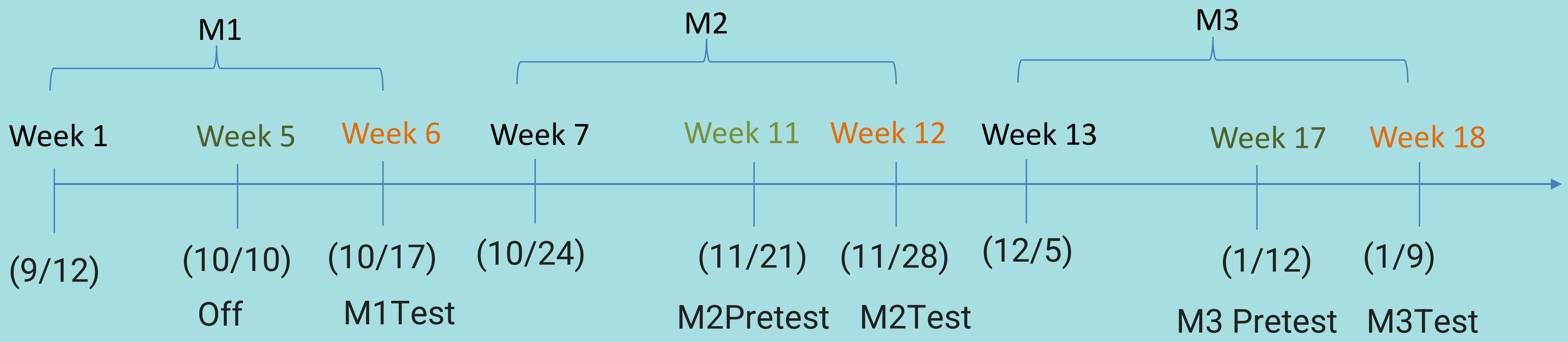
Fundamental Programming Course
Week 2

亞大資工系

Schedule

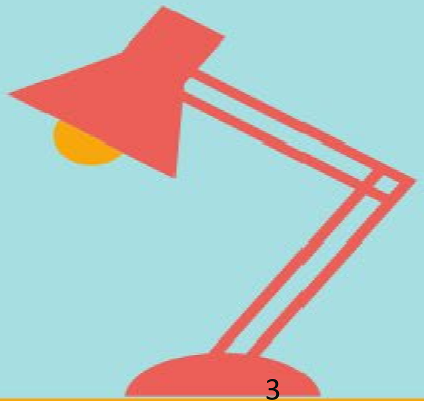

M1
M2
M3
Week 1
Week 5
Week 6
Week 7
Week 11
Week 12
Week 13
Week 17
Week 18
(9/12)
(10/10)
Off
(10/17)
M1Test
(10/24)
(11/21)
M2Pretest
(11/28)
M2Test
(12/5)
(1/12)
M3 Pretest
(1/9)
M3Test

Syllabus

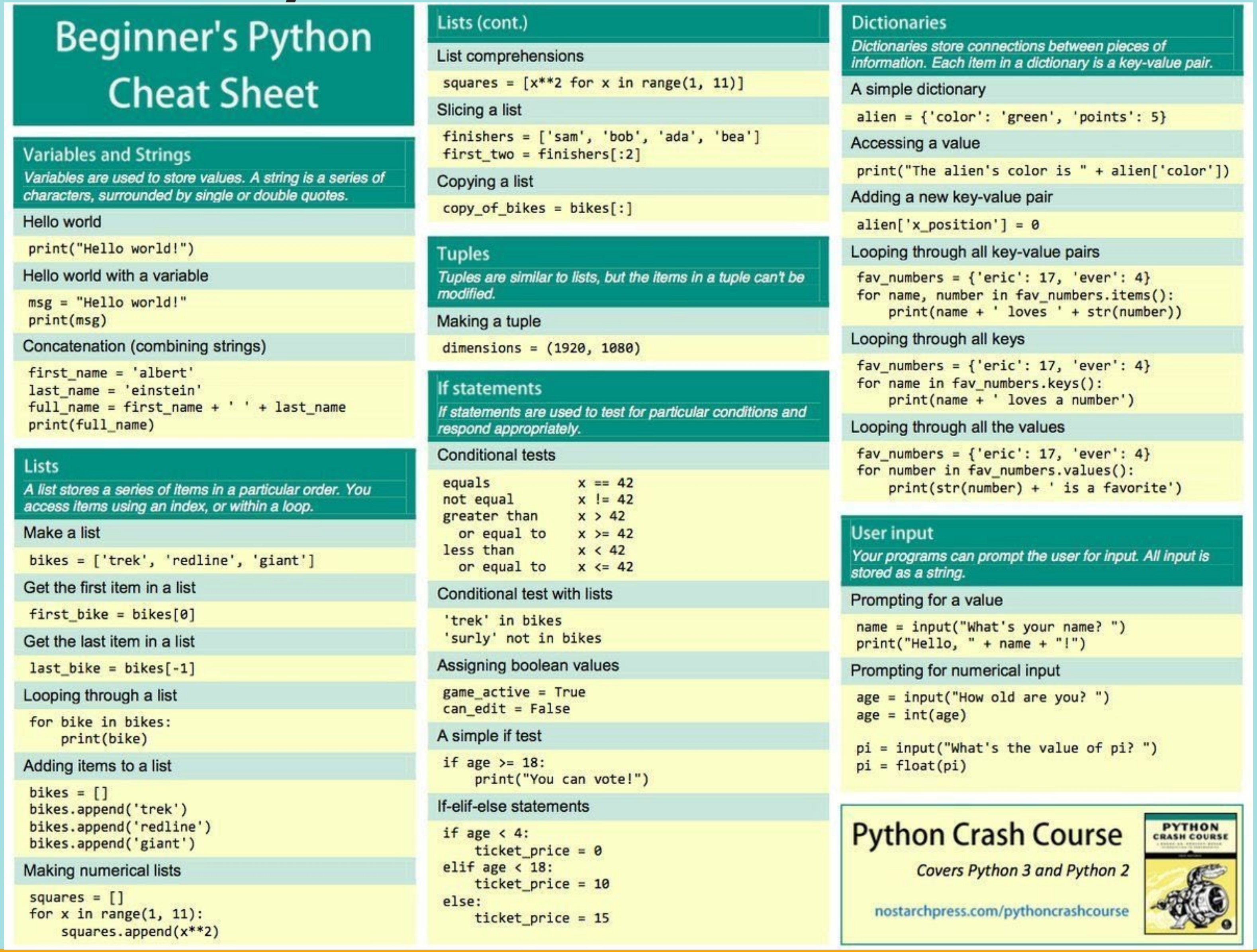
- W1-Python Introduction and Programming Tools
- W2-Variables and Operations
- W3-Loop and formatted output
- W4-Condition and Containers
- W5-String and built-in functions
- W6-M1 test

- 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 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)
```

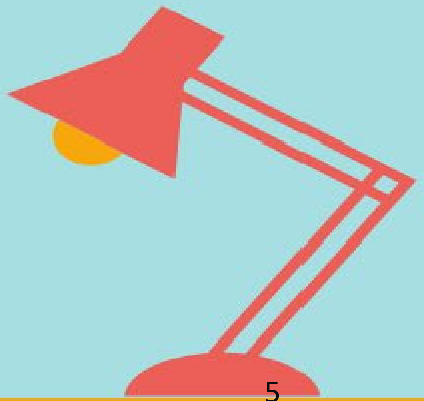
Python Crash Course

Covers Python 3 and Python 2

nostarchpress.com/pythoncrashcourse

Content

- Essential -
  - IPO model: input–process–output
    - Input: input() function, variable type conversion (int, float)
    - Process: Arithmetic Operators and Expressions
    - Process: operator priority order
    - Output: Parameters of the print() function (sep, end, file, flush)
  - Application of standard library math
  - Program Comments
- Advanced-Advanced
  - multi-line string
  - Markdown syntax



Kissipo Learning

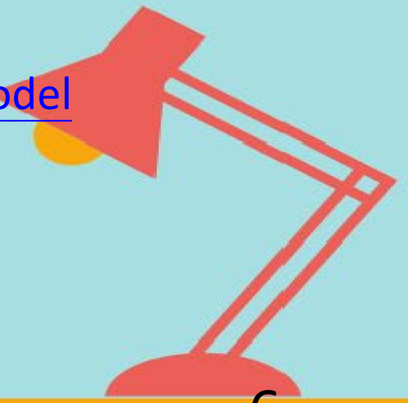
Kissipo = KISS principle + IPO model

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()

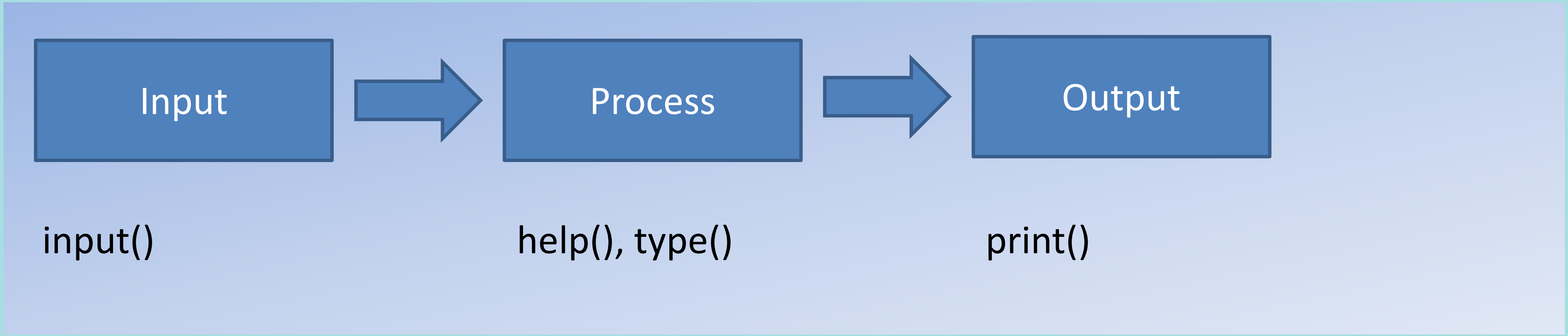
IPO-P: Process

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

IPO-O: output

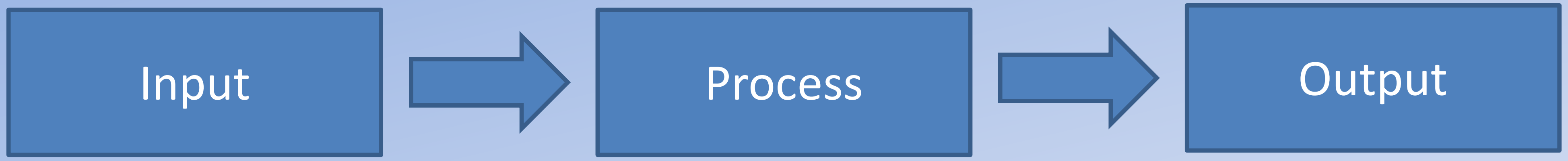
print()
open(), write()

IPO Model(1)



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

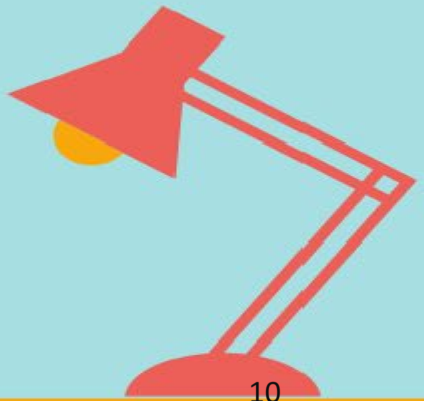
The print() function
Escape Sequence

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.

Topic 1- Input an integer or floating point number

- Step 1: Enter an integer
- Step 2: Enter a floating point number

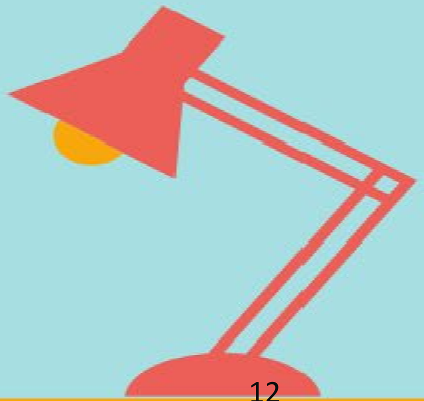


Topic 2- Arithmetic Operators and Expressions

- Step 1: Addition, subtraction, multiplication and division
- Step 2: Quotient and remainder
- Step 3: Power

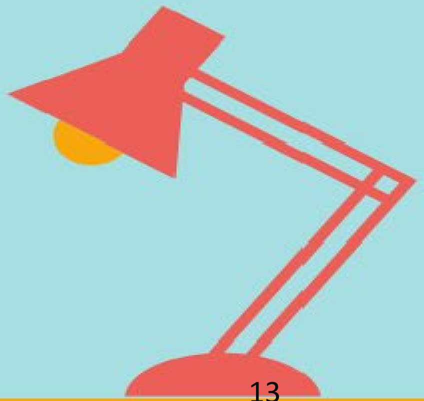
Topic 3: Operator precedence

- Step 1: Multiply and divide first, then add and subtract, parentheses first
- Step 2: The power is given priority over addition, subtraction, multiplication and division



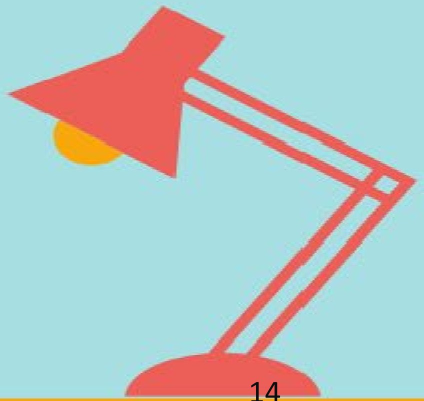
Topic 4- Application of standard library math

- Step 1: Calculate the pi and sin(π/3) functions
- Step 2: Use the pi and sin functions of the math standard library
- Step 3: use as
- Step 4: Use the standard library math's angle (degree) and radian (radian) conversion



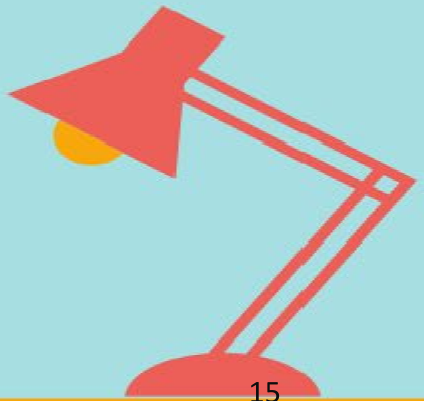
Topic 5: Parameters of the print() function

- Step 1: Hello World with parameters
- Step 2: Escape Sequence



Topic 6: Multiline string

- Step 1:
  - Use the \ at the end of the string to create a long string
- Step 2:
  - Use six double quotes to create long strings ''' ... ''' or """ ... """



Toic 7: character encoding of source code

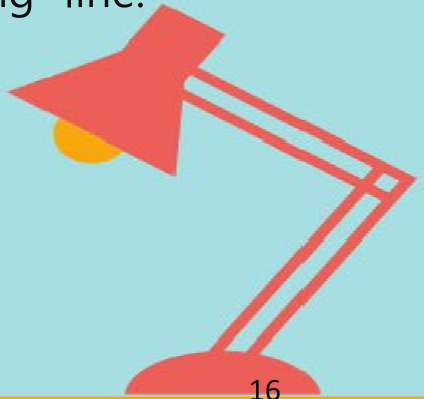
- The default character encoding for Python source files is UTF-8.
- If the default encoding is not used, the encoding of the file should be declared, and the first line of the file should be written as a special comment. The syntax is as follows:

- # -*- coding: encoding -*-
- Among them, encoding can be any codecs supported by Python.

- For example, to declare the use of Windows-1252 encoding, the source file should be written as:

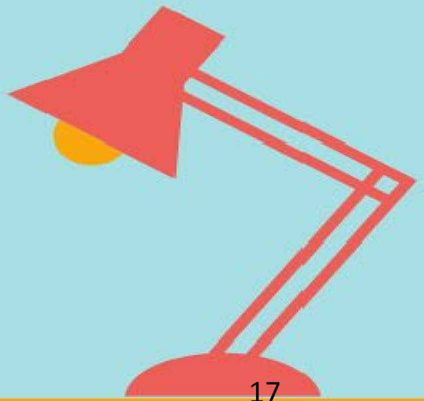
- # -*- coding: cp1252 -*-
- There is also an exception to the first line rule, when the source code begins with a UNIX "shebang" line. In this case, the encoding declaration is to be written on the second line of the file. E.g:

- # !/usr/bin/env python3
- # -*- coding: cp1252 -*-

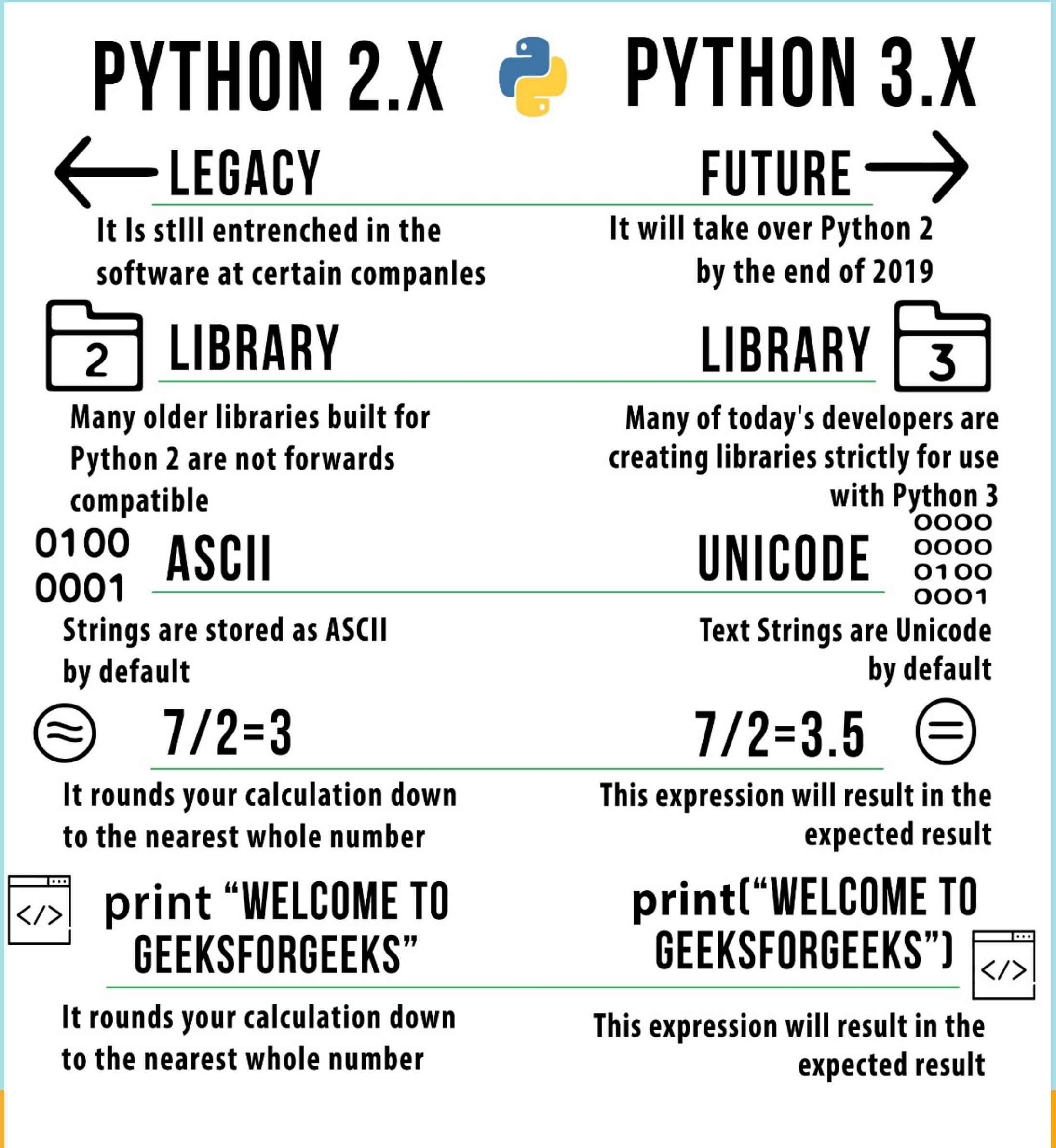


Topic 8: MarkDown syntax

- Step 1: Title
- Step 2: Dividers
- Step 3: Bold and Italic
- Step 4: Checklist



Python Versions



PYTHON 2.X	PYTHON 3.X
LEGACY	FUTURE
It Is stlll entrenched in the software at certain companles	It will take over Python 2 by the end of 2019
LIBRARY	LIBRARY
Many older libraries built for Python 2 are not forwards compatible	Many of today's developers are creating libraries strictly for use with Python 3
ASCII	UNICODE
Strings are stored as ASCII by default	Text Strings are Unicode by default
7/2=3	7/2=3.5
It rounds your calculation down to the nearest whole number	This expression will result in the expected result
print "WELCOME TO GEEKSFORGEEKS"	print("WELCOME TO GEEKSFORGEEKS")
It rounds your calculation down to the nearest whole number	This expression will result in the expected result

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

Thanks! Q&A