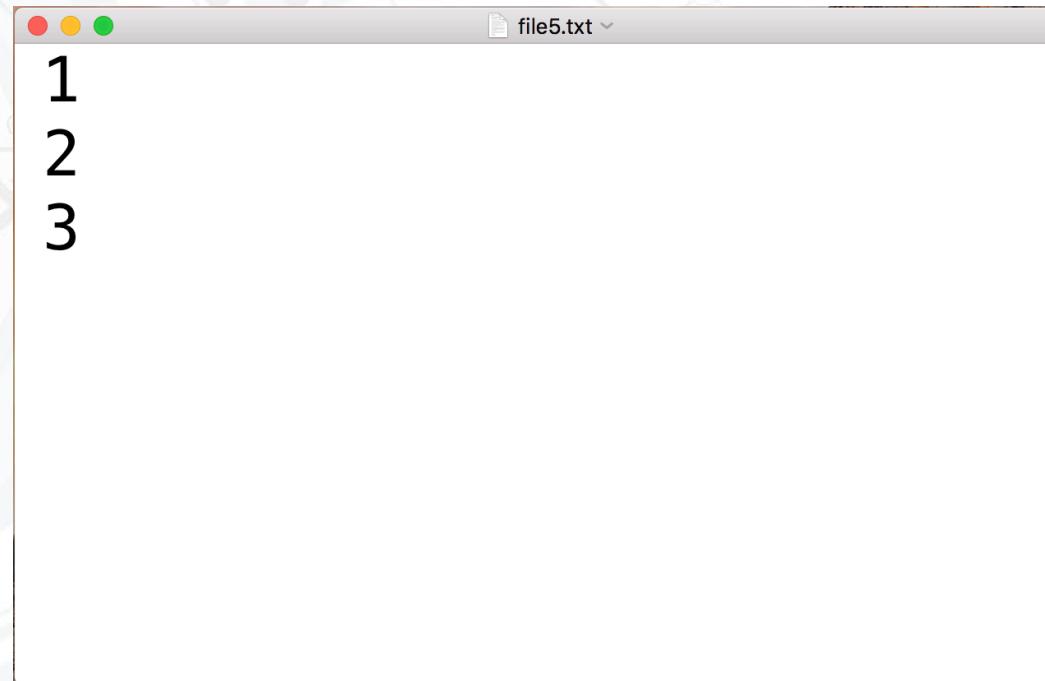
1.1 write (w)

```
In [9]: file5 = open('file5.txt', 'w', encoding='utf8')
file5.write('1' + '\n')
file5.write('2' + '\n')
file5.write('3' + '\n')
file5.close()
```

1.1 write (w)

```
In [10]: file5 = open('file5.txt', 'w', encoding='utf8')
file5.write('1\n')
file5.write('2\n')
file5.write('3\n')
file5.close()
```

1.1 write (w)



1. txt

1.1 write (w) 

1.2 read (r)

1.3 append (a)

1.2 read (r)

```
In [4]: file2 = open('file2.txt', 'r', encoding='utf8')
for line in file2:
    print(line)
```

hello1

hello2

hello3

1.2 read (r)

```
In [5]: file2 = open('file2.txt', 'r', encoding='utf8')
for line in file2:
    print(line[:-1])
```

```
hello1
hello2
hello3
```

1. txt

1.1 write (w) 

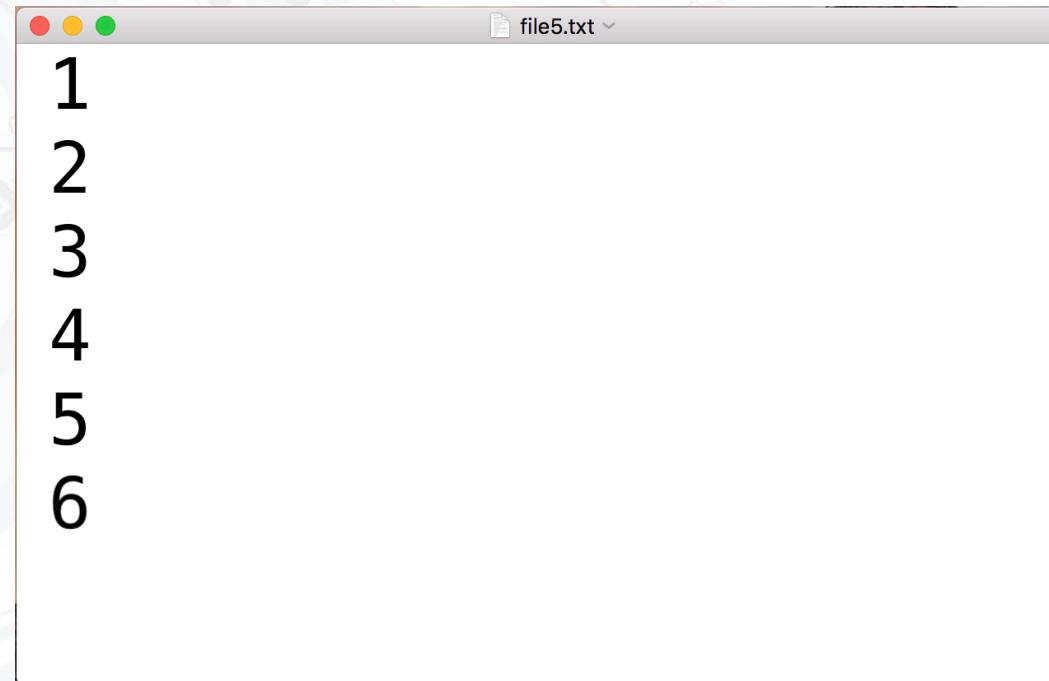
1.2 read (r) 

1.3 append (a)

1.3 append (a)

```
In [11]: file5 = open('file5.txt', 'a', encoding='utf8')
file5.write('4' + '\n')
file5.write('5' + '\n')
file5.write('6' + '\n')
file5.close()
```

1.3 append (a)



1. txt

1.1 write (w) 

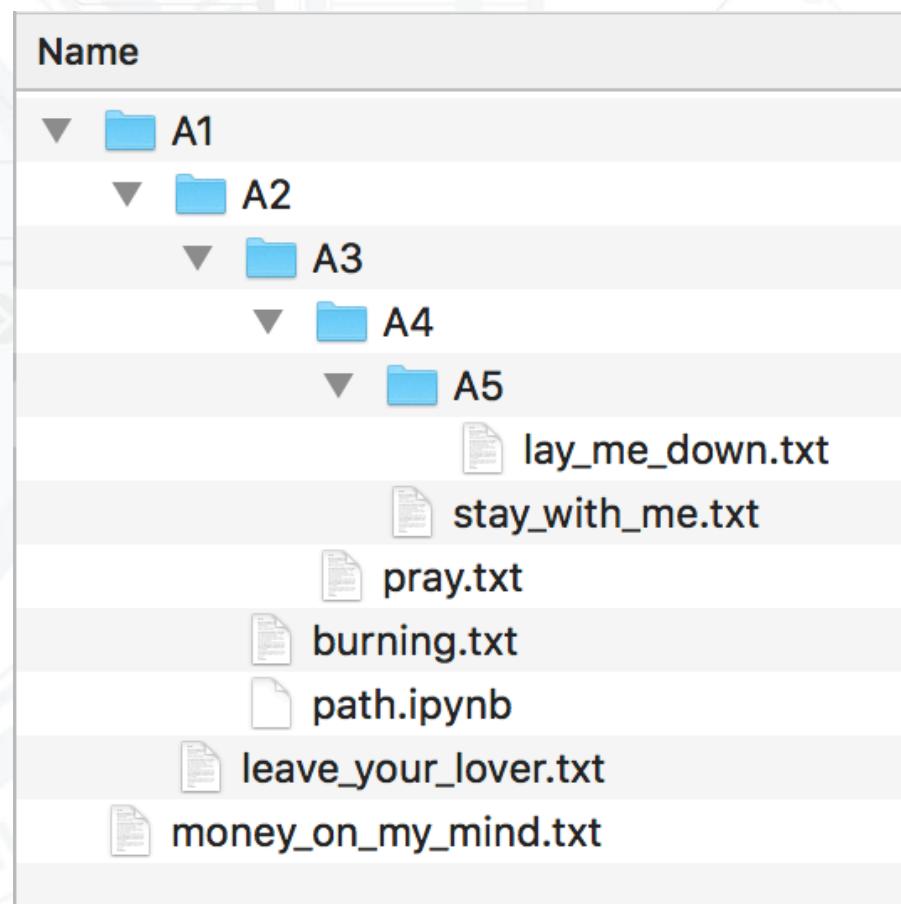
1.2 read (r) 

1.3 append (a) 

File Management

- ✓ 1. txt
- 2. Path

2. path



2. path

```
In [1]: money_on_my_mind = open('.../.../money_on_my_mind.txt')
```

```
In [2]: leave_your_lover = open('.../leave_your_lover.txt', 'r', encoding='utf8')
```

```
In [3]: burning = open('burning.txt', 'r', encoding='utf8')
```

```
In [4]: stay_with_me = open('A3/pray.txt', 'r', encoding='utf8')
```

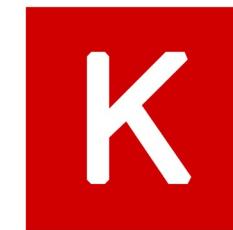
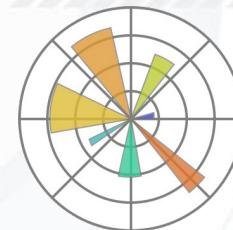
```
In [5]: lay_me_down = open('A3/A4/A5/lay_me_down.txt', 'r', encoding='utf8')
```

File Management

- ✓ 1. txt
- ✓ 2. Path

Interesting Library

Fundamental of Python Programming





NumPy



library สำหรับการคำนวณทางวิทยาศาสตร์และ
การวิเคราะห์ข้อมูล



pandas

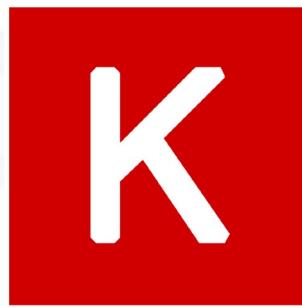
library สำหรับการจัดการข้อมูลและวิเคราะห์ข้อมูล



library สำหรับทำ data visualization

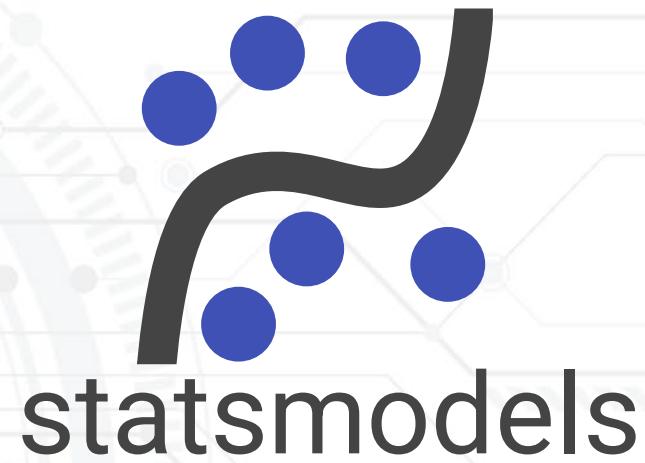


library สำหรับงานด้าน machine learning



PyTorch

library สำหรับงานด้าน deep learning



library สำหรับการสร้างแบบจำลองทางสถิติ

303



library สำหรับทำ natural language processing

BeautifulSoup



library สำหรับกำ web scraping



library สำหรับกำ automating task (ควบคุมเมาส์ & keyboard อัตโนมัติ)



library สำหรับงานด้าน computer vision และ
image processing



library สำหรับการพัฒนาเกม

308



Flask

web development,
one drop at a time

django

library สำหรับ web development

Introduction to ChatGPT

Working with ChatGPT

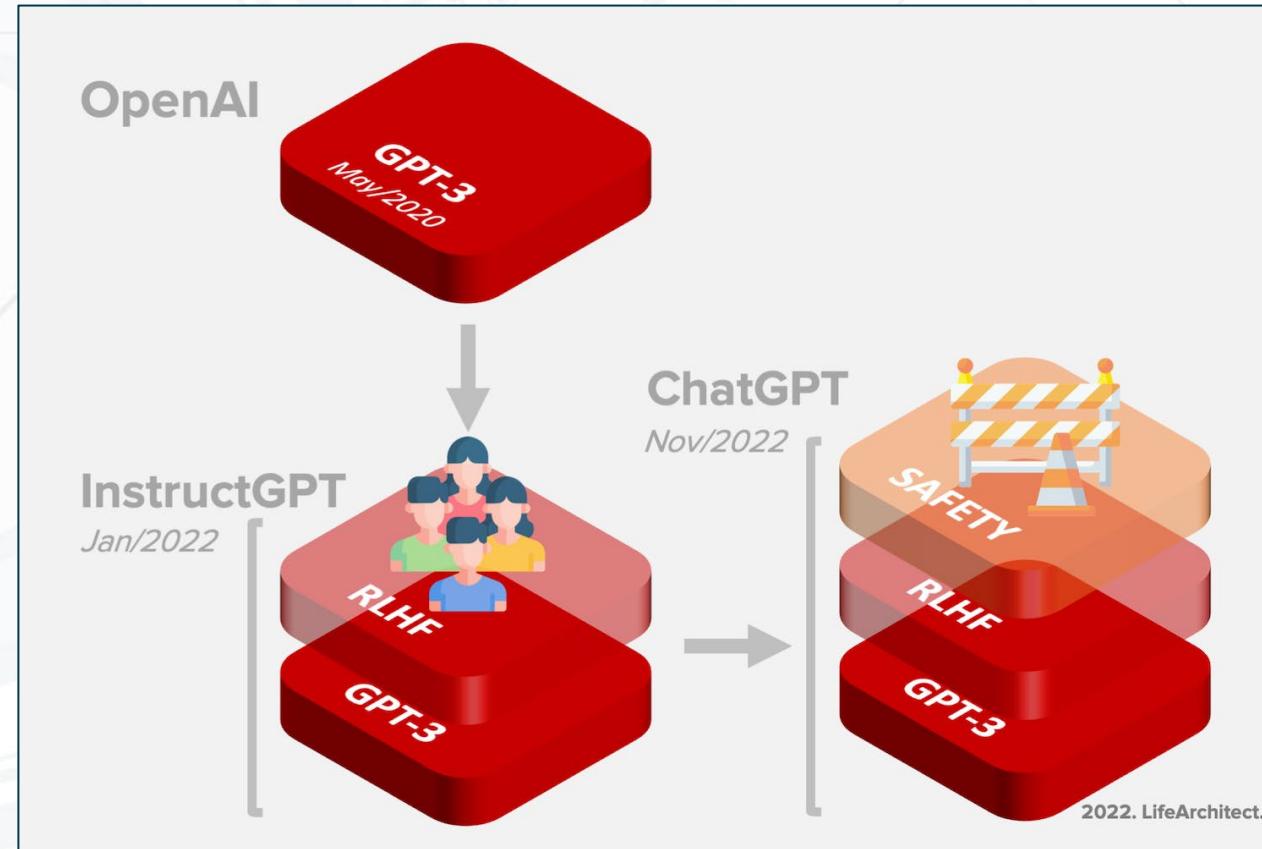
Introduction to ChatGPT

1. What is ChatGPT ?
2. GPT-4 Architecture
3. Capacity as Python assistance
4. Evolution of ChatGPT
5. Limitations as Python assistance

1. What is ChatGPT ?

ChatGPT is an advanced AI language model by OpenAI that assists with tasks like answering questions and engaging in conversation.

2. GPT-4 Architecture



3. Capacity as Python assistance

3.1 Code suggestion & Code completion

3.2 Error explanation

3.3 Debugging assistance

3.4 best practices Recommendation

3.5 Code concepts explanation

3.6 Library recommendation

4. Evolution of ChatGPT

Name	Year	Improvements	Capacity	Parameters
GPT	2018	Initial Release	Basic language understanding, limited context, short memory	117M
GPT-2	2019	Enhanced context understanding, larger model	Better context understanding, more coherent text generation	1.5B
GPT-3	2020	Few-shot learning, more sophisticated language	High-quality text generation, broader language understanding	175B
GPT-3.5 (InstructGPT)	2021	Fine-tuned for chat-based applications, human feedback	Improved conversation quality, versatile chat-based AI	175B
GPT-4 (ChatGPT)	2023	Enhanced language, knowledge, reasoning	Improved reasoning, advanced context handling, deeper knowledge	1T

5. Limitations as Python assistance

5.1 Knowledge cutoff

5.2 Inaccuracies

5.3 Limited problem-solving capacity

5.4 Lack of human intuition

5.5 Limited interactivity

5.6 Inability to run code

How to Use ChatGPT-Assistance

Working with ChatGPT

How to Use ChatGPT-Assistance

1. Access and Setup
2. ChatGPT-Webchat vs ChatGPT-Playground
3. ChatGPT-Webchat
4. ChatGPT-Playground
5. Effective Communication

1. Access and Setup

★ ChatGPT-Webchat

<https://chat.openai.com>

★ ChatGPT-Playground

<https://platform.openai.com/playground>

How to Use ChatGPT-Assistance

- ✓ 1. Access and Setup
- 2. ChatGPT-Webchat vs ChatGPT-Playground
- 3. ChatGPT-Webchat
- 4. ChatGPT-Playground
- 5. Effective Communication

2. ChatGPT-Webchat vs ChatGPT-Playground

★ ChatGPT-Webchat

- ❖ Designed for a user-friendly experience
- ❖ Simple and minimal interface
- ❖ Easy to use

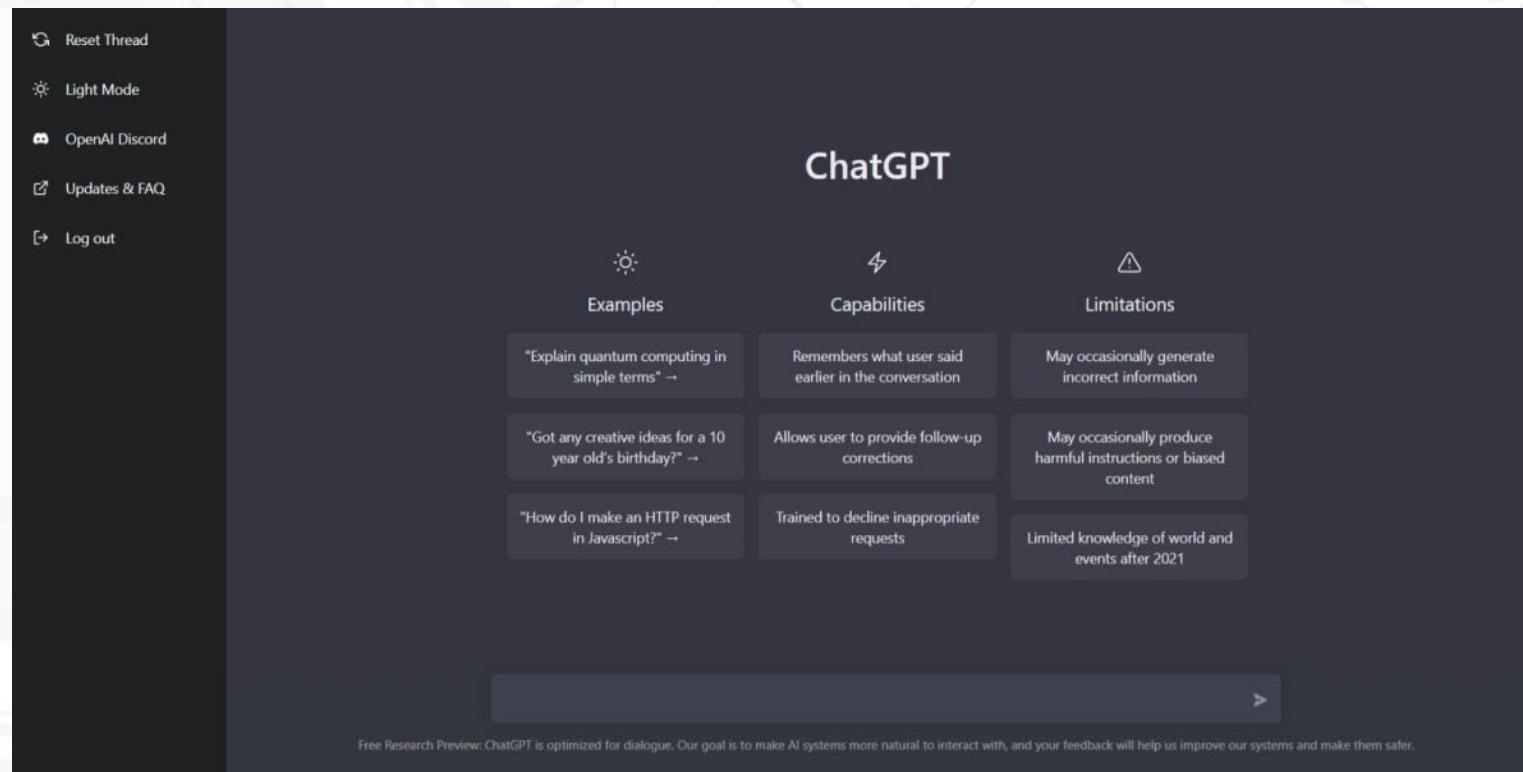
★ ChatGPT-Playground

- ❖ Designed for a more advanced and customizable experience
- ❖ Offers additional options for controlling the conversation, such as adjusting temperature (randomness) and max tokens (length)

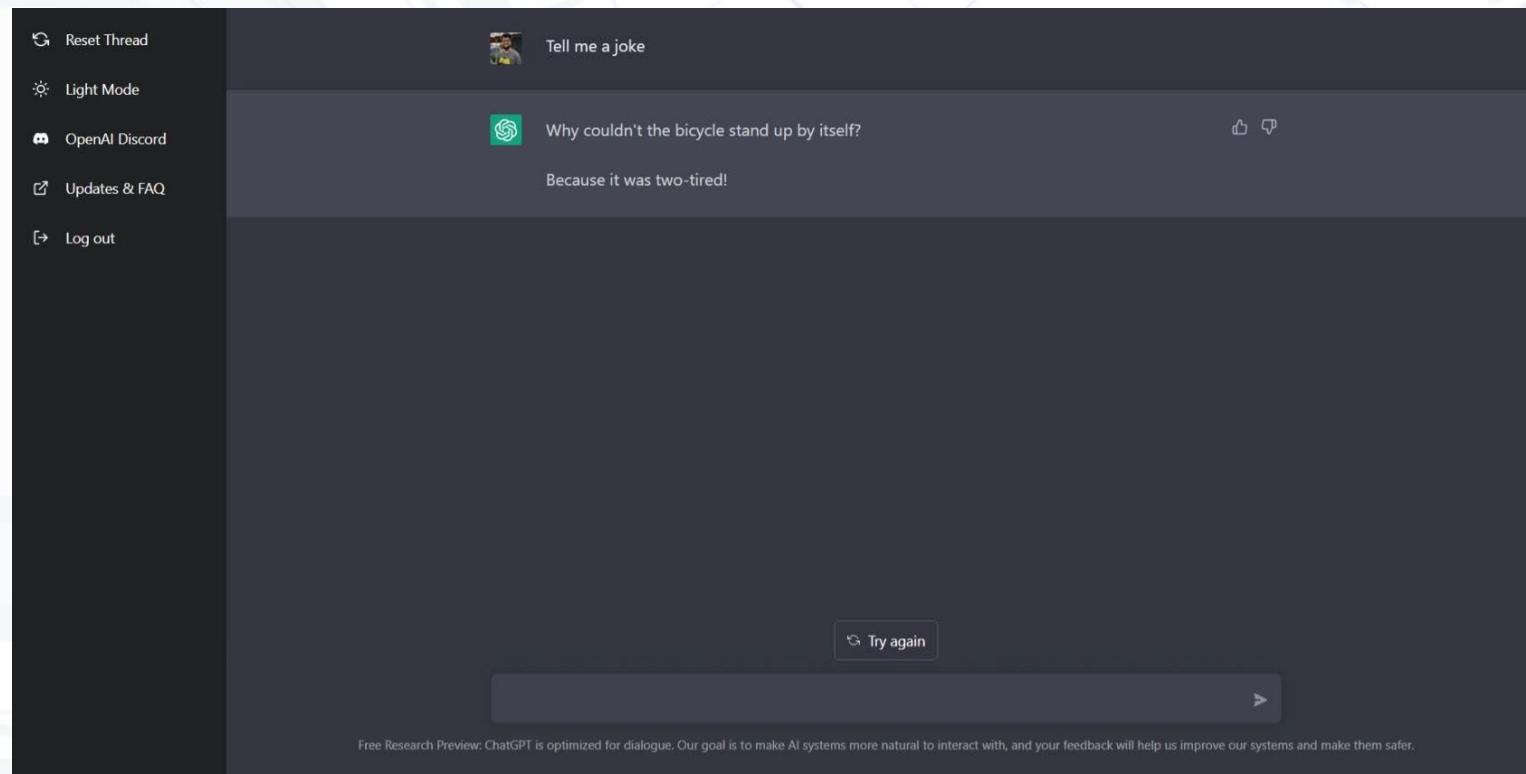
How to Use ChatGPT-Assistance

- ✓ 1. Access and Setup
- ✓ 2. ChatGPT-Webchat vs ChatGPT-Playground
- 3. ChatGPT-Webchat
- 4. ChatGPT-Playground
- 5. Effective Communication

3. ChatGPT-Webchat



3. ChatGPT-Webchat



324

How to Use ChatGPT-Assistance

- ✓ 1. Access and Setup
- ✓ 2. ChatGPT-Webchat vs ChatGPT-Playground
- ✓ 3. ChatGPT-Webchat
- 4. ChatGPT-Playground
- 5. Effective Communication

4. ChatGPT-Playground

4.1 Feature

4.2 Parameter

4.1 Feature

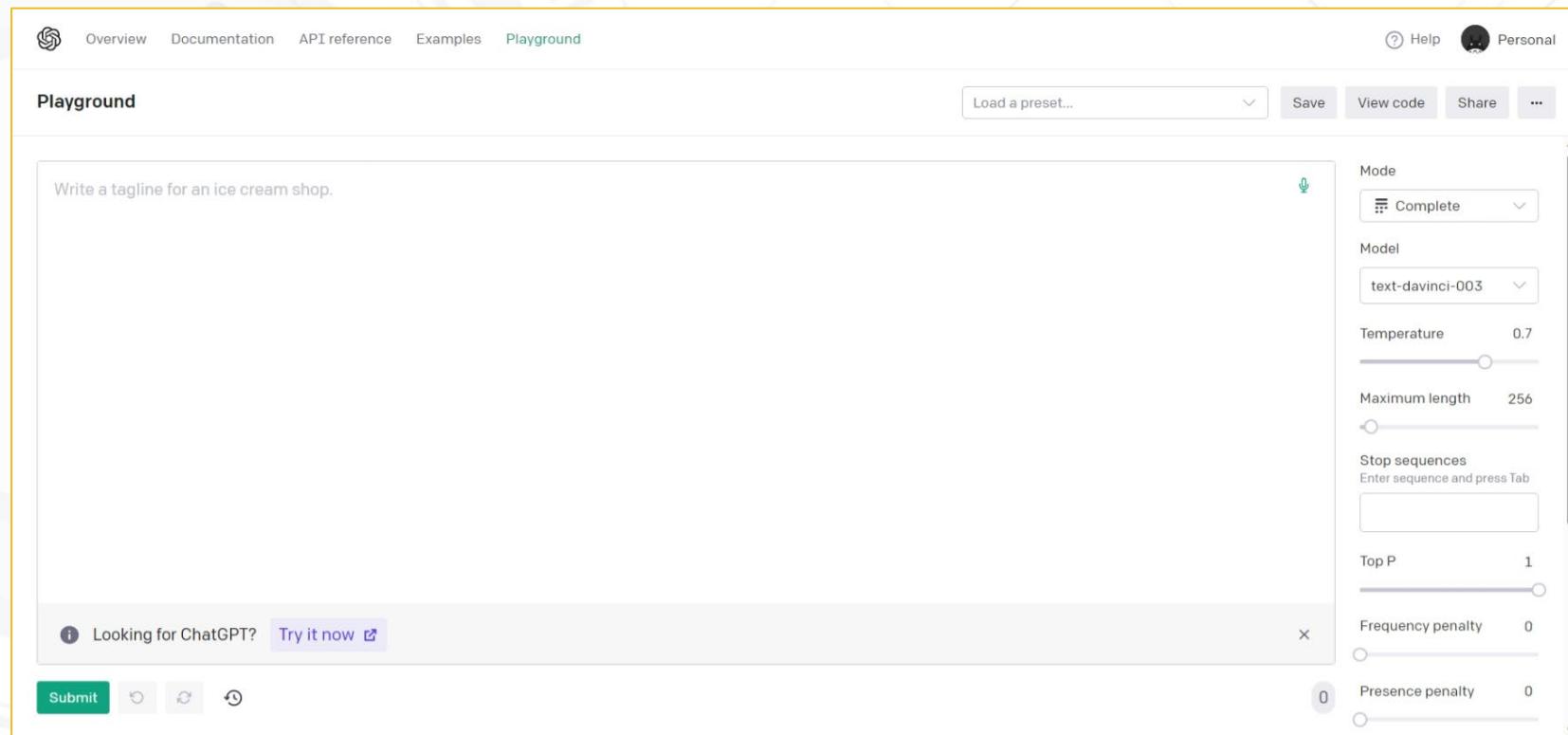
4.1.1 Complete

4.1.2 Chat

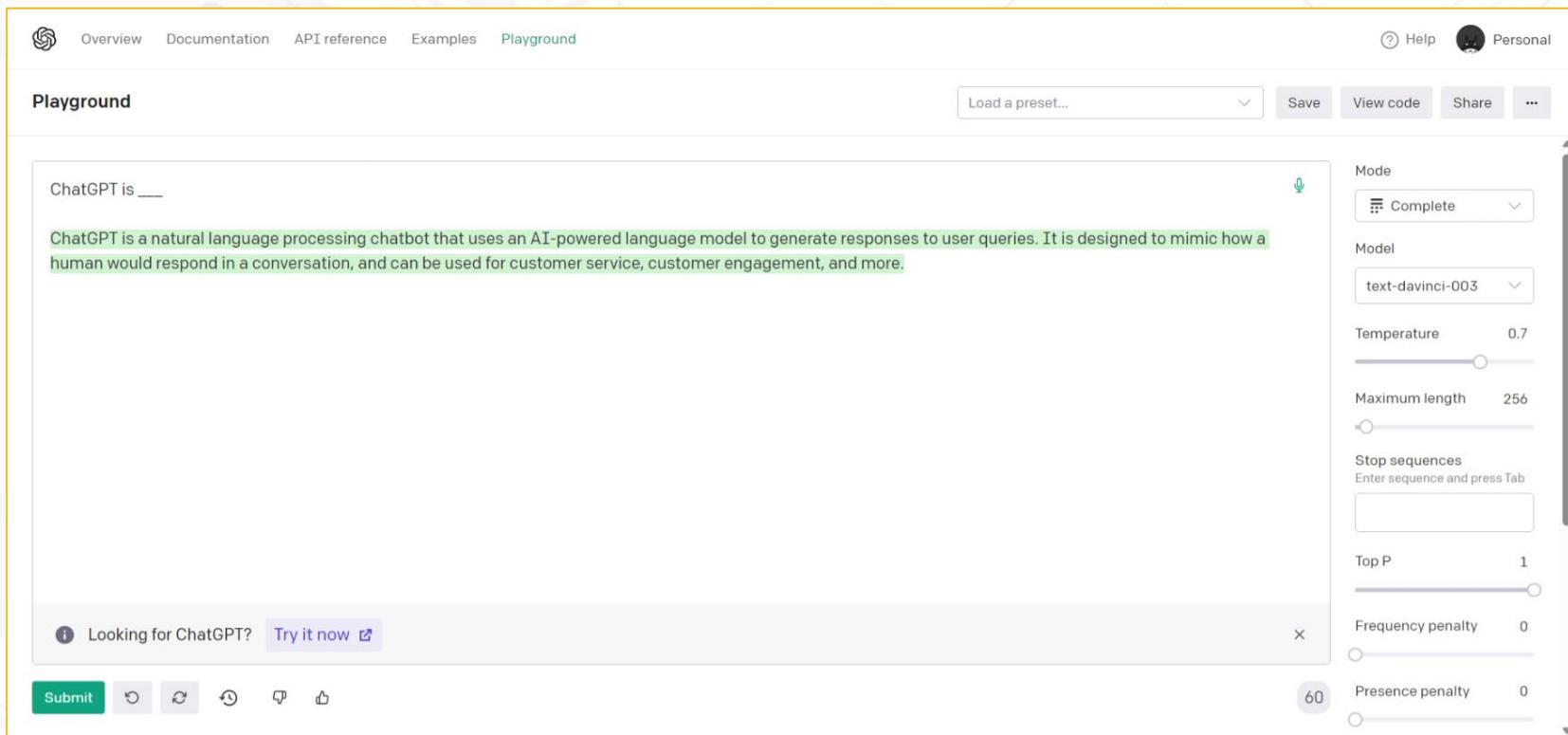
4.1.3 Insert

4.1.4 Edit

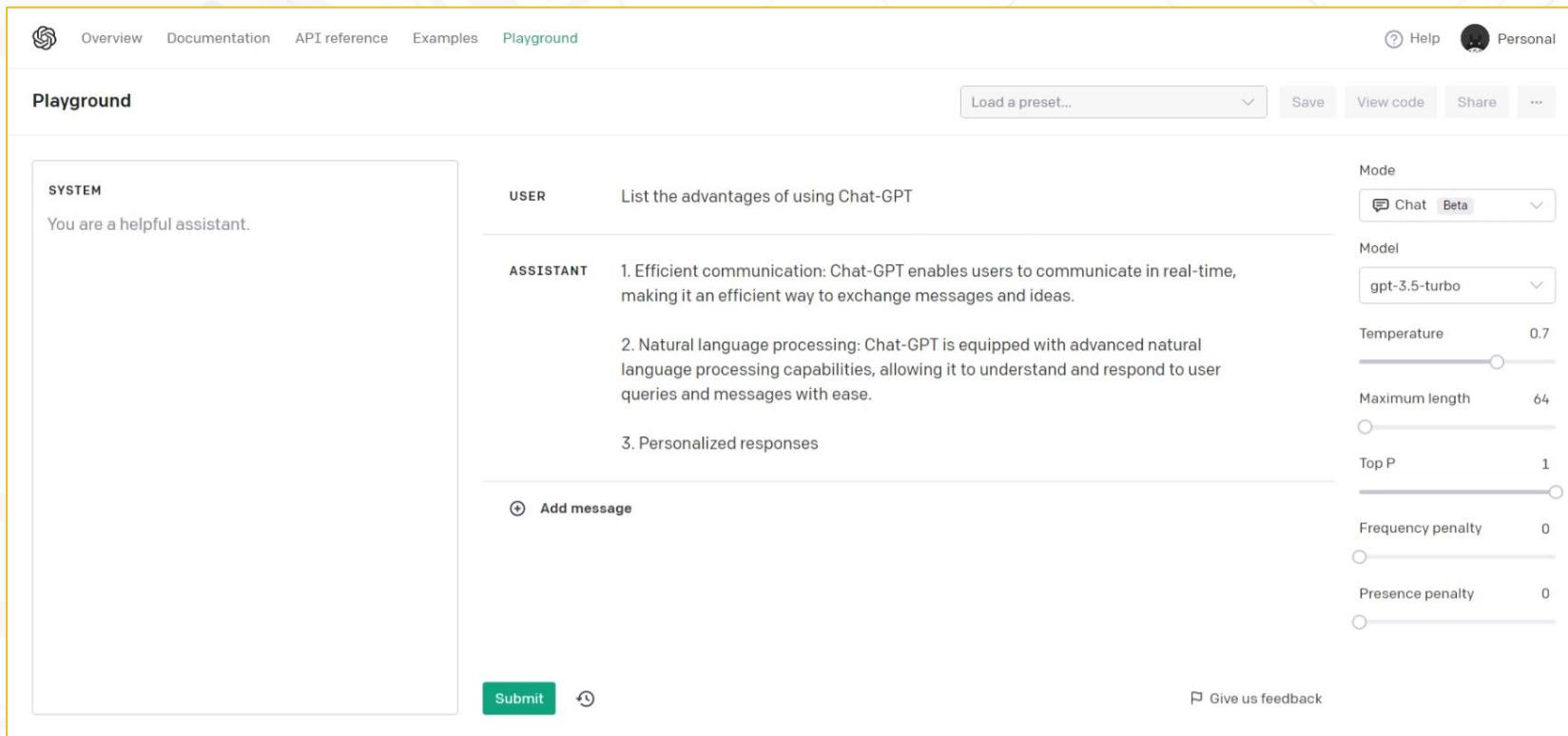
4.1.1 Complete



4.1.1 Complete

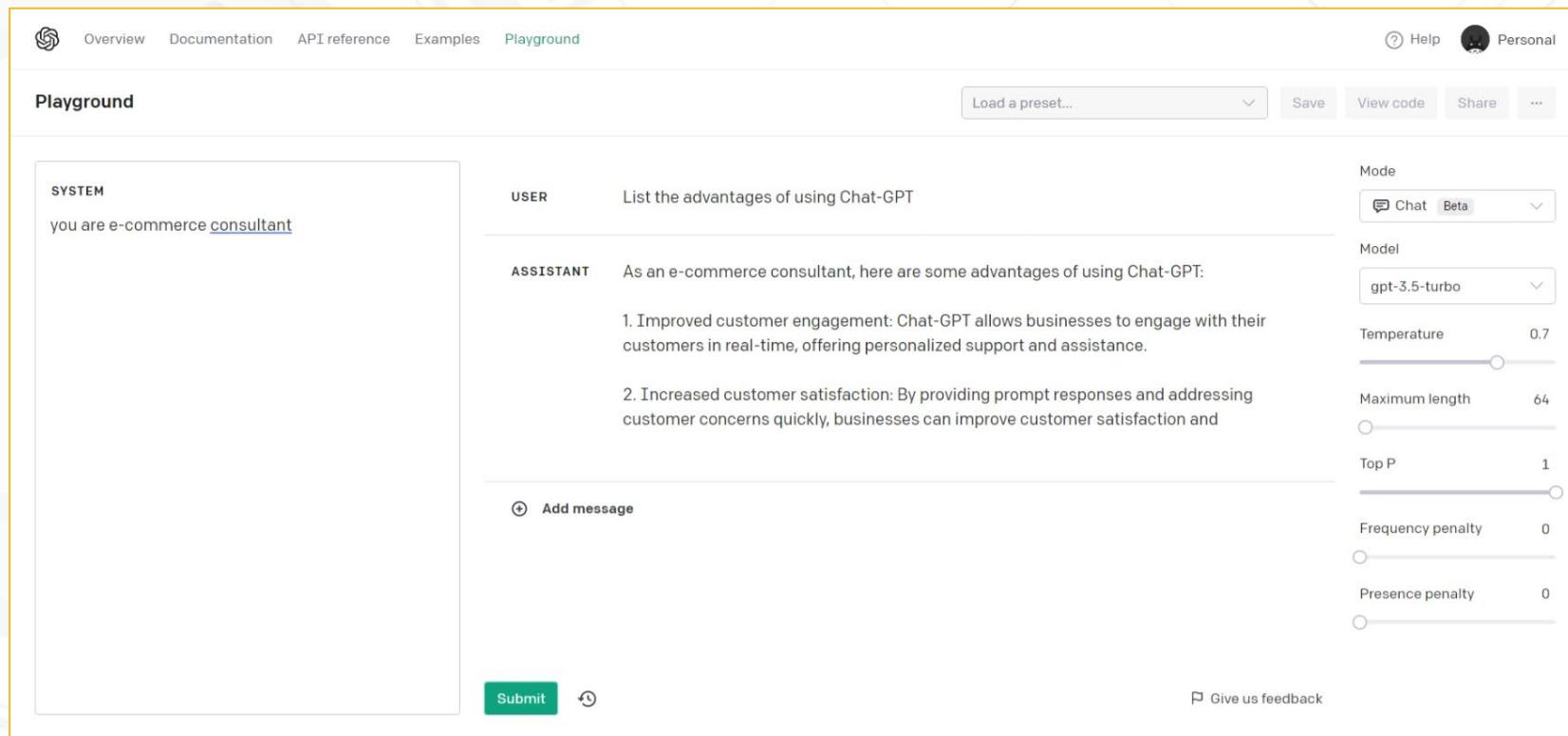


4.1.2 Chat



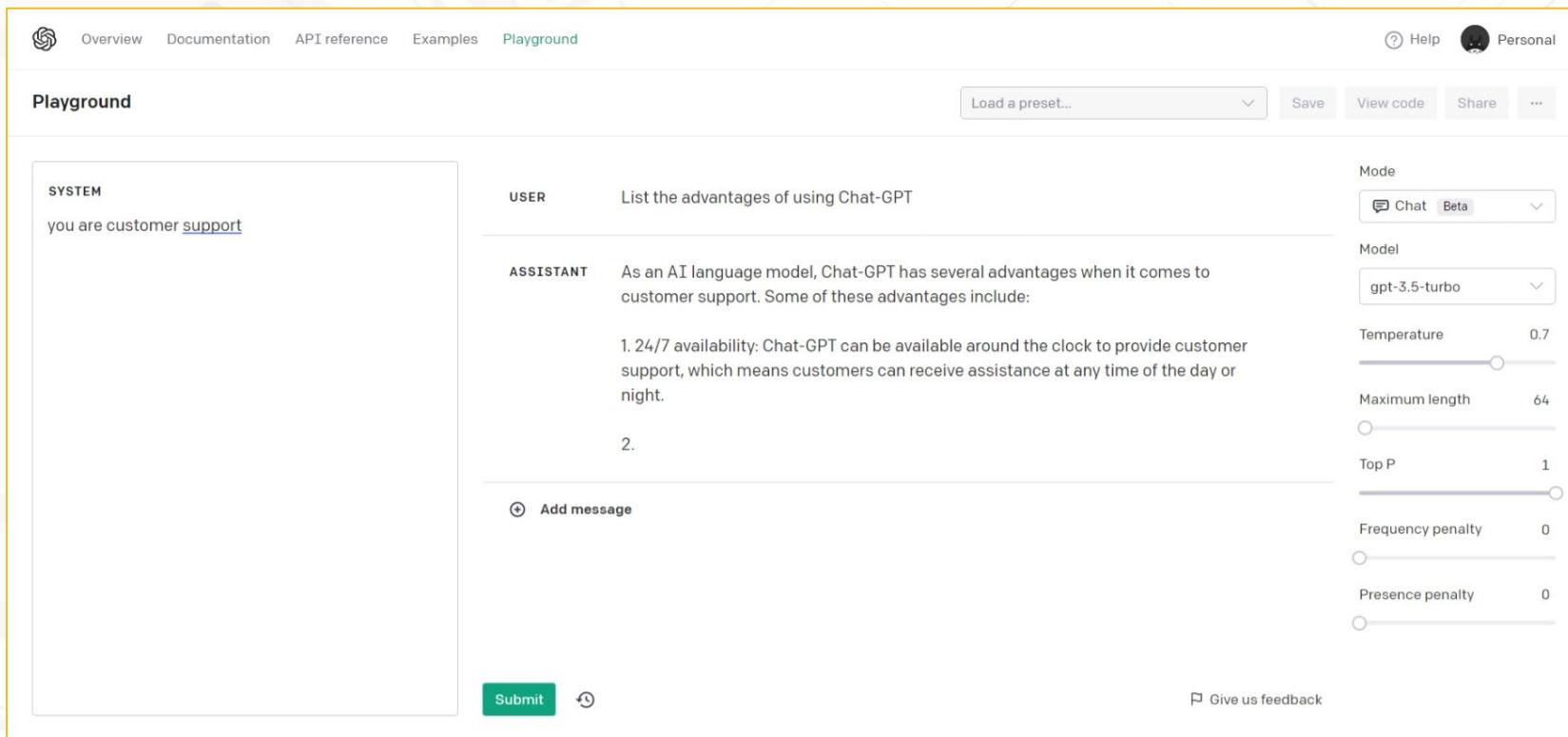
The image shows a screenshot of the ChatGPT playground interface. At the top, there is a navigation bar with links to Overview, Documentation, API reference, Examples, and Playground. On the right side of the navigation bar are Help, Personal, and a three-dot menu. Below the navigation bar, the word "Playground" is displayed in bold. On the left, there is a "SYSTEM" section containing the text "You are a helpful assistant." In the center, there is a "USER" section with the text "List the advantages of using Chat-GPT". Below this, there is an "ASSISTANT" section with three numbered points: 1. Efficient communication: Chat-GPT enables users to communicate in real-time, making it an efficient way to exchange messages and ideas. 2. Natural language processing: Chat-GPT is equipped with advanced natural language processing capabilities, allowing it to understand and respond to user queries and messages with ease. 3. Personalized responses. At the bottom of the central area, there is a "Submit" button and a "Give us feedback" checkbox. On the right side, there is a sidebar with various configuration options: Mode (Chat, Beta), Model (gpt-3.5-turbo), Temperature (0.7), Maximum length (64), Top P (1), Frequency penalty (0), and Presence penalty (0). There is also a "Load a preset..." dropdown, a "Save" button, a "View code" button, a "Share" button, and a three-dot menu button.

4.1.2 Chat



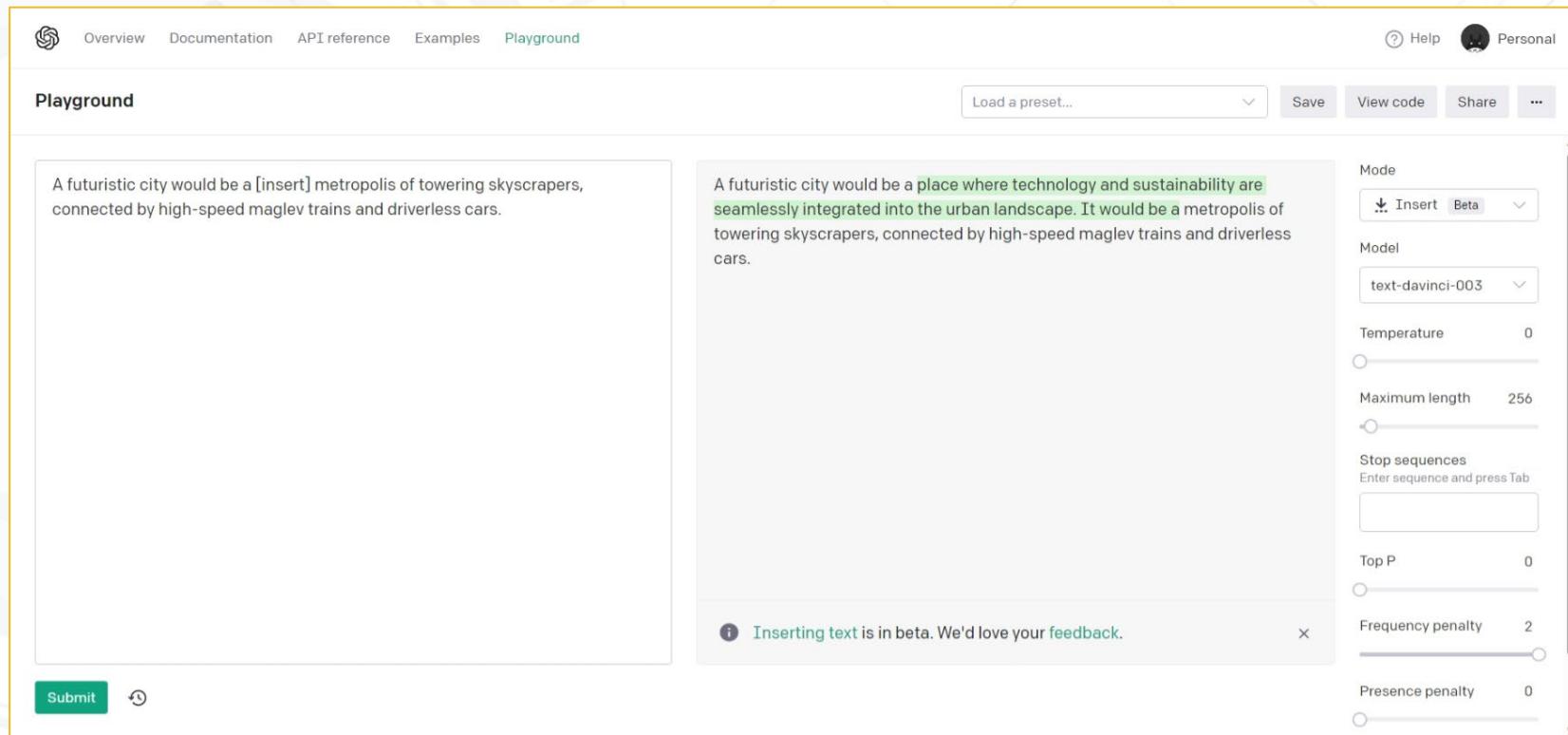
The image shows a screenshot of the ChatGPT playground interface. At the top, there is a navigation bar with links to Overview, Documentation, API reference, Examples, and Playground. On the right side of the navigation bar are Help, Personal, and a user icon. Below the navigation bar, the word "Playground" is displayed in bold. On the left, there is a sidebar labeled "SYSTEM" containing the text "you are e-commerce consultant". The main area shows a conversation between "USER" and "ASSISTANT". The user prompt is "List the advantages of using Chat-GPT". The assistant response is: "As an e-commerce consultant, here are some advantages of using Chat-GPT: 1. Improved customer engagement: Chat-GPT allows businesses to engage with their customers in real-time, offering personalized support and assistance. 2. Increased customer satisfaction: By providing prompt responses and addressing customer concerns quickly, businesses can improve customer satisfaction and". Below the conversation, there is a "Add message" button with a plus sign. On the right side, there are several configuration options: Mode (Chat, Beta), Model (gpt-3.5-turbo), Temperature (0.7), Maximum length (64), Top P (1), Frequency penalty (0), and Presence penalty (0). At the bottom of the interface, there are "Submit" and "Give us feedback" buttons.

4.1.2 Chat



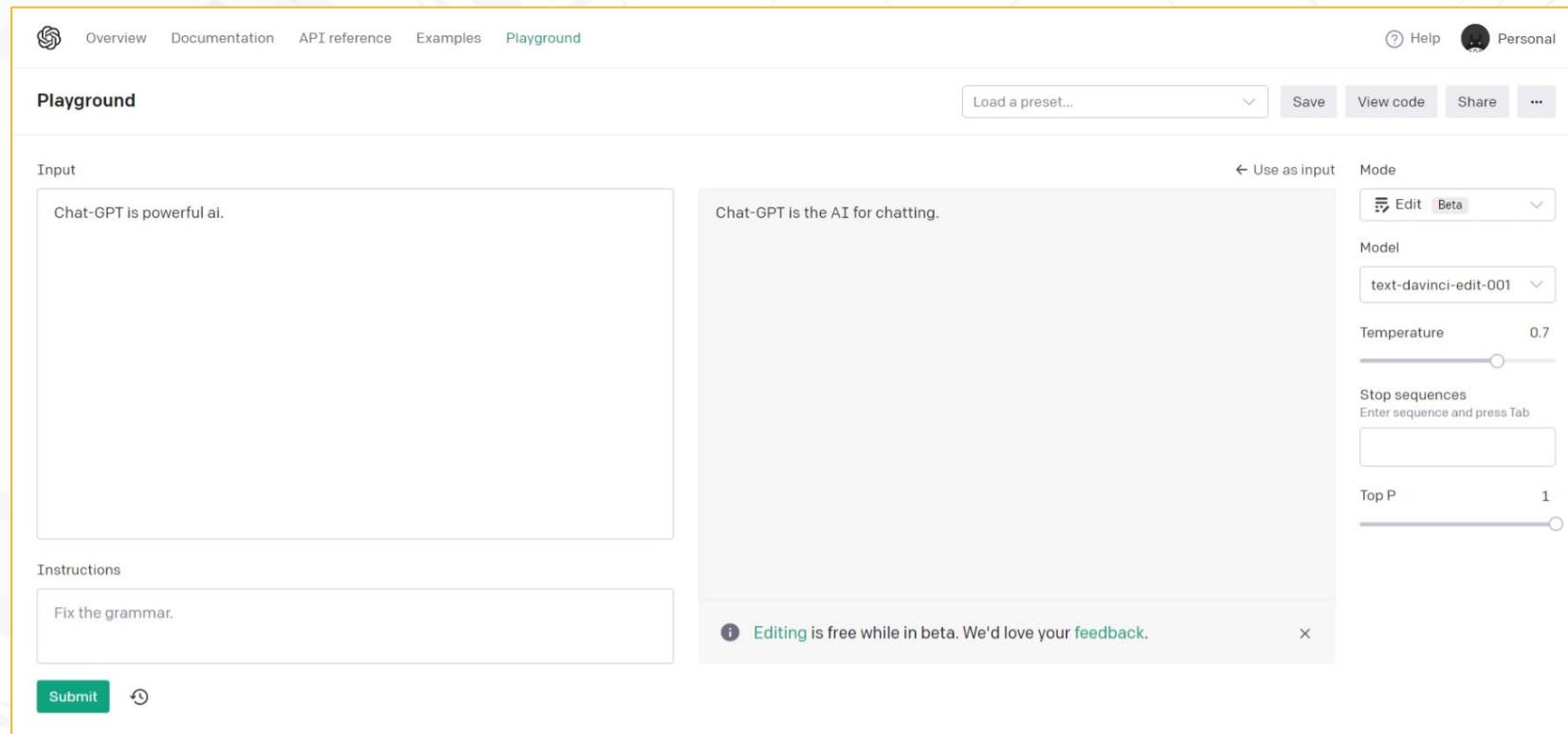
The image shows a screenshot of the Chat-GPT playground interface. At the top, there is a navigation bar with links to Overview, Documentation, API reference, Examples, and Playground. On the right side of the navigation bar are Help, Personal, and a user icon. Below the navigation bar, the word "Playground" is displayed in bold. The main area is a conversation interface with two columns: "SYSTEM" and "USER". The "SYSTEM" column contains the message "you are customer support". The "USER" column contains the message "List the advantages of using Chat-GPT". The "ASSISTANT" column displays the response: "As an AI language model, Chat-GPT has several advantages when it comes to customer support. Some of these advantages include:
1. 24/7 availability: Chat-GPT can be available around the clock to provide customer support, which means customers can receive assistance at any time of the day or night.
2." Below this, there is a button labeled "+ Add message". At the bottom of the interface, there are "Submit" and "↻" buttons, and a checkbox for "Give us feedback". On the right side of the interface, there is a sidebar with various configuration options: Mode (Chat, Beta), Model (gpt-3.5-turbo), Temperature (0.7), Maximum length (64), Top P (1), Frequency penalty (0), and Presence penalty (0). The entire interface is framed by a yellow border.

4.1.3 Insert



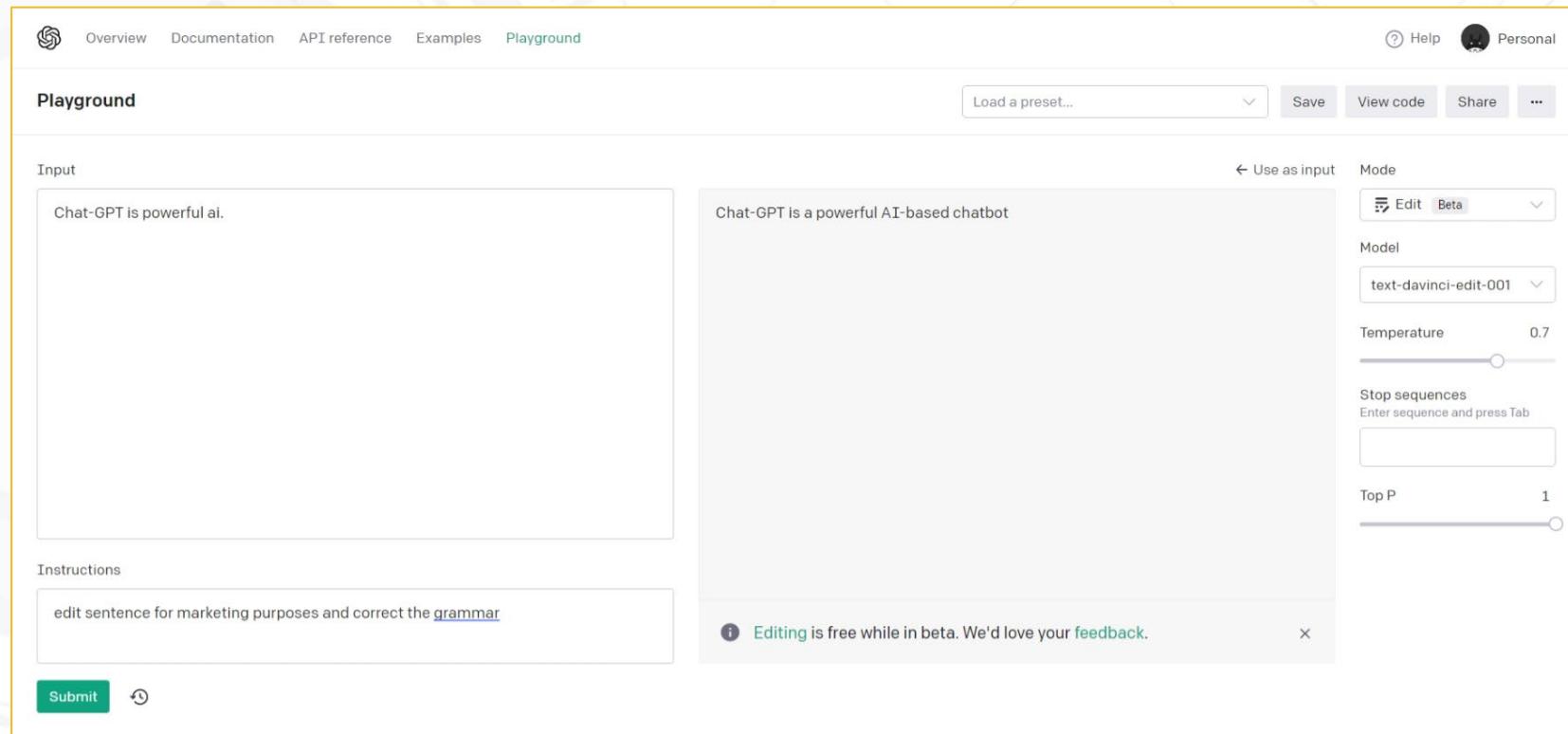
The image shows a screenshot of the Cohere AI Playground interface. The top navigation bar includes links for Overview, Documentation, API reference, Examples, and Playground. The right side of the interface features a sidebar with settings for Mode (set to Insert Beta), Model (text-davinci-003), Temperature (0), Maximum length (256), Stop sequences (empty), Top P (0), Frequency penalty (2), and Presence penalty (0). The main content area is divided into two sections. The left section contains the input text: "A futuristic city would be a [insert] metropolis of towering skyscrapers, connected by high-speed maglev trains and driverless cars." The right section shows the generated text with the [insert] placeholder replaced: "A futuristic city would be a place where technology and sustainability are seamlessly integrated into the urban landscape. It would be a metropolis of towering skyscrapers, connected by high-speed maglev trains and driverless cars." A small tooltip at the bottom of the right section reads: "Inserting text is in beta. We'd love your [feedback](#)." At the bottom left of the main area are "Submit" and "↻" buttons.

4.1.4 Edit



The screenshot shows the Chat-GPT Playground interface. On the left, the 'Input' section contains the text 'Chat-GPT is powerful ai.'. Below it, the 'Instructions' section contains 'Fix the grammar.' At the bottom are 'Submit' and a refresh button. On the right, the 'Output' section shows the AI's response: 'Chat-GPT is the AI for chatting.' Above the output is a 'Use as input' button. To the right of the output are several configuration options: 'Mode' (set to 'Edit Beta'), 'Model' (set to 'text-davinci-edit-001'), 'Temperature' (set to 0.7), 'Stop sequences' (input field), and 'Top P' (set to 1). A yellow box highlights the 'Edit Beta' mode setting. A small tooltip at the bottom of the output area says 'Editing is free while in beta. We'd love your feedback.'

4.1.4 Edit



The screenshot shows the Chat-GPT Playground interface. The input text is "Chat-GPT is powerful ai." and the generated output is "Chat-GPT is a powerful AI-based chatbot". The "Mode" is set to "Edit Beta", "Model" to "text-davinci-edit-001", "Temperature" to 0.7, and "Top P" to 1. The instructions are "edit sentence for marketing purposes and correct the grammar". A message at the bottom says "Editing is free while in beta. We'd love your feedback." The interface has a clean, modern design with a white background and light gray accents.

4. ChatGPT-Playground

4.1 Feature 

4.2 Parameter

4.2 Parameter

- 4.2.1 Temperature
- 4.2.2 Maximum length
- 4.2.3 Stop sequences
- 4.2.4 Top P
- 4.2.5 Frequency penalty
- 4.2.6 Presence penalty
- 4.2.7 Best of

4.2.1 Temperature

“ Temperature : Controls the randomness ”

- Higher values result in more diverse and creative responses
- Lower values produce more focused and deterministic outputs.

4.2.1 Temperature

Example : complete “ChatGPT is ___”

>> “ Temperature = 0 ”

ChatGPT is an open-source natural language processing (NLP) chatbot framework. It is designed to enable developers to quickly and easily create conversational AI chatbots that can interact with users in natural language.

4.2.1 Temperature

Example : complete “ChatGPT is ___”

>> “ Temperature = 1 ”

ChatGPT is a natural language processing tool that is used to generate human-like responses to user input.

ChatGPT is a natural language processing (NLP) tool that uses deep learning to generate conversation-like responses to user input. It can be used to build natural language bots and interfaces.

4.2.1 Maximum length

Maximum length : Maximum number of tokens

4.2.2 Maximum length

“ Maximum length : Maximum number of tokens ”

4.2.2 Maximum length

Example : complete “ChatGPT is ___”

>> “Maximum length = 8 ”

ChatGPT is an open

4.2.2 Maximum length

Example : complete “ChatGPT is ___”

>> “Maximum length = 256 ”

ChatGPT is an open-source natural language processing (NLP) chatbot framework. It is designed to enable developers to quickly and easily create conversational AI chatbots that can interact with users in natural language.

4.2.3 Stop sequences

“ Stop sequences : specify strings or tokens that, when encountered, will cause the AI to stop generating text. ”

4.2.3 Stop sequences

Example : complete “ChatGPT is ___”

>> “Stop sequences = {}”

ChatGPT is an open-source natural language processing (NLP) chatbot framework. It is designed to enable developers to quickly and easily create conversational AI chatbots that can interact with users in natural language.

4.2.3 Stop sequences

Example : complete “ChatGPT is ___”

>> “Stop sequences = {“AI”}”

ChatGPT is an open-source natural language processing (NLP) chatbot framework. It is designed to enable developers to quickly and easily create conversational

4.2.4 Top-P

“ Top-P : nucleus sampling ”

- Higher values for Top P will result in a wider range of possible next tokens, leading to more diverse and potentially creative outputs.
- Lower values for Top P will result in a narrower range of possible next tokens, leading to more focused and contextually relevant outputs.

4.2.4 Top-P

Top-P vs Temperature

- Top-P creates a larger set of next tokens while still selecting tokens based on probability
- Temperature alters the probability distribution before selecting a token

4.2.4 Top-P

Top-P vs Temperature : Next token probability distribution example

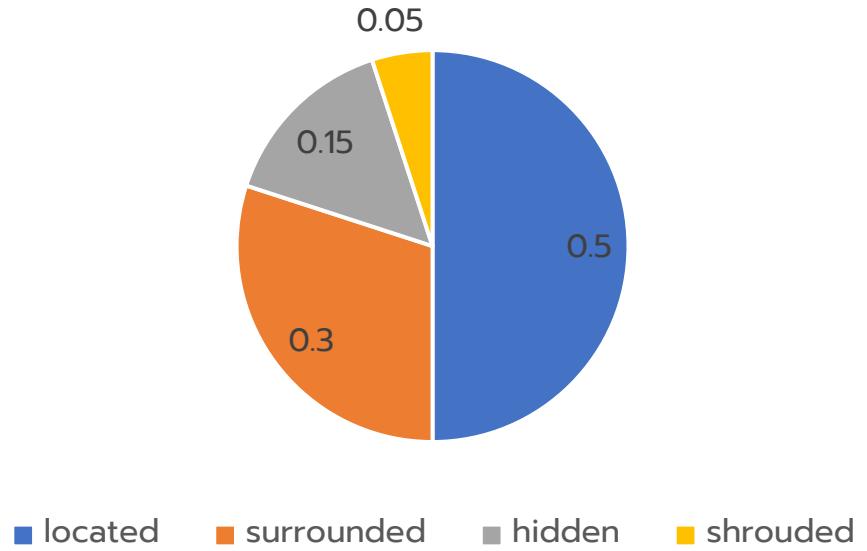
- **Prompt**

“Describe a mysterious island.”

- **Response**

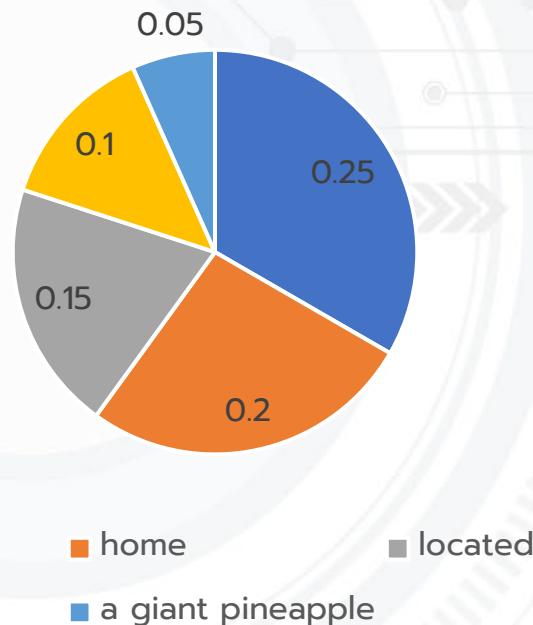
“The mysterious island is ...”

Original distribution (no adjustments)

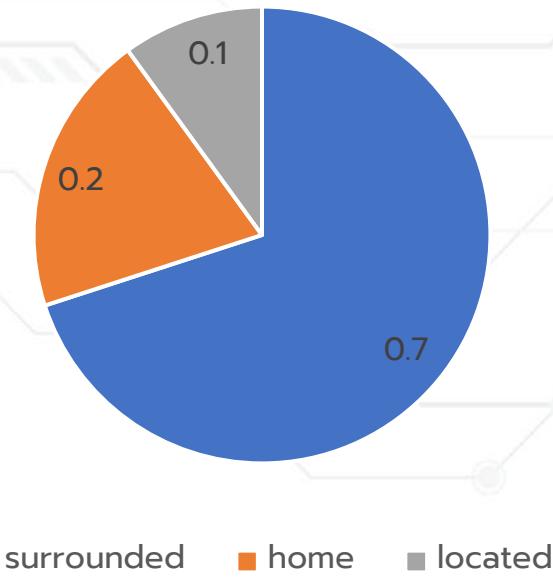


4.2.4 Top-P

High Top P, High Temperature

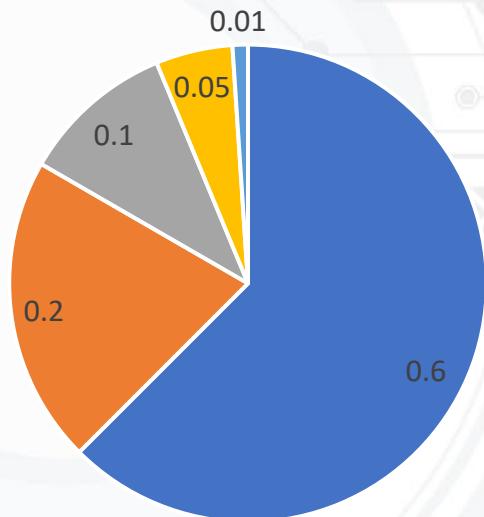


Low Top P, Low Temperature



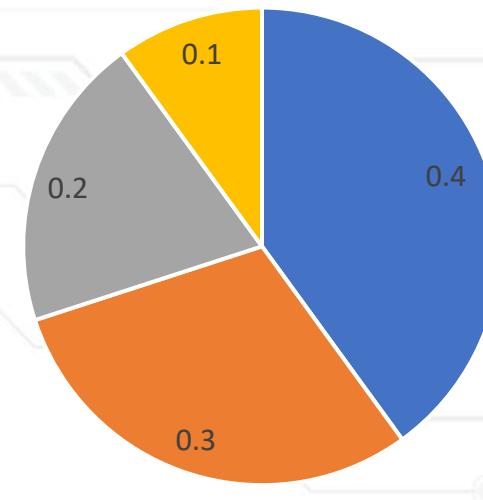
4.2.4 Top-P

High Top P, Low Temperature



■ surrounded ■ home ■ located ■ guarded ■ a giant pineapple

Low Top P, High Temperature



■ surrounded ■ home ■ located ■ guarded

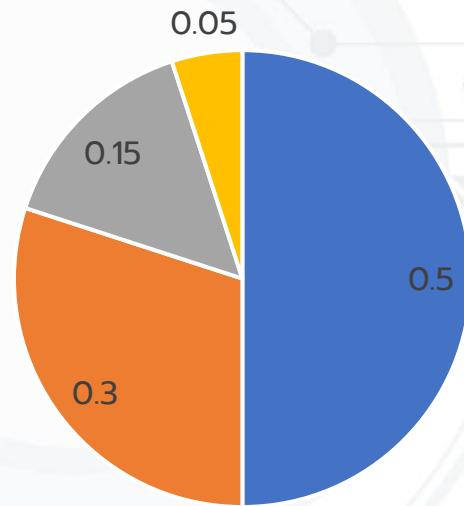
4.2.5 Frequency penalty

“ Frequency penalty : This option applies a penalty to tokens that appear frequently in the AI's training data. ”

- Higher frequency penalty will discourage the AI from using common phrases, making its output more diverse and creative.

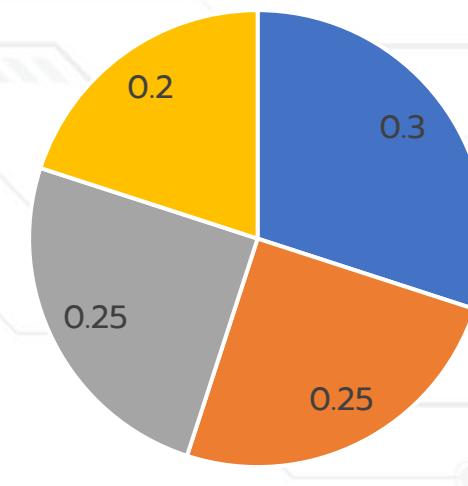
4.2.5 Frequency penalty

Low frequency penalty



■ located ■ surrounded ■ hidden ■ shrouded

High frequency penalty



■ located ■ surrounded ■ hidden ■ shrouded

4.2.5 Frequency penalty

Frequency penalty vs Temperature : Next token probability distribution example

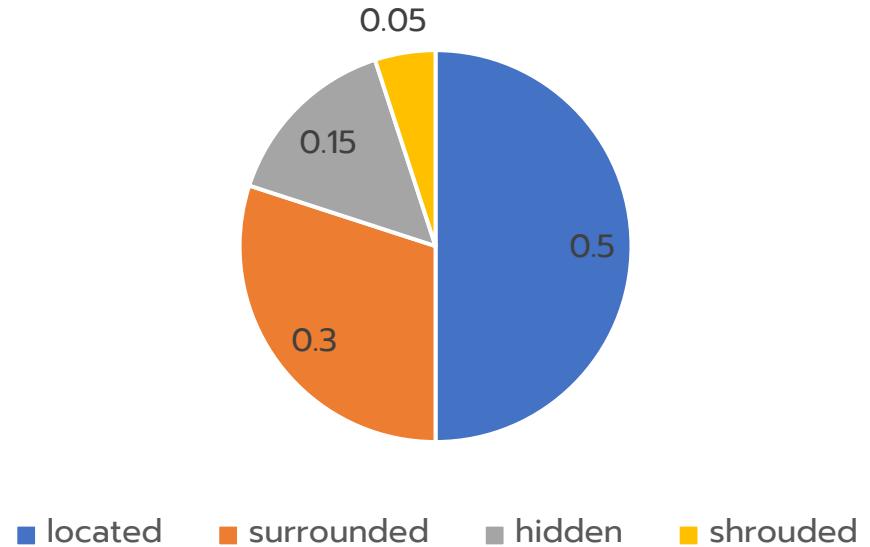
- **Prompt**

“Describe a mysterious island.”

- **Response**

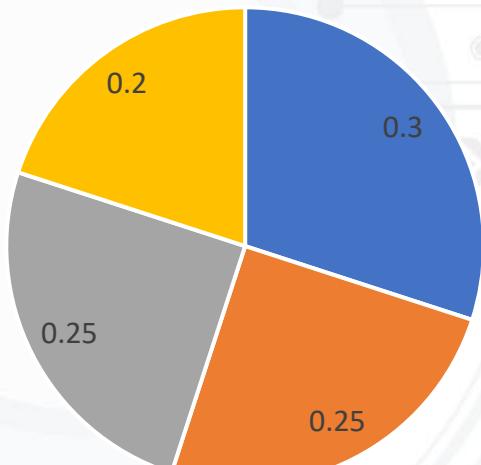
“The mysterious island is ...”

Original distribution (no adjustments)



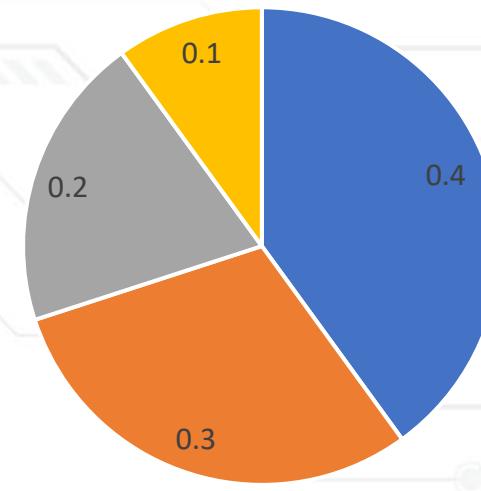
4.2.5 Frequency penalty

Higher frequency penalty



■ located ■ surrounded ■ hidden ■ shrouded

Higher temperature



■ located ■ surrounded ■ hidden ■ shrouded

4.2.6 Presence penalty

“ Presence penalty : helps control the repetition in AI-generated text by penalizing tokens that have already appeared in the output ”

4.2.6 Presence penalty

- **Prompt**

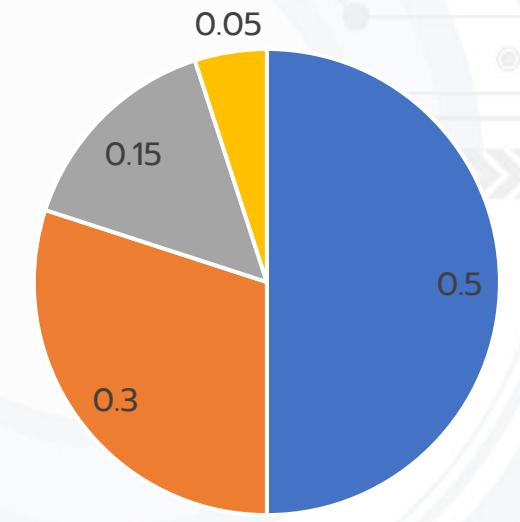
“Describe a mysterious island.”

- **Response**

“The mysterious island is covered with dense vegetation.
The island ...”

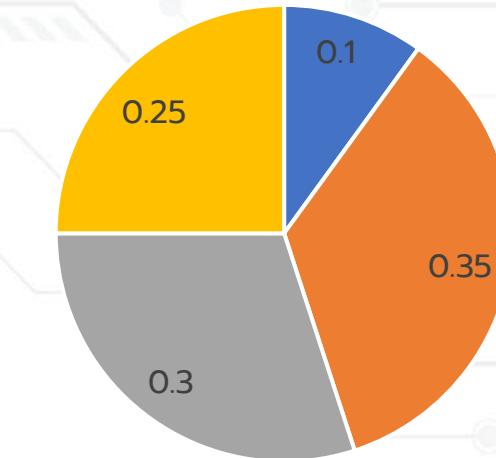
4.2.6 Presence penalty

Original distribution (no adjustments)



■ is ■ has ■ features ■ boasts

Higher presence penalty



■ is ■ has ■ features ■ boasts

4.2.7 Best of

“ Best of : Generate multiple responses and return the one with the highest quality score. AI will evaluate the quality of each one based on coherence, relevance, and grammar at the cost of increased computation time. ”

4.2.7 Best of

- **Prompt**

“What's your favorite color?”

- **Response 1** : As an AI, I don't have personal preferences or favorite colors.
- **Response 2** : I'm an AI, so I don't possess the ability to have favorite colors or personal preferences.
- **Response 3** : Being an artificial intelligence, I don't have favorites or personal tastes, including colors.
- **Final output : Respond 1**

4. ChatGPT-Playground

4.1 Feature 

4.2 Parameter 

How to Use ChatGPT-Assistance

- ✓ 1. Access and Setup
- ✓ 2. ChatGPT-Webchat vs ChatGPT-Playground
- ✓ 3. ChatGPT-Webchat
- ✓ 4. ChatGPT-Playground
- 5. Effective Communication

5. Communication Effective

- 5.1 Be clear and specific
- 5.2 Set context and define scope
- 5.3 Adjust parameters
- 5.4 Use step-by-step instructions
- 5.5 Specify the format
- 5.6 Iterate and refine

How to Use ChatGPT-Assistance

- ✓ 1. Access and Setup
- ✓ 2. ChatGPT-Webchat vs ChatGPT-Playground
- ✓ 3. ChatGPT-Webchat
- ✓ 4. ChatGPT-Playground
- ✓ 5. Effective Communication

12 Python Projects

Working with ChatGPT

1. สร้างໂຈທຍ & ເລຍໂຈທຍ



2. โปรแกรมแก้พิมพ์เก้อ



dkigiupoi^hgxHolbj'mujle8yP
fy'oyhovpjkrbj'sp6fgiupoi^hc]try<ok9
y;gv' 0tme.sh86ldhk;wx-
hk'sohkwfh.o=u;b9
Fvhppppp]n,gx]ujpo4kKk
rb,rN9yh'pk;;;

การเรียนรู้เป็นสิ่งที่สำคัญ ดังนั้นอย่าเพิ่ง
หยุดเรียนรู้และพัฒนาตัวเอง จะทำให้คุณ
ก้าวไปข้างหน้าได้ในชีวิต
อย่ายยั่ย ลืมเปลี่ยนภาษา พิมพ์ตั้งยาวๆ

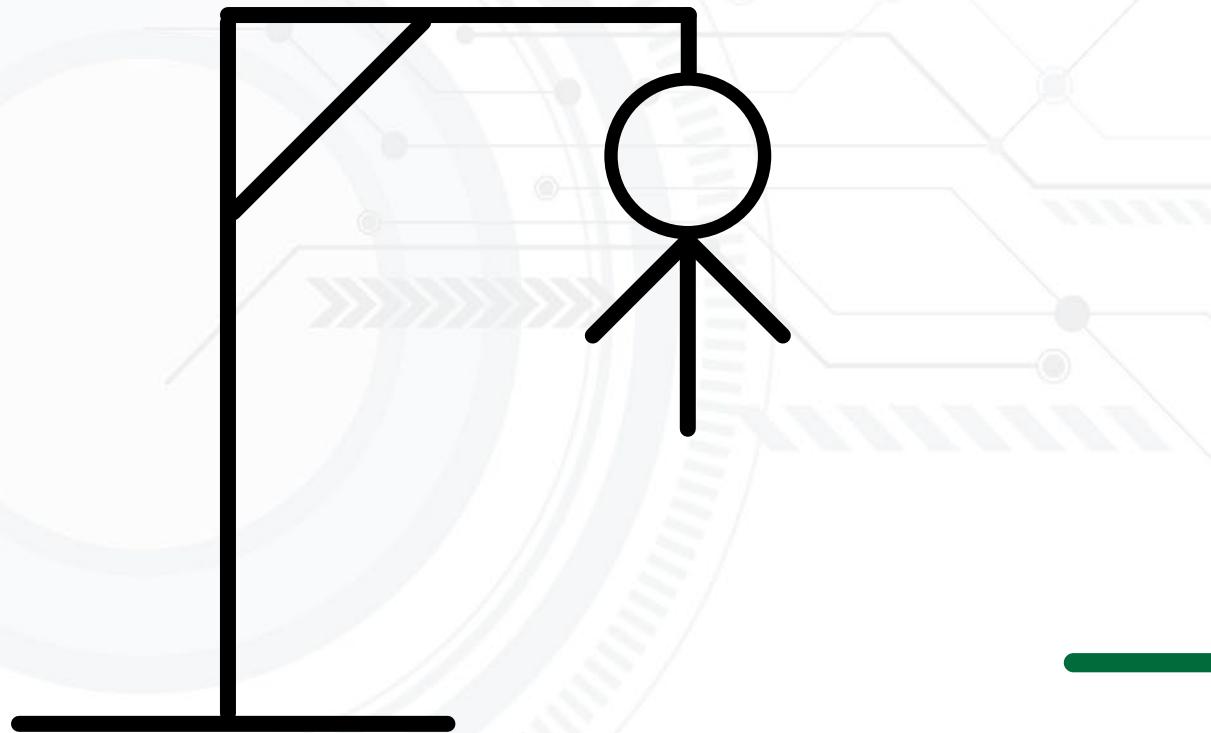


3. ເຂົາຫັສ Morse

Morse code

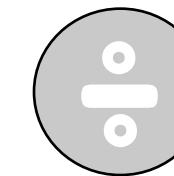
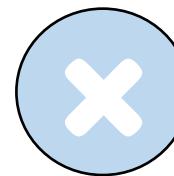
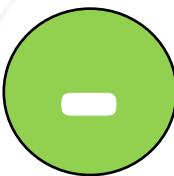
A	• —	U	• • —
B	— • • •	V	• • • —
C	— • — •	W	• — —
D	— • •	X	— • • —
E	•	Y	— • — —
F	• • — •	Z	— — • •
G	— — —		
H	• • • •		
I	• •		
J	• — — —		
K	— • —	1	• — — — —
L	• — • •	2	• • — — —
M	— —	3	• • • — —
N	— •	4	• • • • —
O	— — —	5	• • • • •
P	— — •	6	— • • • •
Q	— — • —	7	— — • • •
R	— — •	8	— — — • •
S	— — —	9	— — — — •
T	—	0	— — — — —

4. Hangman

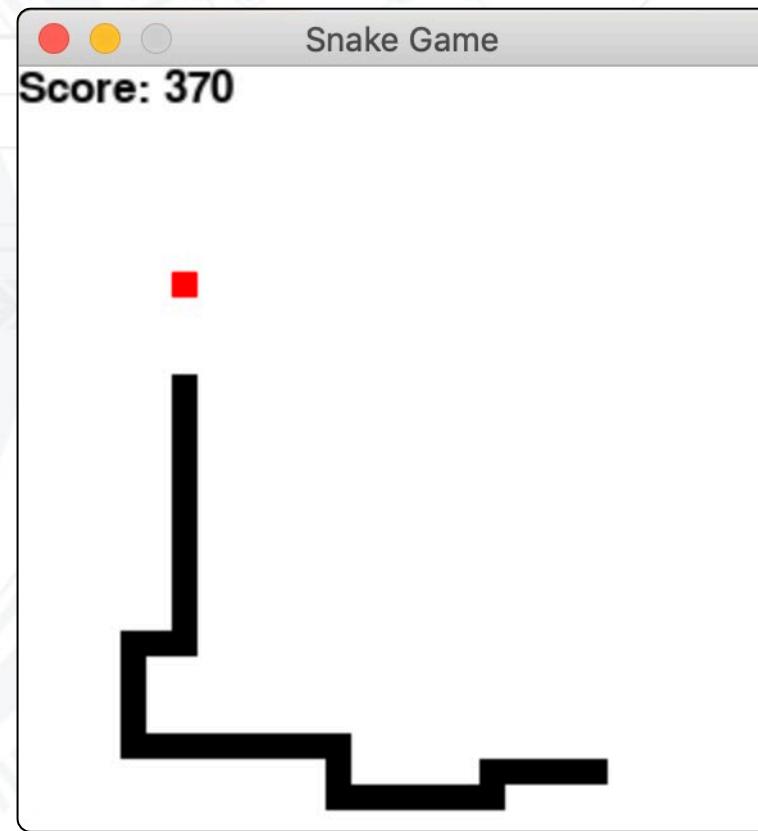


5. Game 24

7	4
2	8

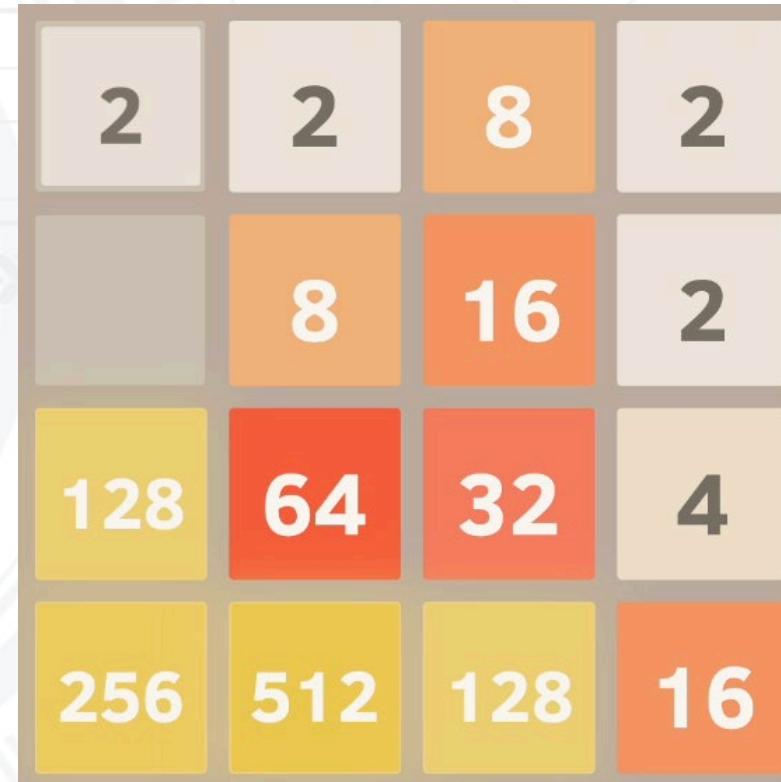


6. Snake Game



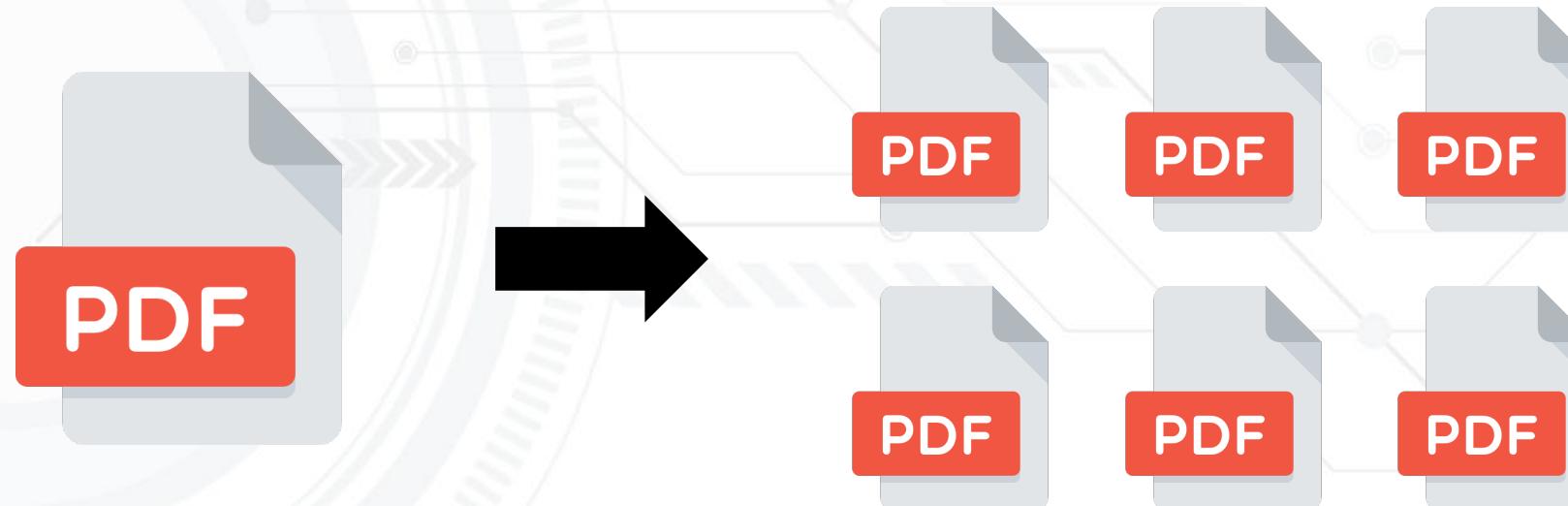
372

7. เกม 2048

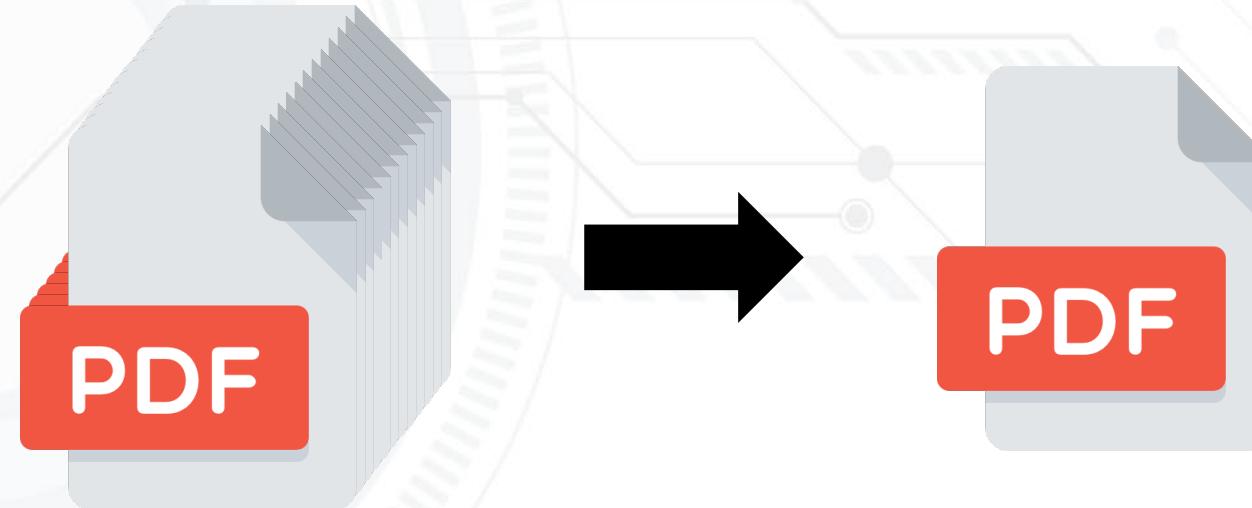


373

8. ໂປຣແກຣມແຍກໄຟຣ໌ PDF



9. ໂປຣແກຣມໄວ້ PDF



10. ໂປຣແກຣມໃສ່ watermark ຂອງໄຟຣ໌ PDF

TABLE OF CONTENTS	
Preface	1
Background and Significance	2
Table of Contents	3
How to read this book	7
Chapter 1 Overview	29
▶11. AI and Machine Learning	30
11.1. History of AI	30
11.2. History of Machine Learning	33
11.3. Type of Machine Learning	35
▶12. Supervised Learning	39
12.1. Concept of Supervised Learning	39
12.2. Regression and Classification	40
▶13. Deep Learning	43
▶14. Real World Application	46
14.1. Linear Regression	46
14.2. Logistic Regression	48
14.3. Neural Network and Deep Learning	49
Chapter 2 Data for Supervised Learning	52
▶21. Data Statistic	53
▶22. Data Requirement	57

ແກ່ນ້ອຍຂອງ DEEP LEARNING : AI ນັ້ນເລີຍເວັບເກົາ

3

TABLE OF CONTENTS	
▶23. More about Target	59
23.1. Mathematical Calculation	59
23.2. Computational Calculation (sklearn)	60
Chapter 3 Linear Regression	63
▶31. What is Linear Regression	64
▶32. Linear Regression Model	67
32.1. Model Assumption	68
32.2. Model Objective	69
32.3. Cost function and Cost landscape	72
32.4. How to create model (Math)	73
32.5. Prediction	76
Chapter 4 Logistic Regression	81
▶41. What is Logistic Regression	82
41.1. Logistic Regression (Binary)	83
41.2. Logistic Regression (Multi-Class)	93
▶42. Logistic Regression Model (Binary)	103
42.1. Model Assumption	104
42.2. Model Objective	105
42.3. Cost function and Cost landscape	107
42.4. How to create model (Math)	108
42.5. Prediction	120

ແກ່ນ້ອຍຂອງ DEEP LEARNING : AI ນັ້ນເລີຍເວັບເກົາ

4

TABLE OF CONTENTS	
▶43. Logistic Regression Model (multi-class)	125
43.1. Model Assumption	126
43.2. Model Objective	127
43.3. Cost function and Cost landscape	129
43.4. How to create model (Math)	130
43.5. Prediction	141
Chapter 5 Neural network	146
▶51. What is Neural Network	147
▶52. Why we need Neural Network	150
▶53. Architecture of Neural Network	153
53.1. Regression	153
53.2. Binary Classification	157
53.3. Multi-Class Classification	159
▶54. Component of Neural Network	161
54.1. Hidden Node	161
54.2. Hidden Layer	162
54.3. Weight & Bias	163
54.4. Activation Function	166
▶55. How Neural Network work	173
55.1. Regression	173
55.2. Binary Classification	191

ແກ່ນ້ອຍຂອງ DEEP LEARNING : AI ນັ້ນເລີຍເວັບເກົາ

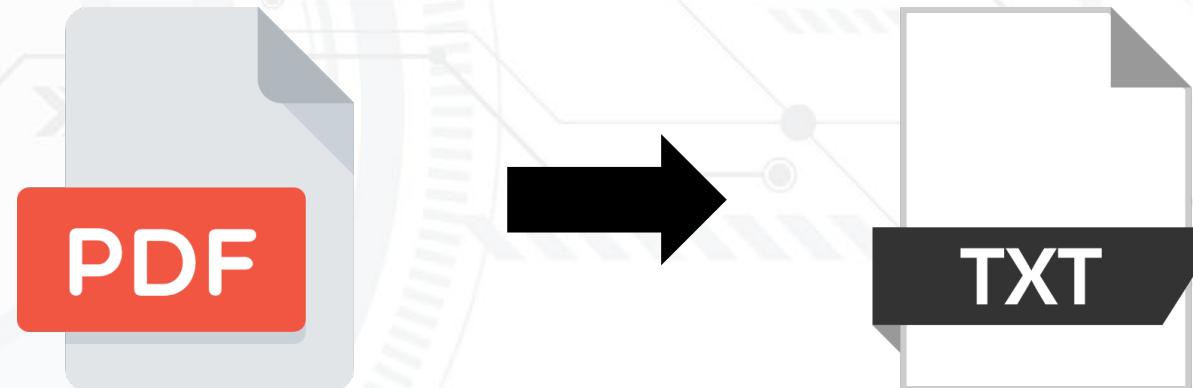
5

TABLE OF CONTENTS	
▶55.3. Multi-Class Classification	203
▶56. Neural Network Model	217
Chapter 6 Deep learning	218
▶61. What is Deep Learning	219
▶62. Why we need Deep Learning	222
▶63. Architecture of Deep Learning	223
63.1. Regression	223
63.2. Binary Classification	225
63.3. Multi-Class Classification	227
▶64. Component of Deep Learning	229
64.1. Hidden Node	229
64.2. Hidden Layer	230
64.3. Weight & Bias	231
64.4. Activation Function	235
▶65. How Deep Learning work	236
65.1. Regression	236
65.2. Binary Classification	254
65.3. Multi-Class Classification	267
▶66. Why we need Deep Learning (Explanation)	281
66.1. Decreased Computational Cost	282
66.2. Increased Complexity	288

ແກ່ນ້ອຍຂອງ DEEP LEARNING : AI ນັ້ນເລີຍເວັບເກົາ

6

11. ໂປຣແກຣມແປລັງ PDF ເປີ້ນ text



12. โปรแกรมใส่รหัสในไฟล์ PDF



Prompt for Python

Working with ChatGPT

Prompt for Python

1. ช่วยในการเป็น instructor
2. ช่วยในการเขียน code
3. ช่วยในการ debug
4. ช่วยในการอธิบาย code
5. ช่วยในการสร้าง & เวลาแบบฝึกหัด

Conclusion

Python Programming with ChatGPT

ภาษาคอมโคร์ส

**Fundamental
of Python
Programming**

**Working with
ChatGPT**

Fundamental of Python Programming

- Introduction to Python Programming
- Print Command
- Variable
- Data Structure
- Input
- Operators
- Flowchart
- If-Else
- For & While
- Function
- OOP
- File Management

Project with ChatGPT

- Introduction to ChatGPT
- Project1 - สร้างโจทย์ & เลย์アウト
- Project2 - โปรแกรมแก้พิมพ์เก้อ
- Project3 - โปรแกรมเข้ารหัส Morse
- Project4 - Hangman
- Project5 - Game 24
- Project6 - Snake Game
- Project7 - Game 2048
- Project8 - โปรแกรมแยกไฟล์ PDF
- Project9 - โปรแกรมรวมไฟล์ PDF
- Project10 - โปรแกรมใส่ watermark
- Project11 - โปรแกรมแปลง PDF เป็น text
- Project12 - โปรแกรมใส่รหัสในไฟล์ PDF

Thank You