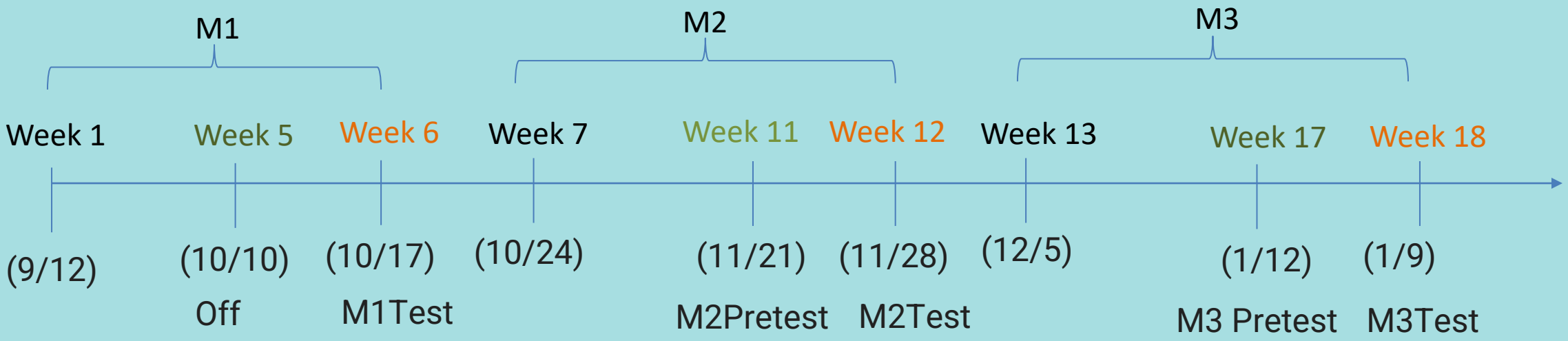
Fundamental Programming Course

Week 3

Huseh-Ting Chu@Asia University, 2021

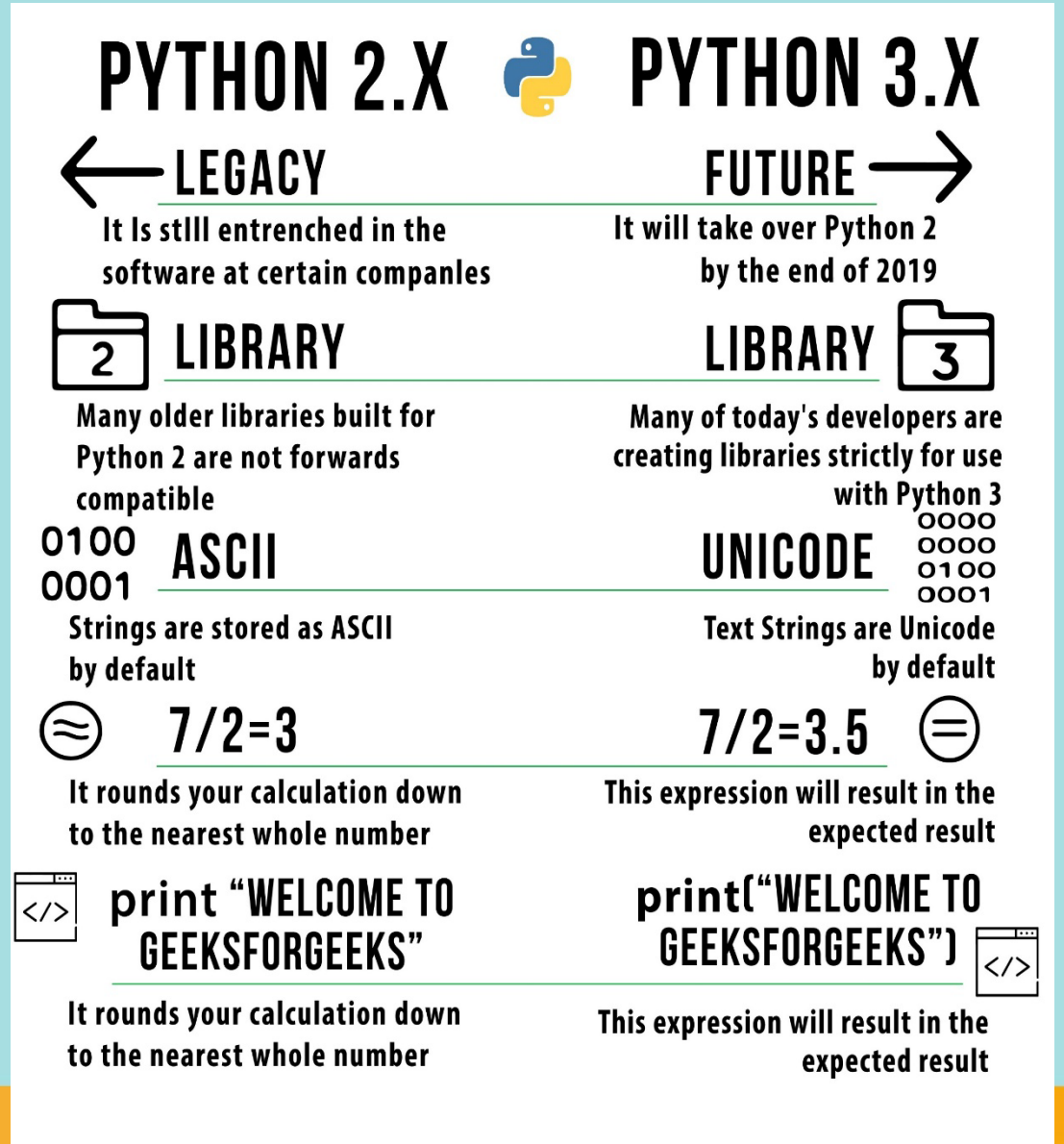
Schedule


M1
M2
M3
Week 1
Week 5
Week 6
Week 7
Week 11
Week 12
Week 13
Week 17
Week 18
(9/12)
(10/10)
Off
(10/17)
M1Test
(10/24)
(11/21)
M2Pretest
(11/28)
M2Test
(12/5)
(1/12)
M3 Pretest
(1/9)
M3Test

Syllabus

- W1-Python Introduction and Programming Tools
- W2-Variables and Operations
- W3-Loop and formatted output
- W4-Condition and Containers
- W5-String and built-in functions
- W6-M1 test
- W07-Dictionary Container
- W08-File I/O
- W09-Function
- W10-Advanced flow control
- W11-Advanced operations and generators
- W12-M2 test
- W13-Advanced functions
- W14-Class fundamentals (classes, objects, properties, constructors, methods)
- W15-Advanced Classes (Static methods, class Methods and class decorators)
- W16-Modules and Packages
- W17-Advanced programming(Argparse and Venv)
- W18-M3 test

Python Versions



Python new features:

- Python 3.10: Structural Pattern Matching
- Python 3.6 : f-Strings
- Python 3.3 : Virtual Environments
- Python 3.2: Argparse

Python powerful features:

- Iterators
- Generators
- Decorators
- Context Managers

Python cheat sheet

Beginner's Python Cheat Sheet

Variables and Strings

Variables are used to store values. A string is a series of characters, surrounded by single or double quotes.

Hello world

```
print("Hello world!")
```

Hello world with a variable

```
msg = "Hello world!"
print(msg)
```

Concatenation (combining strings)

```
first_name = 'albert'
last_name = 'einstein'
full_name = first_name + ' ' + last_name
print(full_name)
```

Lists

A list stores a series of items in a particular order. You access items using an index, or within a loop.

Make a list

```
bikes = ['trek', 'redline', 'giant']
```

Get the first item in a list

```
first_bike = bikes[0]
```

Get the last item in a list

```
last_bike = bikes[-1]
```

Looping through a list

```
for bike in bikes:
    print(bike)
```

Adding items to a list

```
bikes = []
bikes.append('trek')
bikes.append('redline')
bikes.append('giant')
```

Making numerical lists

```
squares = []
for x in range(1, 11):
    squares.append(x**2)
```

Lists (cont.)

List comprehensions

```
squares = [x**2 for x in range(1, 11)]
```

Slicing a list

```
finishers = ['sam', 'bob', 'ada', 'bea']
first_two = finishers[:2]
```

Copying a list

```
copy_of_bikes = bikes[:]
```

Tuples

Tuples are similar to lists, but the items in a tuple can't be modified.

Making a tuple

```
dimensions = (1920, 1080)
```

If statements

If statements are used to test for particular conditions and respond appropriately.

Conditional tests

```
equals            x == 42
not equal         x != 42
greater than      x > 42
  or equal to     x >= 42
less than         x < 42
  or equal to     x <= 42
```

Conditional test with lists

```
'trek' in bikes
'surly' not in bikes
```

Assigning boolean values

```
game_active = True
can_edit = False
```

A simple if test

```
if age >= 18:
    print("You can vote!")
```

If-elif-else statements

```
if age < 4:
    ticket_price = 0
elif age < 18:
    ticket_price = 10
else:
    ticket_price = 15
```

Dictionaries

Dictionaries store connections between pieces of information. Each item in a dictionary is a key-value pair.

A simple dictionary

```
alien = {'color': 'green', 'points': 5}
```

Accessing a value

```
print("The alien's color is " + alien['color'])
```

Adding a new key-value pair

```
alien['x_position'] = 0
```

Looping through all key-value pairs

```
fav_numbers = {'eric': 17, 'ever': 4}
for name, number in fav_numbers.items():
    print(name + ' loves ' + str(number))
```

Looping through all keys

```
fav_numbers = {'eric': 17, 'ever': 4}
for name in fav_numbers.keys():
    print(name + ' loves a number')
```

Looping through all the values

```
fav_numbers = {'eric': 17, 'ever': 4}
for number in fav_numbers.values():
    print(str(number) + ' is a favorite')
```

User input

Your programs can prompt the user for input. All input is stored as a string.

Prompting for a value

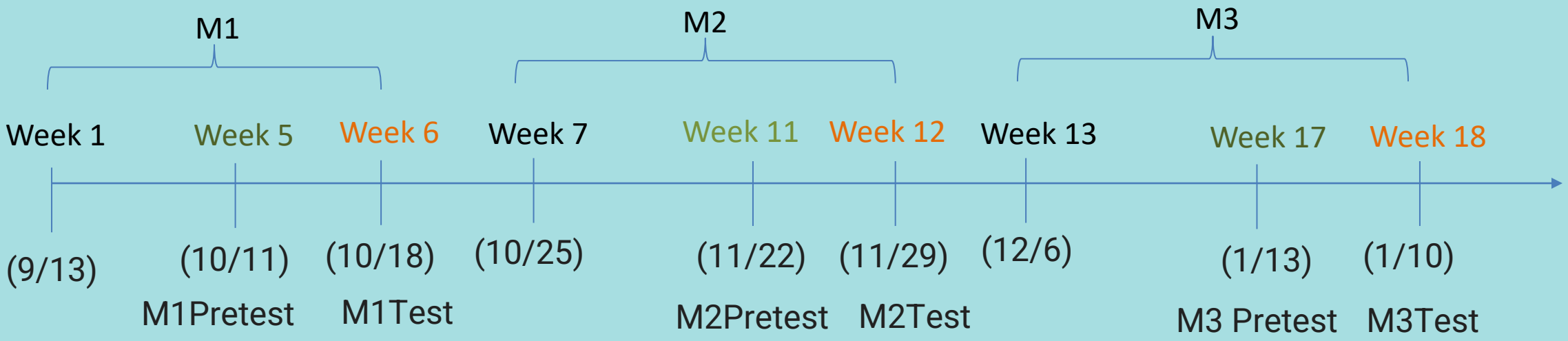
```
name = input("What's your name? ")
print("Hello, " + name + "!")
```

Prompting for numerical input

```
age = input("How old are you? ")
age = int(age)

pi = input("What's the value of pi? ")
pi = float(pi)
```

Schedule


M1
M2
M3
Week 1
Week 5
Week 6
Week 7
Week 11
Week 12
Week 13
Week 17
Week 18
(9/13)
(10/11)
(10/18)
(10/25)
(11/22)
(11/29)
(12/6)
(1/13)
(1/10)
M1Pretest
M1Test
M2Pretest
M2Test
M3 Pretest
M3Test

This Week's Content – Loop and Formatting Output

- Essential - basic
  - Input: sequence (List), string split (str.split()), len() function
  - Process: for-loop
  - Process: multiple assignment
  - Output: f-format string, formatted output of print()
    - print 99 multiplication table
    - range() function
- Advanced-Advanced
  - Legacy string formatting (1) % string formatting (2) str.format() function
  - Sequence (List) initialization and insertion append() function

Kissipo Learning

Kissipo = KISS principle + IPO model

KISS principle

"keep it simple, stupid" or "keep it stupid simple", is a design principle noted by the U.S. Navy in 1960.
https://en.wikipedia.org/wiki/KISS_principle

IPO model

The input–process–output (IPO) model is a widely used approach in systems analysis and software engineering for describing the structure of an information processing program or other process.
https://en.wikipedia.org/wiki/IPO_model

Kissipo Learning for Programming with Python(PWP)

Courseware: Notebook+ Github

(1) Teaching with Notebook (Google Colab).
(2) Use Github to build lesson plans

Keep:

Variables and assignment
operator and expression
left-hand side and right-hand side
unpacking

S&S:

help(), type(), len(), size()

IPO-I: input

input()
int(), float(), str()
split(), map()

IPO-P: Process

for-loop/while-loop
if, elif, else
range()

IPO-O: output

print()
open(), write()

IPO Model(1)

input() help(), type() print()

The basic idea of this chapter is that students should know:
Input with input(), output with print()
help() can view the description of the function or category
type() can check the type of the variable

IPO Model (2)

input() input a variable
Convert to integer variable using int()
Convert to float variable using float()

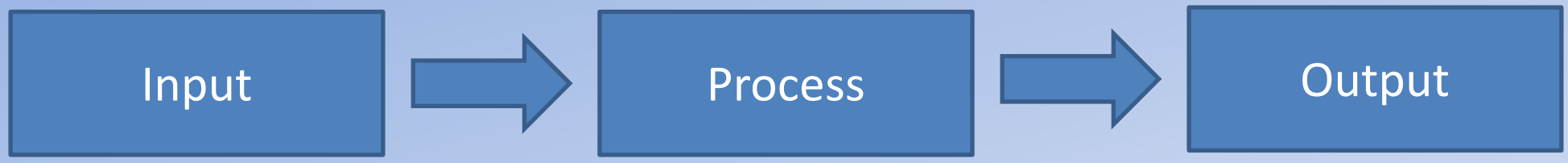
Arithmetic operator
Operator precedence
String representation in program
Comment

The print() function
Escape Sequence

The basic idea of this chapter is that students should know:

- How to use input() to input numbers of different types
- Output print() has two parameters sep and end to control the output
- Arithmetic operations in Python include: addition, subtraction, multiplication and division (+ - * /), power (**), quotient (//) and remainder (%).
- The result of addition, subtraction, multiplication and division (+ - * /), power (**) is a floating point number. The quotient (//) and remainder (%) calculations result in integers.

IPO Model (3)



input() input multiple variables
Convert to a list using split()

for-loop,
Multiple assignment
Indexing and Slicing
Use of the range() function
Initialization of a list (List)

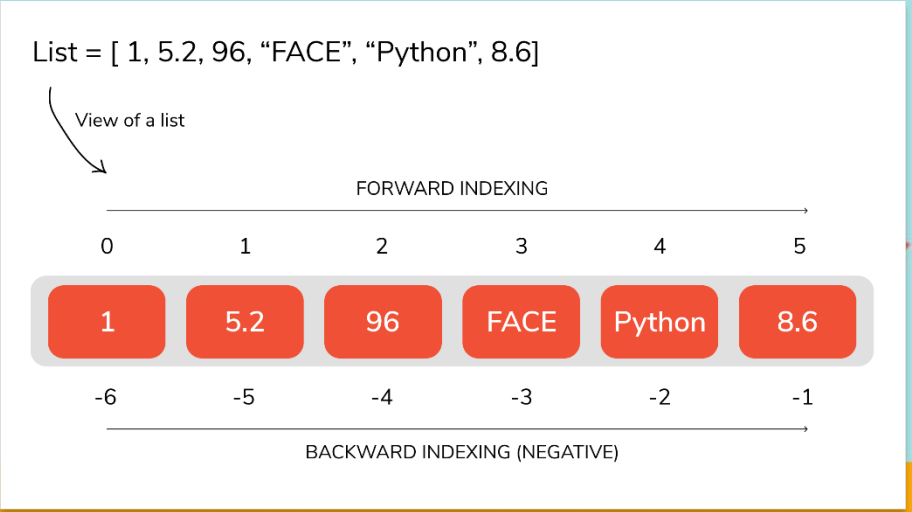
Formatted output of print()
f - string format

The basic idea of this chapter is that students should know:

- How to enter multiple variables with input()
- formatted output with f-format string
- Use of loops.
- What is Indexing and Slicing

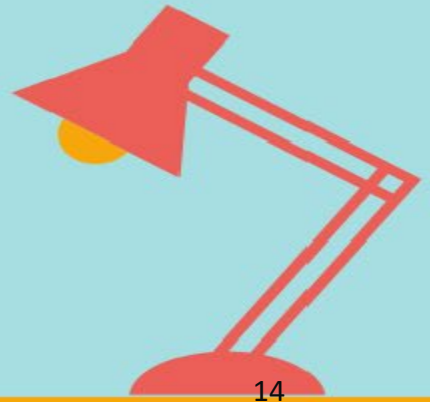
Topic 1-list

- Serial put a bunch of variables into [] to form a row of data
- A serial variable is a collection of a bunch of variables
- Step 1: A bunch of variables without using list variables
- Step 2: Use the list variable as a set variable, and use the index key to get the variable/indexing of list elements
- Step 3: Negative index keys get variables
- Step 4: Length of list/Length of a list
- Step 5: Add the elements of the list (list)



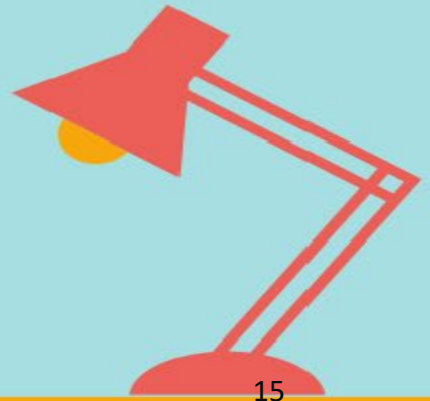
Topic 2-str.split()

- Step 1: Substring separated by whitespace
- Step 2: Use split() to split the substring
- Step 3: Split the input string with split()
- Step 4: Split the input string with split() and convert it to an integer



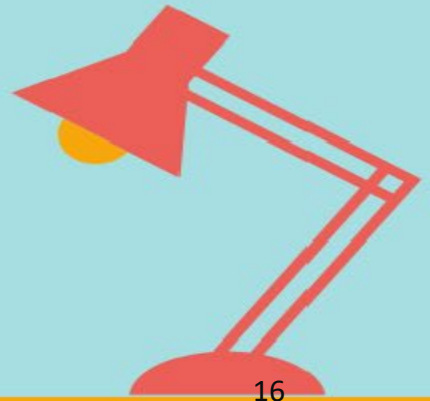
Topic 3-for-loop

- Step 1: Use a loop on the list variable
- Step 2: Using the loop, print out the numbers 1-9/print out the numbers 1-9



Topic 4: multiple assignment

- Step 1: assignment individually
- Step 2: Multiple assignment, also known as unpacking
- Step 3: Partial unpacking



Topic 5: f-string/formatted output

- Step 1: f-string
- Step 2: formatted output
- Step 3: formatting of numbers

f-string格式化字串

A = 435; B = 59.058

```
print(f"Art:{A:5d}, Price per Unit: {B:8.2f}")
```

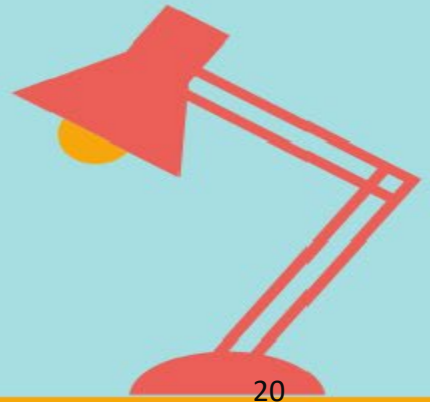
說明：
{A:5d} #將變數A以寬度5格印成整數字串
{B:8.2f} #將變數B以寬度8格印成成小數2位的字串

Topic 6: print out the 99 multiplication table

- Step 1: Use 1 layer loop, print out 1-9
- Step 2: Use 2 layers of loops to print out the 99 multiplication table

Topic 7-range()

- Step 1: parameters of range(): begin/end
- Step 2: parameters of range(): step
- Step 3: Using range(), print out the 99 multiplication table



Topic 8- old-style string formatting

- Step 1: %-formatting
- Step 2: str-format（Python 2.6+）
- Step 3: f-string（Python 3.6+）

formatted print() function

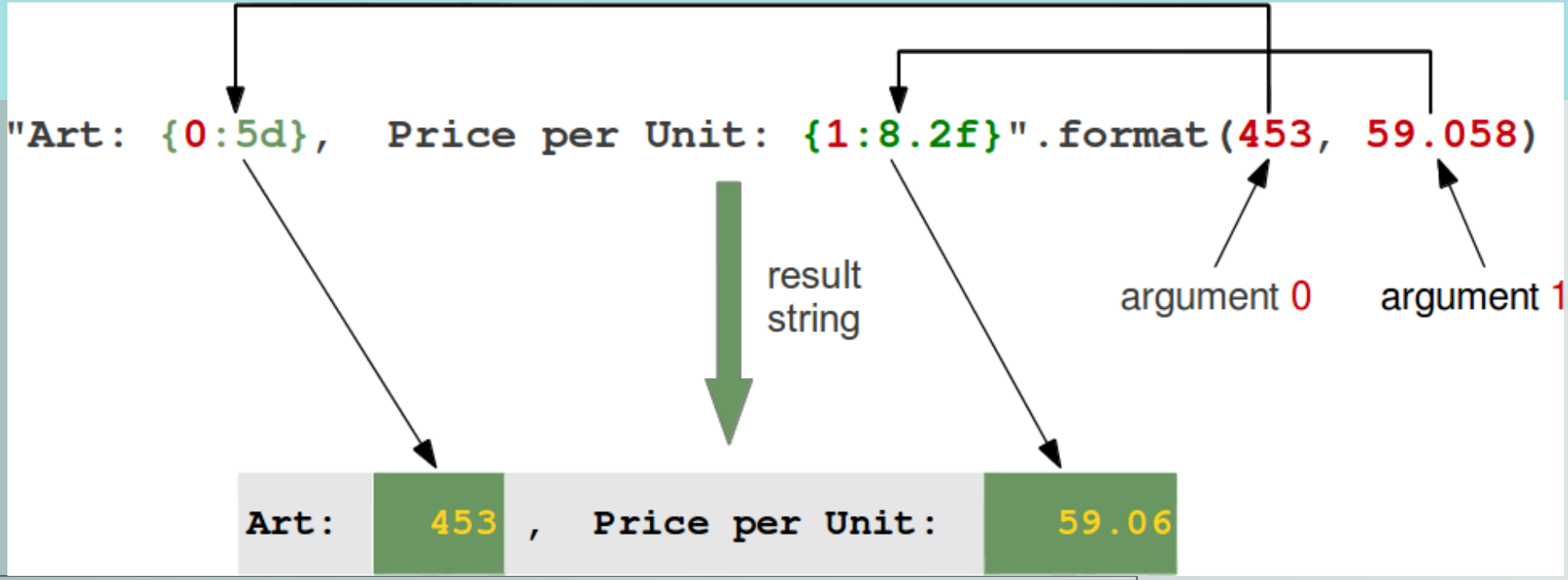
```
illustrate:
%-formatting formatted printing
print('Art: %5d, Price per Unit: %8.2f' % (435, 59.058))

str-format (Python 2.6+) formatted print
print("Art: {0:5d}, Price per Unit: {1:8.2f}".format(435, 59.058))

f-string (Python 3.6+) formatted printing
A = 435; B = 59.058
print(f"Art:{A:5d}, Price per Unit: {B:8.2f}")
```

str-format

Var = “{} {}”.format(varA, varB)



說明：
{0:5d} #將後面第0個變數以寬度5格印成整數字串
{1:8.2f} #將後面第1個變數以寬度8格印成成小數2位的字串

Topic 9- List initialization and list comprehension

- Step 1: List initialization
- Step 2: Use List comprehension to initialize

Thanks! Q&A