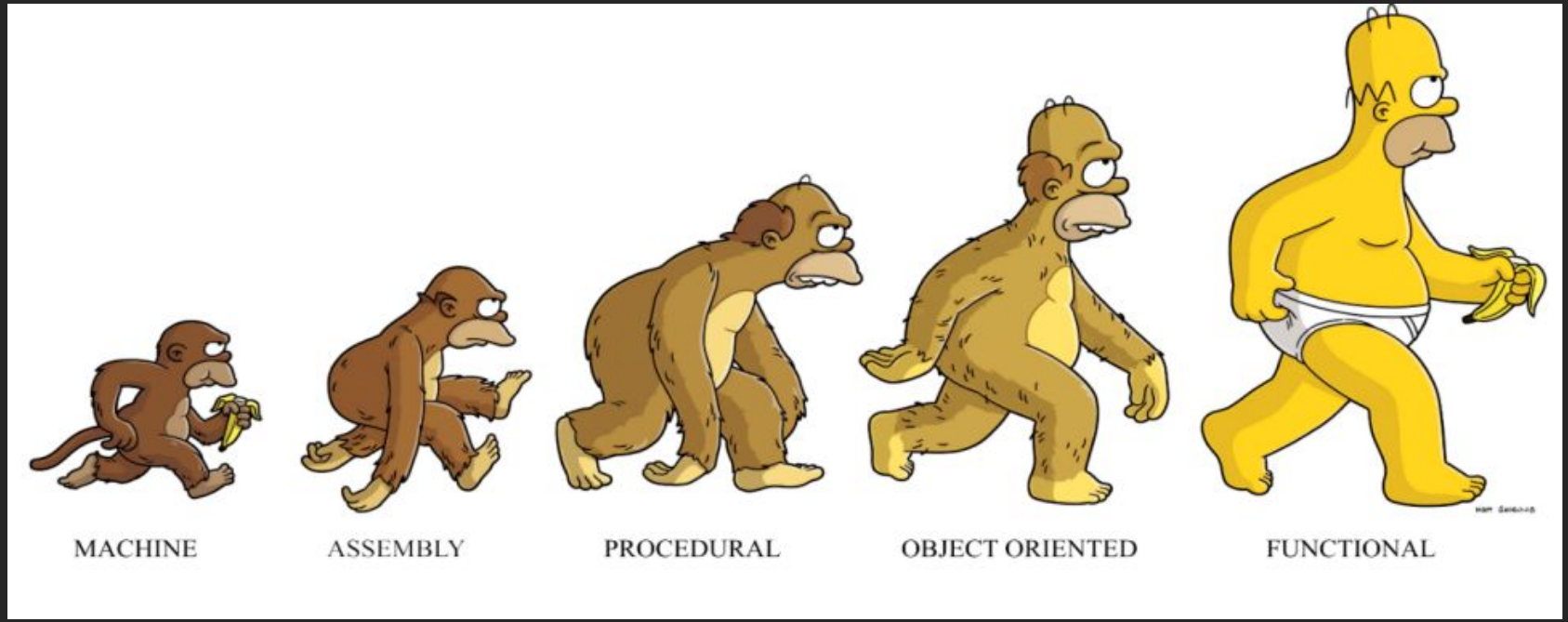
แนวความคิดภาษาเชิงวัตถุ

OOP (Object Oriented Programming)

คือ การเขียนโปรแกรมอีกรูปแบบหนึ่ง โดยมองสิ่งต่างๆเป็น **วัตถุ** โดยในวัตถุจะมี**คุณสมบัติ**และ**พฤติกรรม**ซึ่งมีมุมมองจากพื้นฐานความจริงในชีวิตประจำวัน



https://www.youtube.com/channel/UCQ1r_4x-P-fETLIU4pqf98w https://www.facebook.com/KongRuksiamTutorial/

เชิงกระบวนการ VS เชิงวัตถุ

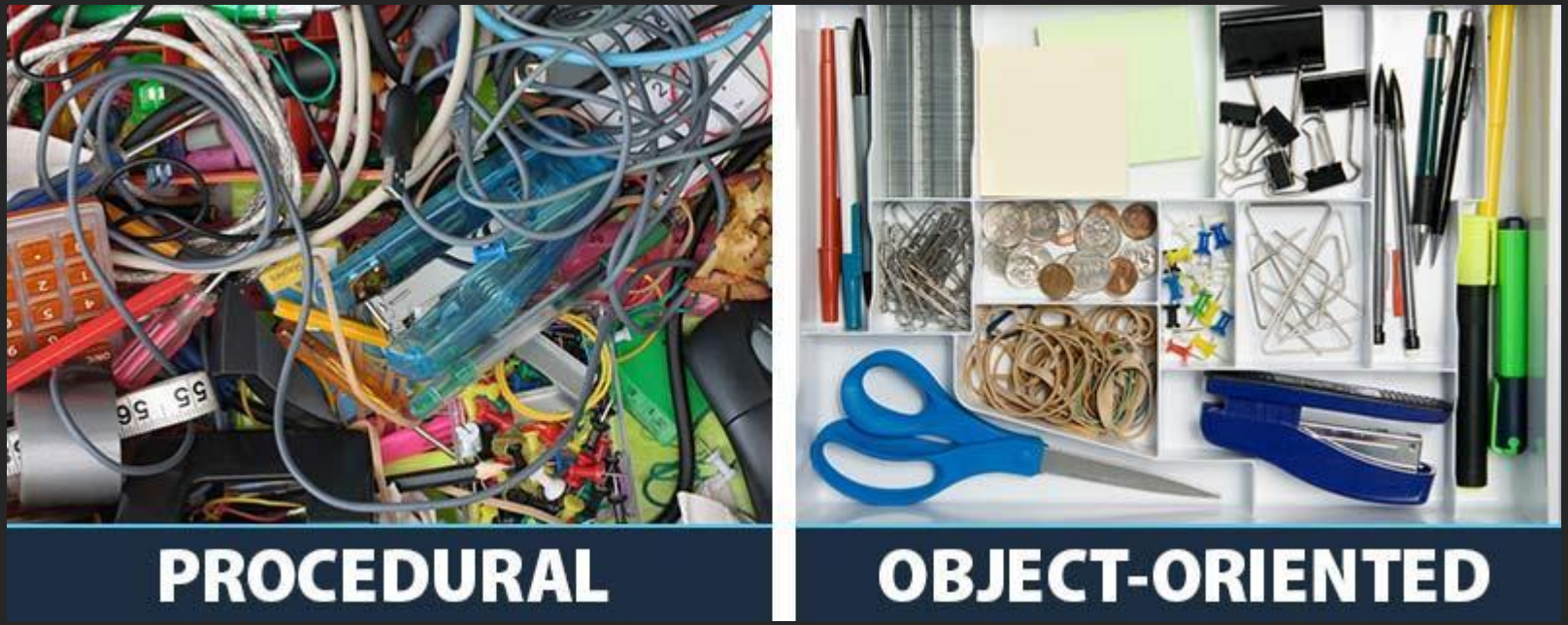
ภาษาเชิงกระบวนการ (Procedural Programming Language)

- โปรแกรมจะแบ่งออกเป็นส่วนย่อยๆ เรียกว่าโมดูล (Module)
- แต่ละโมดูลควรออกแบบให้มีการทำงานเพียง 1 งานเท่านั้น
- การออกแบบให้แต่ละโมดูลมีความเป็นอิสระต่อกันนั้นทำได้ยาก

ภาษาเชิงวัตถุ (Object Oriented Programming Language)

- การพัฒนาโปรแกรมเป็นการเลียนแบบการทำงานเชิงออปเจ็ค
- การออกแบบให้วัตถุมีความเป็นอิสระต่อกันทำได้ง่ายด้วยคุณสมบัติเชิงวัตถุ
- สามารถนำโปรแกรมกลับมาใช้ใหม่ (Reuse) ได้ดีกว่าภาษาเชิงกระบวนการ

เชิงกระบวนการ VS เชิงวัตถุ



https://www.youtube.com/channel/UCQ1r_4x-P-fETLIU4pqf98w https://www.facebook.com/KongRuksiamTutorial/

เชิงกระบวนการ VS เชิงวัตถุ

ภาษาเชิงกระบวนการ	ภาษาเชิงวัตถุ
กำหนดขั้นตอนการแก้ปัญหา	กำหนดปัญหาเป็นองค์ประกอบ (วัตถุ)
โปรแกรมและข้อมูลอยู่คนละส่วนกัน	เอาส่วนโปรแกรมและข้อมูลไว้ด้วยกัน
ออกแบบจากล่างขึ้นบน	ออกแบบเป็นวัตถุ
แก้ไขง่ายเพราะแต่ละส่วนไม่มีความสัมพันธ์กัน	การแก้ไขไม่กระทบส่วนอื่นๆของโปรแกรมเพราะวัตถุจะมีความสมบูรณ์ในตัวเอง

แนวความคิดภาษาเชิงวัตถุ

องค์ประกอบพื้นฐานในภาษาเชิงวัตถุ

1. คลาส (Class) & วัตถุ (Object)
2. การห่อหุ้ม (Encapsulation)
3. การสืบทอดคุณสมบัติ (Inheritance)
4. การพ้องรูป (Polymorphism)

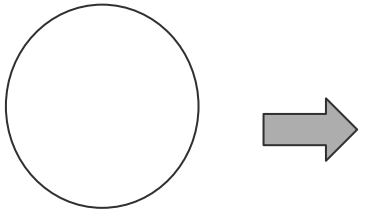
แนวความคิดภาษาเชิงวัตถุ

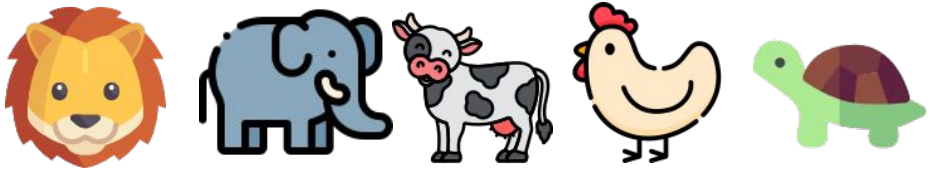
คลาส (class) คือ ต้นแบบของวัตถุการจะสร้างวัตถุขึ้นมาได้จะต้องสร้างคลาสขึ้นมาเป็นโครงสร้างต้นแบบสำหรับวัตถุก่อนเสมอ

วัตถุหรือออบเจ็ค (object) คือ สิ่งที่ถูกสร้างจากคลาส ประกอบด้วยคุณสมบัติ 2 ประการ คือ คุณลักษณะ และพฤติกรรม

แนวความคิดภาษาเชิงวัตถุ

Class → **Object**

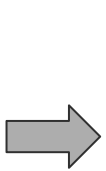




Animal — สิงโต, ช้าง, วัว, ไก่, เต่า

แนวความคิดภาษาเชิงวัตถุ

Class



Object

Employee Accounting Programmer Sale

แนวความคิดภาษาเชิงวัตถุ

คุณลักษณะ (Attribute หรือ Data member) สิ่งที่บ่งบอกลักษณะทั่วไปของวัตถุ

พฤติกรรม (Behavior หรือ Method) คือ พฤติกรรมทั่วไปของวัตถุที่สามารถกระทำได้

Animal

คุณสมบัติ (Attribute)

ชื่อ : ช้าง

สี : ฟ้าอ่อน

ประเภท : สัตว์บก

น้ำหนัก : 6 ตัน

จำนวนเท้า : 4 เท้า

พฤติกรรม (Method/Behavior)

- **วิ่ง**
- **นอน**
- **ส่งเสียงร้อง**

Animal

คุณสมบัติ (Attribute)

ชื่อ : นก

สี : เหลือง

ประเภท : สัตว์ปีก

น้ำหนัก : 0.8 กิโลกรัม

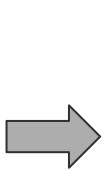
จำนวนเท้า : 2 เท้า

พฤติกรรม (Method/Behavior)

- **บิน**
- **เดิน**
- **ส่งเสียงร้อง**

แนวความคิดภาษาเชิงวัตถุ

Class



Object

Employee Accounting Programmer Sale

Employee

Accounting	Programmer	Sale
Attribute - ชื่อ - เงินเดือน	**Attribute** - ชื่อ - เงินเดือน - ประสบการณ์ทำงาน	**Attribute** - ชื่อ - เงินเดือน - เขตพื้นที่รับผิดชอบ
Method - คำนวณเงินเดือน - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าล่วงเวลา - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าคอมมิสชั่น - แสดงรายละเอียด

สรุปง่ายๆ

- **Class – ต้นแบบของวัตถุ**
- **Object – สิ่งที่ถูกสร้างขึ้นมาจาก Class ประกอบด้วย**
  - คุณสมบัติ (Attribute)
  - พฤติกรรม (Method)
- **คุณสมบัติของการเขียนโปรแกรมเชิงวัตถุ**
  - การห่อหุ้ม(Encapsulation)
  - การสืบทอด (Inheritance)
  - การพ้องรูป (Polymorphism)

การสร้าง Class & Object

- **คลาส (class)** คือ ต้นแบบของวัตถุ หรือ แม่แบบสำหรับวัตถุ (Template, Prototype)
- **วัตถุ (Object)** คือ สิ่งที่ถูกสร้างขึ้นจากคลาส

https://www.youtube.com/channel/UCQ1r_4x-P-fETLIU4pqf98w

https://www.facebook.com/KongRuksiamTutorial/

Attribute เป็นกลไกที่กำหนดคุณสมบัติให้กับคลาส

การสร้าง Attribute

self.name = ชื่อพนักงาน

self.salary = เงินเดือนพนักงาน

self.age = อายุพนักงาน

Method เป็นกลไกที่กำหนดพฤติกรรมให้กับคลาส

การสร้าง Method

```
def getName(self):
    return self.name
```

การเรียกใช้งาน

```
ชื่อวัตถุ.getName()
```

คีย์เวิร์ด Self

การใช้คีย์เวิร์ด **self** จะเป็นตัวชี้หรือตัวที่บ่งบอกว่าตอนนี้เราทำงานกับวัตถุใด ให้บอกตัวตนของวัตถุนั้นๆ เช่น การกำหนดคุณสมบัติต่างๆ ในวัตถุ เป็นต้น

Constructor

เป็น Method พิเศษที่จะถูกใช้งานเมื่อตอนเริ่มต้นสร้างวัตถุ

(ไม่ระบุก็ได้)

โครงสร้าง Constructor

```
def __init__(self):
    pass
```

Destructor

เป็น Method พิเศษที่ตรงข้ามกับ Constructor จะถูกใช้งาน**เมื่อสิ้นสุดการทำงานของ class** หรือถูกทำก่อนจะสลาย object ส่วนใหญ่จะเป็นกลุ่มคำสั่งที่ทำหน้าที่คืนหน่วยความจำให้ระบบ **(ไม่ระบุก็ได้)**

โครงสร้าง Destructor

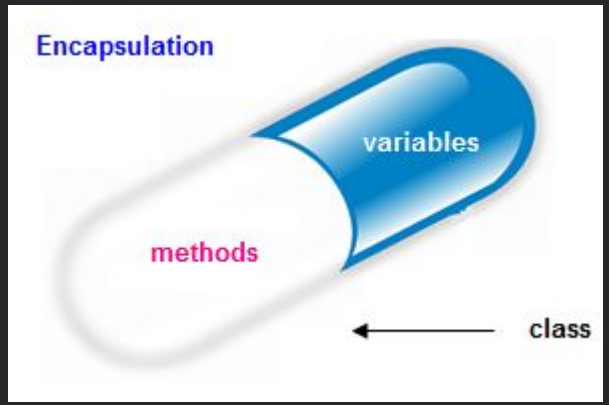
```
def __del__(self):
    pass
```

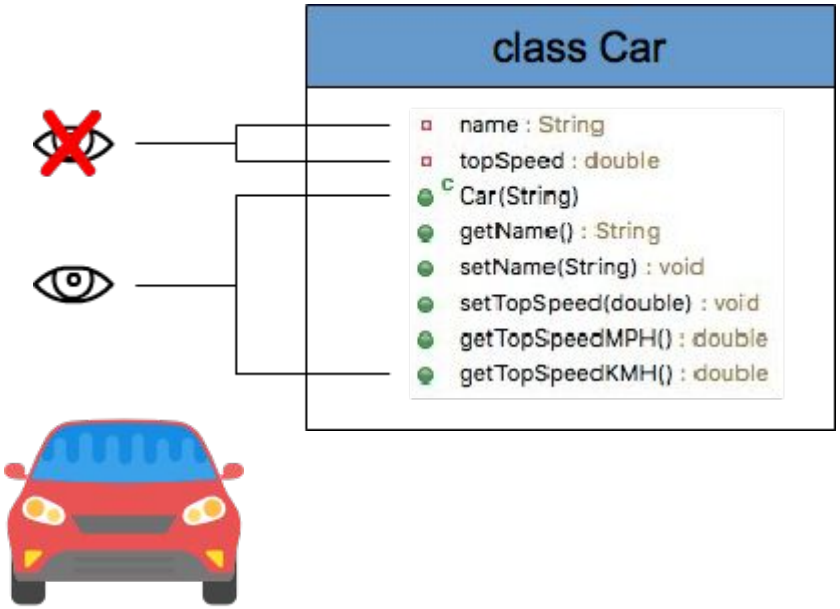
isinstance และ dir คือฟังก์ชั่นที่ทำงานกับ class และ object โดยมีรายละเอียดดังนี้

- isinstance => เช็คว่า object นี้ถูกสร้างจาก class ที่เรานิยามหรือไม่
- dir => แสดง Attribute และ Method
- __class__ => แสดงชื่อ class ของ object

การห่อหุ้ม (Encapsulation)

- เป็นกระบวนการซ่อนรายละเอียดการทำงานและข้อมูลไว้ภายในไม่ให้ภายนอกสามารถมองเห็นได้
- ทำให้ภายนอกไม่สามารถทำการเปลี่ยนแปลงแก้ไขข้อมูลภายในได้ ซึ่งเป็นผลทำให้เกิดความเสียหายแก่ข้อมูล
- ข้อดีของการห่อหุ้มคือสามารถสร้างความปลอดภัยให้แก่ข้อมูลได้ เนื่องจากข้อมูลจะถูกเข้าถึงจากผู้มีสิทธิ์เท่านั้น





https://www.youtube.com/channel/UCQ1r_4x-P-fETLIU4pqf98w https://www.facebook.com/KongRuksiamTutorial/

Access Modifier คือ ระดับในการเข้าถึง Class, Attribute, Method และอื่น ๆ ในภาษาเชิงวัตถุ โดยมีประโยชน์อย่างมากในเรื่องของการกำหนดระดับการเข้าถึง สิทธิในการเข้าใช้งาน การซ่อนข้อมูล และอื่น ๆ

Public เป็นการประกาศระดับการเข้าถึงที่เข้มงวดน้อยที่สุด หรือกล่าวได้ว่าใคร ๆ ก็สามารถเข้าถึงและเรียกใช้งานได้

Protected (_) เป็นการประกาศระดับการเข้าถึงเฉพาะคลาสของตัวมันเองและคลาสลูกที่สืบทอดคุณสมบัติไปใช้เท่านั้น

Private (__) เป็นการประกาศระดับการเข้าถึงที่เข้มงวดที่สุด กล่าวคือ จะมีแต่คลาสของตัวมันเองเท่านั้นที่มีสิทธิ์ใช้งานได้

Setter , Getter Method

Setter การกำหนดค่าให้ Object

```
def setName(self,newname):
    self.__name = newname
```

Gettter การดึงค่าจาก Object

```
def getName(self):
    return self.__name
```

Class Variable & Instance Variable

- Class Variable คือ ตัวแปรที่ทำงานภายใน class ส่วนอื่นสามารถเข้าถึงข้อมูลส่วนนี้ได้เลย (static attrbiute) โดยไม่จำเป็นต้องสร้าง Object ขึ้นมา

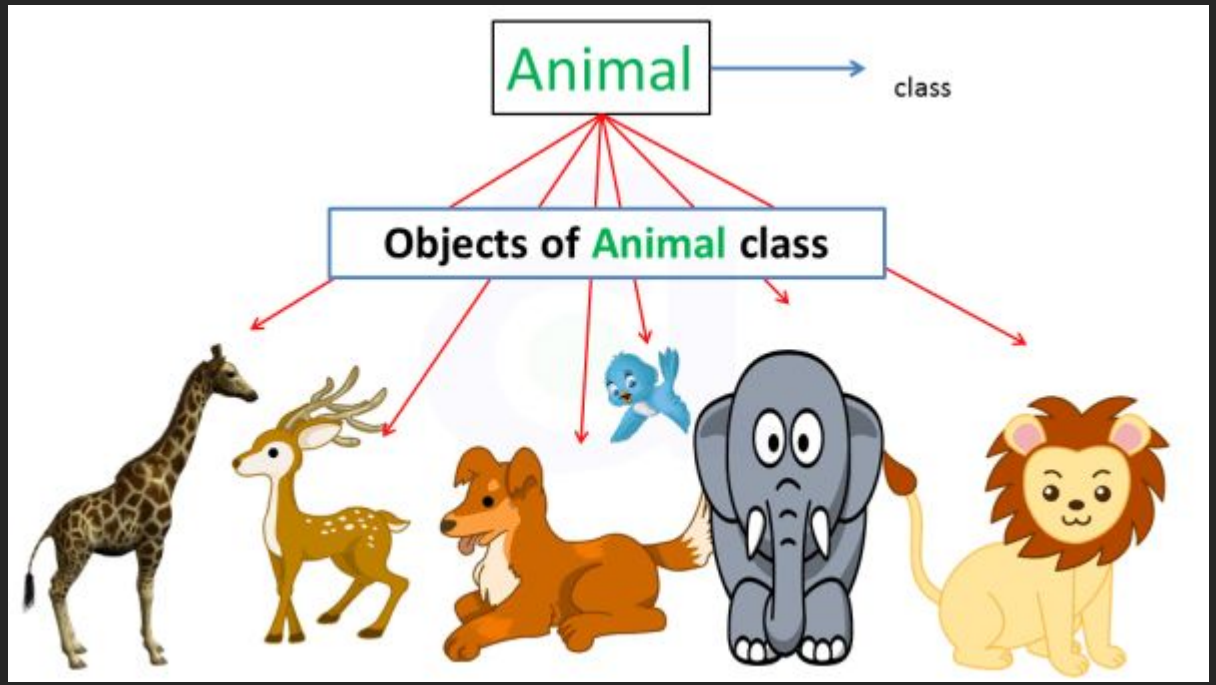
- Instance Variable คือ ตัวแปรที่อยู่ภายใน object สามารถเข้าถึงข้อมูลส่วนนี้โดยการต้องสร้าง Object ขึ้นมา

การสืบทอดคุณสมบัติ (Inheritance)

หลักการของ inheritance คือ ทำการสร้างสิ่งใหม่ขึ้นด้วยการสืบทอดหรือรับเอา (inherit) คุณสมบัติบางอย่างมาจากสิ่งเดิมที่มีอยู่แล้วโดยการสร้างเพิ่มเติมจากสิ่งที่มีอยู่แล้วได้เลย

ข้อดีของการ inheritance คือ จากการที่สามารถนำสิ่งที่เคยสร้างขึ้นแล้วนำกลับมาใช้ใหม่ (re-use) ได้ ทำให้ช่วยประหยัดเวลาการทำงานลง เนื่องจากไม่ต้องเสียเวลาพัฒนาใหม่หมด

คลาสแม่ (Superclass) คลาสลูก (Subclass)



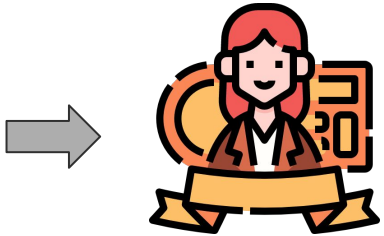
Employee

Accounting	Programmer	Sale
Attribute - ชื่อ - เงินเดือน	**Attribute** - ชื่อ - เงินเดือน - ประสบการณ์ทำงาน	**Attribute** - ชื่อ - เงินเดือน - เขตพื้นที่รับผิดชอบ
Method - คำนวณเงินเดือน - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าล่วงเวลา - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าคอมมิสชั่น - แสดงรายละเอียด

คุณสมบัติต่างๆจากแม่จะถูกถ่ายทอดไปยังลูก

Class

ยกเว้น Private Attribute & Private Method



Employee

Accounting

Programmer

Sale

การสืบทอดคุณสมบัติ

คลาสแม่

```
class Employee:
```

คลาสลูก

```
class Programmer(Employee)
```

คีย์เวิร์ด super

เมื่อต้องการเรียกใช้งานคุณบัติต่างๆในคลาสแม่ เช่น Constructor , Method , Attribute

```
super().__init__(name)
```

Employee

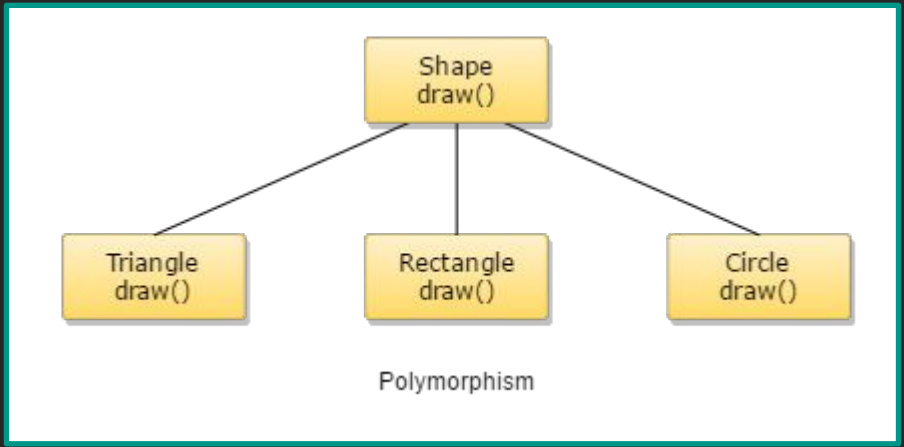
Accounting	Programmer	Sale
Attribute - ชื่อ - เงินเดือน - อายุ	**Attribute** - ชื่อ - เงินเดือน - ประสบการณ์ทำงาน	**Attribute** - ชื่อ - เงินเดือน - เขตพื้นที่รับผิดชอบ
Method - คำนวณเงินเดือน - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าล่วงเวลา - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าคอมมิสชั่น - แสดงรายละเอียด

การแปลง Object เป็น String

```
def __str__(self):
    return “ชุดข้อความ”
```

การพ้องรูป (Polymorphism)

Polymorphism เกิดจาก **poly** (หลากหลาย) + **morphology** (รูปแบบ)

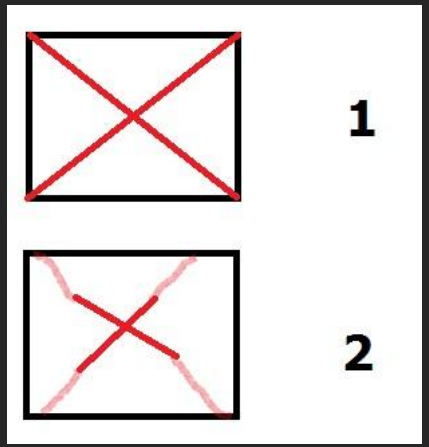


ในทางโปรแกรมคือการที่เมธอดชื่อเดียวกัน สามารถรับอาร์กิวเมนต์ที่แตกต่างกันได้หลายรูปแบบ โดยเมธอดนี้จะถูกเรียกว่า overload method (เมธอดถูกโอเวอร์โหลด)

Polymorphism

“ ข้อความเดียวกันแต่กระบวนการทำงานภายในแตกต่างกัน นั้น เรียกว่า **การพ้องรูป** หรือ **polymorphism** ”

กา



คุณสมบัติการพ้องรูป คือ คุณสมบัติที่สามารถตอบสนองต่อ Method เดียวกันด้วยวิธีการที่ต่างกันและสามารถกำหนด object ได้หลายรูปแบบ

ข้อดี คือ ทำให้โปรแกรมสามารถปรับเปลี่ยนหรือเพิ่มเติมได้ง่ายขึ้น

Overloading & Overriding Method

- **Overloading method** คือ **เมธอดที่มีชื่อเหมือนกัน**และ**อยู่ภายในคลาสเดียวกัน** สิ่งที่แยกความแตกต่างของเมธอดที่เป็น overload method คือ **พารามิเตอร์** (เป็นผลมาจากคุณสมบัติ OO คือ **polymorphism**)

- **Overriding method** คือ **เมธอดของคลาสลูก (subclass) ที่มีชื่อเหมือนกับเมธอดของคลาสแม่ (superclass)** (เป็นผลมาจากคุณสมบัติ OO
คือ inheritance)

Employee

Accounting	Programmer	Sale
Attribute - ชื่อ - เงินเดือน - อายุ	**Attribute** - ชื่อ - เงินเดือน - ประสบการณ์ทำงาน - ทักษะการทำงาน	**Attribute** - ชื่อ - เงินเดือน - เขตพื้นที่รับผิดชอบ
Method - คำนวณเงินเดือน - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าล่วงเวลา - แสดงรายละเอียด	**Method** - คำนวณเงินเดือน - ค่าคอมมิสชั่น - แสดงรายละเอียด