GENERAL ASSEMBLY

PYTHON FOR BEGINNERS

WiFi: GA
Password: yellowpencil

SIOW Yi Sheng
Second Career Full Stack Developer @ The Artling Pte. Ltd.

Siow Yi Sheng

Full Stack Developer @ The Artling

- Self-learned Second Career Developer
- Data Analyst / Full Stack Developer @ The Artling
- Python mentor and teacher
- PyCon SG Workshop Teacher
- Python User Group SG Speaker

- Websites?
- Robots?
- Data Science?
- Games?
- Cryptocurrencies?

WHAT IS **PROGRAMMING**?

Teaching Computers to **Do Things**

```
queryset.filter(title='game of thrones')
```

```
classifier.fit(X, y)
```

```
show(plot)
```

```
pdf.image('logo.png')
```

```
image_to_string('contract.png')
```

- Programming - 1940s
- Punch card programming - 1970s
- Python - 1991
- Guido van Rossum (BDFL)
- Django - 2003
- Sklearn - 2010

Readable

Python

```
stuff = ["Hello, World!", "Hi there, Everyone!", 6]
for i in stuff:
    print(i)
```

Readable

Java

```java
public class Test {
    public static void main(String args[]) {
        String array[] = {"Hello, World", "Hi there, Everyone", "6"};
        for (String i : array) {
          System.out.println(i);
        }
    }
}
```

Agenda

1. 🖼️ The Bigger Picture
2. 📦 Variables *naming and organising our things*
3. 👨‍🔧 Functions *doing things*
4. 🔡🔢 Types *doing different things*
5. ⛔ If *sometimes don't do the thing*
6. 💯 Loops *doing things many times*
7. 👨‍🚀 Modules *doing things coded by other people*
8. 👩‍💻 Practice

The Bigger Picture

The Bigger Picture: Language vs domain

Eng

- “Please mow the lawn at 3pm, then if the baby is not asleep, read her some stories from the library until she falls asleep.”

Chi

- “请在下午3点割草坪，如果婴儿没有睡觉，请从图书馆读一些故事直到她入睡。

Py

-
```
if time == ‘3pm’:
    lawn.mow()
    if not baby.is_sleeping:
        borrow_books_from_library()
        while not baby.is_sleeping:
            read_to_baby()
```

Language

Domain: Housekeeping

The Bigger Picture: Language vs domain

Eng
- “Split the dataset into a 80-20 test train split, then train the model on the test dataset using a decision tree classifier”

Chi
- “将数据集拆分为80-20测试列分割，然后使用决策树分类器在测试数据集上训练模型”

Py
- `test, train = test_train_split(X)`
- `DecisionTreeClassifier.fit(train)`

Language

Domain:
Data Science

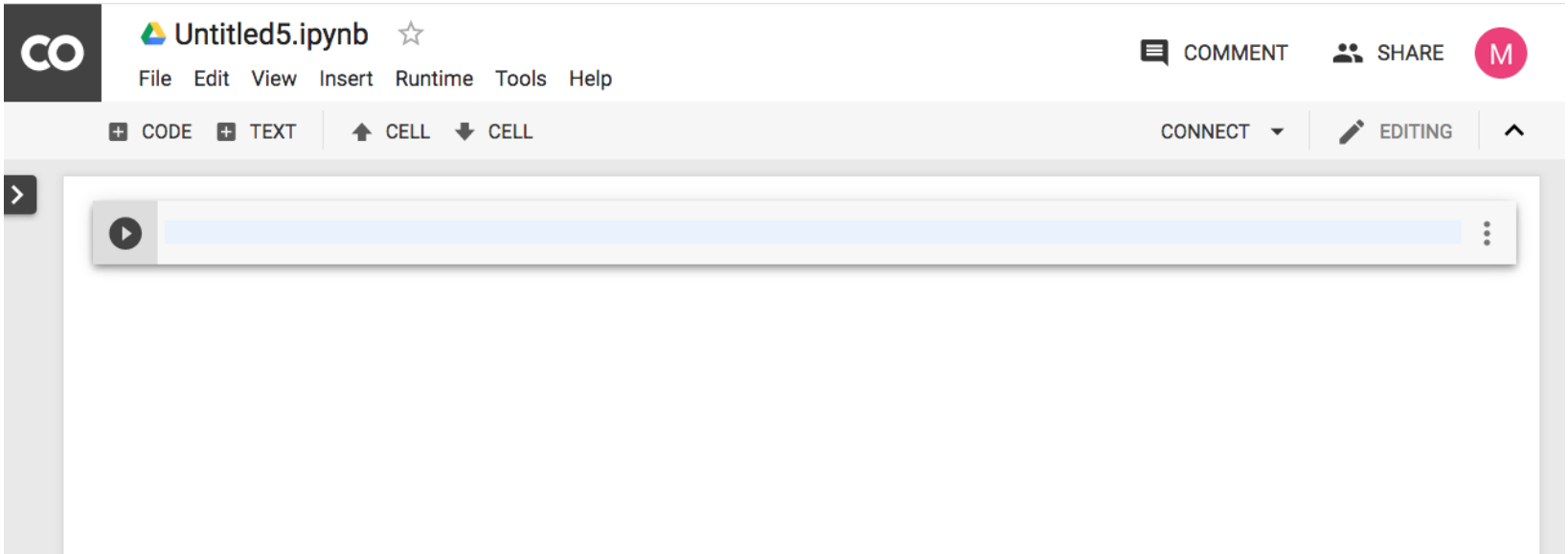
- Google for example code
- Copy and paste and edit sample code
- Read ‘documentation’
- Follow the **Getting Started** section of a new library
- Youtube tutorials if the documentation is bad
- Google for error messages
- Google questions => StackOverflow

- 'Data work' (analysis / science / collection / visualization)
  - pandas / numpy / sklearn / tensorflow / bokeh
- Backend web development
  - django
- Scripting
  - requests
- Embedded systems
  - PyBDM
- Financial Backtesting
  - zipline

1. Text editor / IDE (like vscode)
2. Jupyter notebooks (or Colab notebooks)
3. Command line

The Bigger Picture: Colab Notebooks

1. WIFI: GA / Password: yellowpencil
2. Go to https://colab.research.google.com/ in Chrome
3. Click at "New Python 3 Notebook” (You should see something like below)



Open http://tinyurl.com/ga-py-bulbasaur in another tab

The Bigger Picture: Using Notebooks

- The Notebook consists of **Cells** that are mini code editors.
- Each Cell can contain any amount of Python code
- **SHIFT+ENTER:** runs code and shows output
- **Cmd/Ctrl + m, d**: delete the cell
- Saving creates **filename.ipynb** files
- You can open these .ipynb files from jupyter

 The Bigger Picture: Real Code in Numbers

- A statement ➡️ 1 LOC (Line of code)
- A function ➡️ 5 - 50 LOC
- A file ➡️ 10 - 1,000 LOC
- A module ➡️ 100 - 100,000 LOC

Agenda

1. 🖼️ The Bigger Picture
2. 📦 Variables *naming and organising our things*
3. 👨‍🔧 Functions *doing things*
4. 🔡🔢 Types *doing different things*
5. ⛔ If *sometimes don't do the thing*
6. 💯 Loops *doing things many times*
7. 👨‍🚀 Modules *doing things coded by other people*
8. 👩‍💻 Practice

Variables

naming and organising things

- Named boxes which can hold information 📦📦📦📦📦.
- We can choose any names, and we should choose good names.
- This makes teaching computers easier.
- The box can grow and is *unlimited in size*, or it can be *empty.*

The right side <u>always</u> gets evaluated first!

```
students = 15
difference = 8 - 2
difference = 100 - 20
difference = students + 10
difference = difference - 5
is_sleeping = True
word_of_the_day = ‘preposterous’
```

- Extra spaces in Python have no meaning to the computer
- But they are important to keep the code neat for humans
- In practice, this is mostly handled by the text editor / notebook

Variables: Naming Rules

- Names cannot have spaces.
- Names cannot start with a number.
- Names should use all small letters.
- Multiple words should be separated by underscores _
- You should not use names of built-in Python functions.

 Variables: Good and Bad Names

- Good names tell you about the info the box should hold.
- Good names are short.
- Bad names are ambiguous.

- What would you name the following?
  - The number of students who didn’t come to class on time
  - Whether a game character is alive or not
  - The text to show on a website’s banner

```
a = 5
b = 10
c = a + b
d = a + c
d = d - 2
e = a + b + c + d
```

Agenda

1. 🖼️ The Bigger Picture
2. 📦 Variables *naming and organising our things*
3. 👨‍🔧 Functions *doing things*
4. 🔡🔢 Types *doing different things*
5. ⛔ If *sometimes don't do the thing*
6. 💯 Loops *doing things many times*
7. 👨‍🚀 Modules *doing things coded by other people*
8. 👩‍💻 Practice

Functions

doing things

A FUNCTION IS A WORKER 👨‍🔧

1. You **create and name** the worker 👶
2. You **teach** 👩🏻‍🏫 him to do things (often based on some **input** 🔢), and what to give you back **(return** 🧾**)** when he's done
3. Then you **tell him to do what you taught** him 👨‍🔧

- In practice, most functions you use are already written by others
- They are built-in to Python or imported in ‘modules’
- So it’s most important to know how to run(call) functions
- `print(‘hello world’)`

```
def add_one_plus_one():
  return 1 + 1


answer = add_one_plus_one()
```

```
def add_one_plus_one👶()🔢:
  return 1 + 1👩‍🏫🧾

answer = add_one_plus_one()👨‍🔧
```

- Indentation has meaning to the computer
- Levels of indentation must be consistent
- The standard is 4 ‘spaces’
- In practice, this is usually handled by the text editor / notebook

```
def add_one(n):
  return n + 1


answer = add_one(5)
```

```
def add(m, n):
  return m + n


answer = add(2, 4)
```

```
def add(m, n=5):
  return m + n


answer = add(2)
answer = add(2, 7)
answer = add(2, n=7)
```

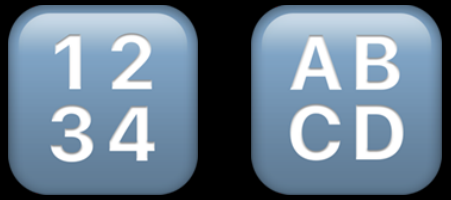
```
class Computer:
  def add(self, m, n):
    return m + n

computer = Computer()
answer = computer.add(2, 2)
```

Agenda

1. 🖼️ The Bigger Picture
2. 📦 Variables *naming and organising our things*
3. 👨‍🔧 Functions *doing things*
4. 🔡🔢 Types *doing different things*
5. ⛔ If *sometimes don't do the thing*
6. 💯 Loops *doing things many times*
7. 👨‍🚀 Modules *doing things coded by other people*
8. 👩‍💻 Practice

Types



doing different things

Functions won’t work if you give them input of the wrong type

- `chr('a')`
- `pow('a', 3)`

Number

- Written normally
- There are integers, floats, Decimals
- But they are all Numbers
- `students = 15`

Words / Characters (String)

- Written with quotes ‘’ around them
- Single quotes or double quotes are fine, but be consistent
- `pokemon_name = 'pikachu'`
- `character = 'p'`
- `works_of_shakespeare = '...'`

True or False (Boolean)

- `True`
- `False`
- Note the capital T and F
- `is_sleeping = True`
- `is_bold = False`

List

- A type that can hold many things.
- Written with square brackets around the items(elements), and commas between items `[item1, item2]`
- An item can be of any type, even another list!
- Each item in a list has an address (index), starting from 0
- `cars = [‘honda’, ‘ferrari’]`

List

- We can refer to items in lists using their address, by writing the address in square brackets right after the list.
- `cars[1]`
- `['honda', 'ferrari'][1]`

List

- We can add items to lists using `.append(item)`
- `cars = [‘honda’, ‘ferrari’]`
- `cars.append(‘mitsubishi’)`
- We can change values of items in lists
- `cars[1] = ‘subaru’`

Tuple (Special List)

- Written with either brackets around the items or nothing around the items, and commas between items
- `items = (item1, item2)`
- `items = item1, item2`
- Similar to lists.
- Commonly used when returning multiple values from a function
- `return item1, item2`

Dictionary

- A type that can hold many things, each with a name!

```
pokemon = {‘name’: ‘pikachu’, ‘attack’: 25 }
```

```
{
   ‘name’: [‘pikachu’, ‘raichu’],
   ‘attack’: [25, 45],
}
```

Dictionary

- Made up of pairs (key-value pairs)
- Written with curly brackets around the pairs, colons between the key and value, and commas between pairs.
- The keys are often strings, sometimes numbers.

Dictionary

- We access the values by providing the key in square brackets after the dictionary.
- `my_pokemon = pokemon['name']`
- We can set new pairs.
- `pokemon['name'] = 'bulbasaur'`
- `pokemon['defense'] = 20`

Objects

- Another type that can hold many things, each with a name!
- Objects can have attributes.
- `apple.color = ‘red’`
- `book.title = ‘A Game of Thrones’`
- `book.year = 1996`
- `my_favorite_color = apple.color`

Agenda

1. 🖼️ The Bigger Picture
2. 📦 Variables *naming and organising our things*
3. 👨‍🔧 Functions *doing things*
4. 🔡🔢 Types *doing different things*
5. ⛔ If *sometimes don't do the thing*
6. 💯 Loops *doing things many times*
7. 👨‍🚀 Modules *doing things coded by other people*
8. 👩🏿‍💻 Practice

If

sometimes don't do the thing

- Let’s say we have a thing (object).
- We put it in a variable and gave it a name `baby`.
- It has an attribute `is_sleeping`, and the value is `True`
-
```
if baby.is_sleeping:
    take_a_nap()
```

- We can also do something if the condition is false.
- ```
  if baby.is_sleeping:
      take_a_nap()
  else:
      read_to_baby()
  ```
  ```

- The condition is boiled down to either `True` or `False`.
- If a condition or value boils down to `True`, we call it **truthy**.
- If it boils down to `False`, we call it **falsy**.
-
```
if baby.is_sleeping:
    take_a_nap()
else:
    read_to_baby()
```

- We can then modify/combine conditions using `and / or / not`
- ```
  if baby.is_sleeping and i.am_tired:
      take_a_nap()
  else:
      read_to_baby()
  ```
  ```

⛔ If

```
if baby.is_sleeping or i.am_tired:
    take_a_nap()
else:
    read_to_baby()
```

- ```
  if not baby.is_sleeping:
      read_to_baby()
  ```
  ```

- We can also make comparisons which will evaluate to `True` or `False`.
-
```
if person.age < 9:
    bus_fair = ‘free’
```

-
```
if number_of_criminals == number_of_policeman:
    call_batman()
```

-
```
if groceries_bought != groceries_wanted_to_buy:
    place_palm_to_forehead()
```

Agenda

1. 🖼️ The Bigger Picture
2. 📦 Variables *naming and organising our things*
3. 👨‍🔧 Functions *doing things*
4. 🔡🔢 Types *doing different things*
5. ⛔ If *sometimes don't do the thing*
6. 💯 Loops *doing things many times*
7. 👨‍🚀 Modules *doing things coded by other people*
8. 👩🏻‍💻 Practice

Loops

doing things many times

- The ‘for loop’ is the most important and common loop.
- It lets us do something n times.
- We can loop over lists, list-like objects, dictionaries.
- `for pokemon in pokedex:`
- `    catch(pokemon)`
- `for student in students:`
- `    teach_python_to(student)`

💯 Loops: for

```
for i in [1,2,3,4,5]:
    print('pika')
```

```
for i in [1,2,3,4,5]:
    if i == 3:
        continue
    print(i)
```

```
for i in [1,2,3,4,5]:
    if i == 3:
        break
    print(i)
```

💯 Loops: while

- The ‘while loop’ only ends if the condition turns falsy.
- `while not pokedex.is_complete:`
- `    catch_more_pokemon()`

Agenda

1. 🖼️ The Bigger Picture
2. 📦 Variables *naming and organising our things*
3. 👨‍🔧 Functions *doing things*
4. 🔡🔢 Types *doing different things*
5. ⛔ If *sometimes don't do the thing*
6. 💯 Loops *doing things many times*
7. 👨‍🚀 Modules *doing things coded by other people*
8. 👩🏿‍💻 Practice

Modules

doing things coded by other people

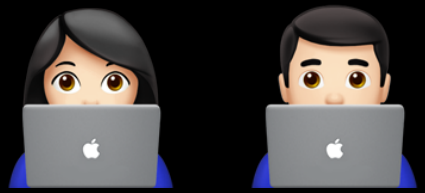
- Modules (libraries/packages) are pre-written code.
- We can download them easily from the command line or from within the notebook using `pip`.
- `$ pip install requests`
- Then we can use the code by importing it as a thing (object).
- `import random`
- `from random import randint`

- In practice, every domain has its own common modules and you will often be learning how to use new modules.
- To do so, we google for and read ‘documentation’.

Agenda

1. 🖼️ The Bigger Picture
2. 📦 Variables *naming and organising our things*
3. 👨‍🔧 Functions *doing things*
4. 🔡🔢 Types *doing different things*
5. ⛔ If *sometimes don't do the thing*
6. 💯 Loops *doing things many times*
7. 👨‍🚀 Modules *doing things coded by other people*
8. 👩‍💻 Practice

Practice



Task:

- Create a file called "random.txt"
- Write 10 random numbers into the file
- Each number should be on it's own line

Task:

- Open the file "random.txt" for reading
- Print all the lines inside the file on the screen

Task:

- Get Singapore's PSI data
- Print the national psi_twenty_four_hourly value on the screen
- You can use this API: https://data.gov.sg/dataset/psi?resource_id=82776919-0de1-4faf-bd9e-9c997f9a729d
- Send a GET request to the correct API endpoint
- From the response, find the number that corresponds to the national psi_twenty_four_hourly reading
- Print that reading on the screen

Task:

- Draw a chart
- Any chart
- Learn how to change the values

Reference: Installing Python and Jupyter

- Download Python from https://www.python.org/
- Run `pip install jupyter` in the command line
- Run `jupyter notebook` in the command line
- You can click "New" and then "Python 3" to create a new Notebook.

Please take the survey:

http://bit.ly/Pyth101

siowyisheng@gmail.com

Want to Learn More?

Full-Time Courses

Pathways to fulfilling new careers in data science, software engineering, UX design & digital marketing

10 - 12 weeks -
Mon to Fri,
9 AM - 5 PM

Part-Time Courses

Evening, weekend, and accelerated opportunities spanning data, business, tech, and design

10 Weeks Mon/Wed or Tue/Thu
7 PM - 9 PM or Sat 10 AM - 5 PM
1-Week Accelerated - Mon to Fri
9 AM - 5 PM

Classes and Workshops

Online and on-campus happenings highlight leading innovations, technologies, and trends

GA.CO/EDUCATION

Q&A

@GeneralAssemblySG

@siowyisheng

#LifeatGA

siowyisheng@gmail.com