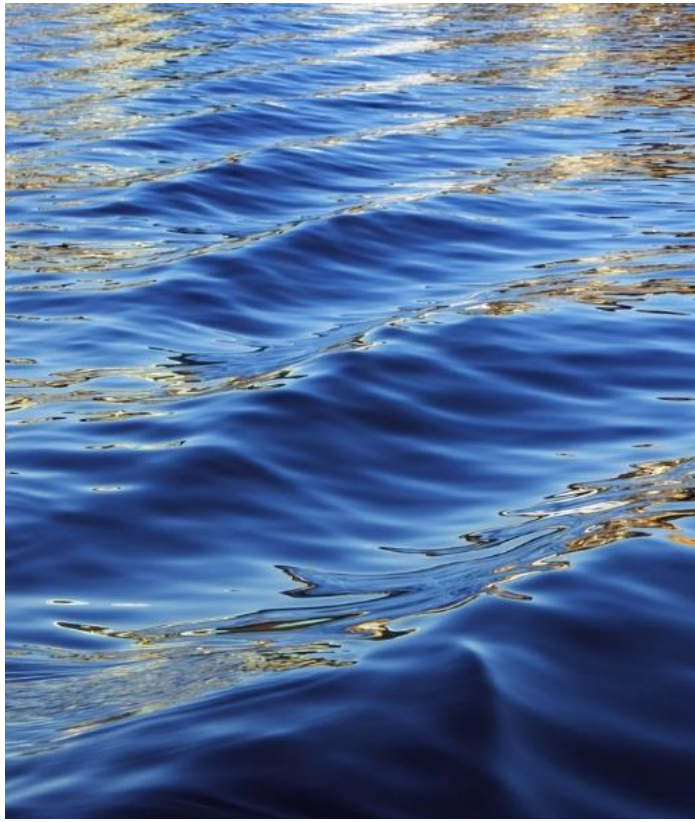
Basic Structure - Stack and Queues

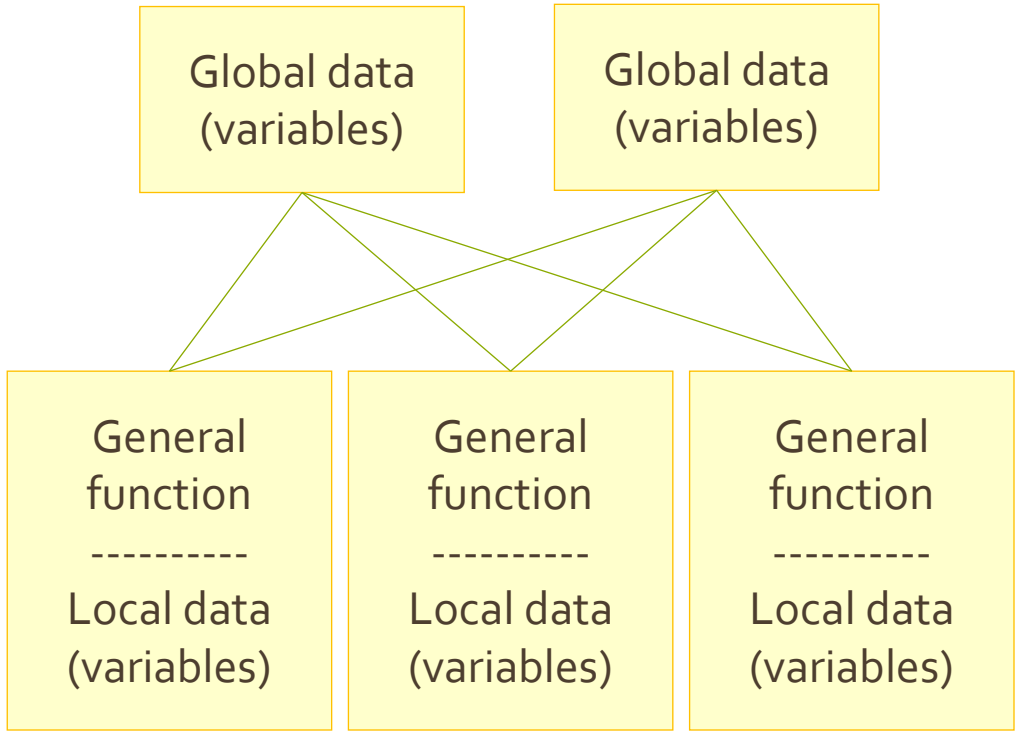
Lecture 5



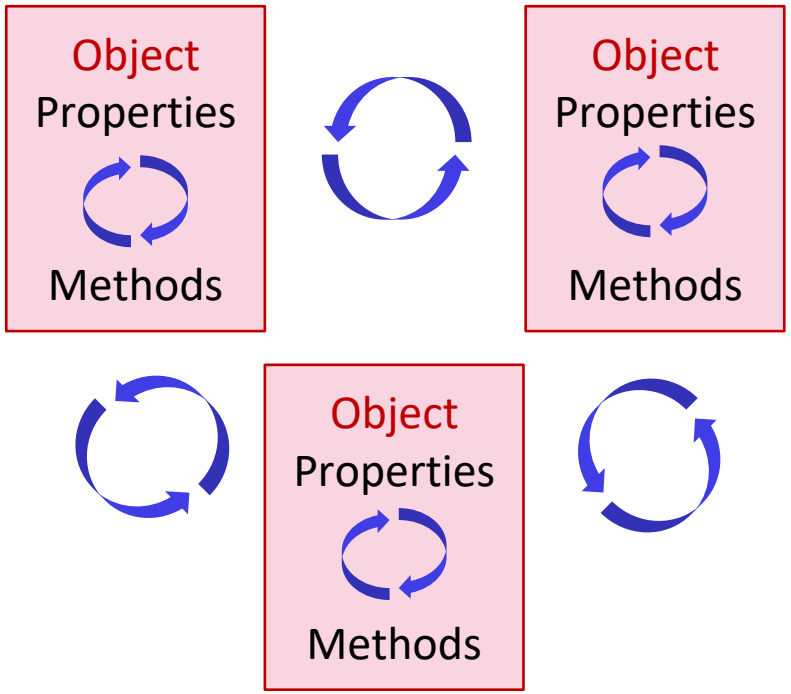
Object-Oriented Programming

As the name implies, the main "actors" in the object-oriented paradigm are called **"objects"**.

Procedural Programming



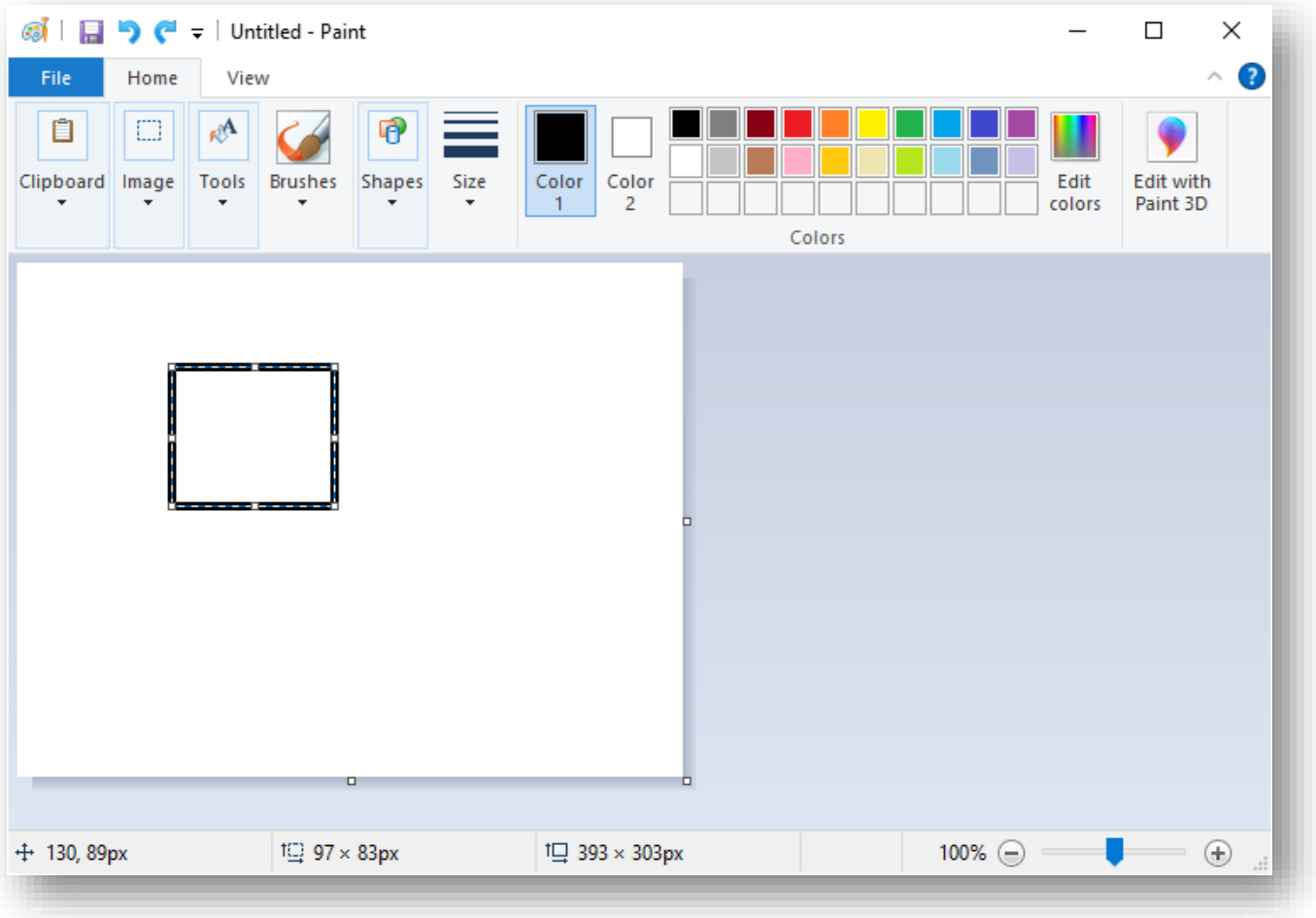
Object-Oriented Programming



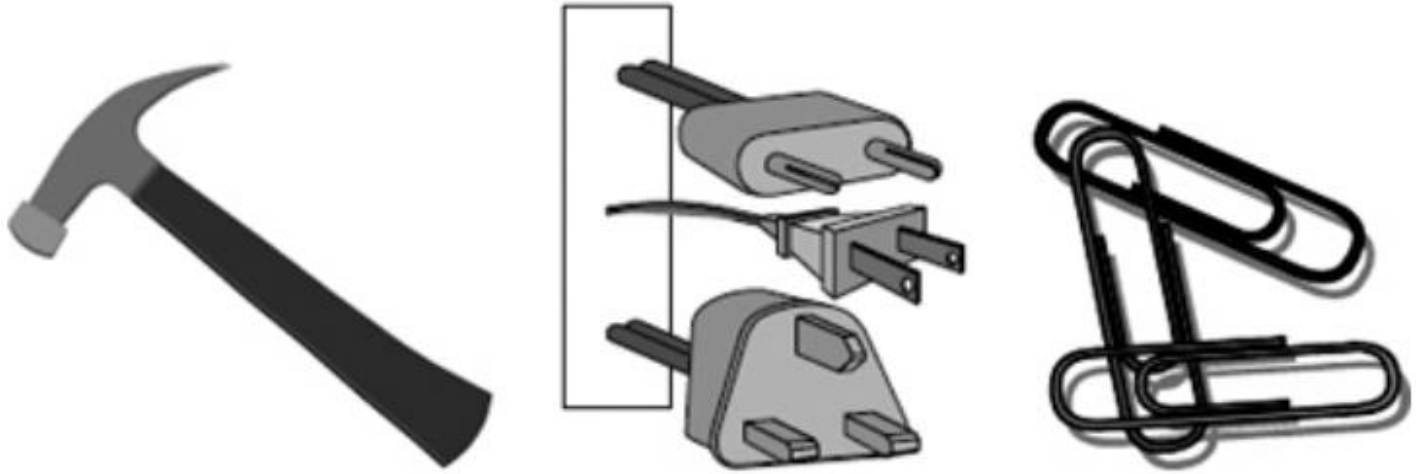
Object-Oriented Programming

Class / Object / Attribute / Method

Rectangle
Point Width Height Stroke Fill
Position Rotate Fill



Object-Oriented Design Goal



Robustness Adaptability Reusability

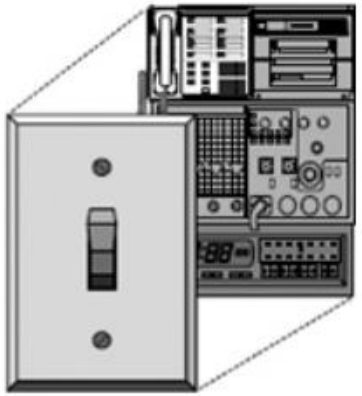
Every good programmer wants to develop software that is correct, which means that a program produces the right output for all the anticipated inputs in the program's application.

Software needs to be able to **evolve over time** in response to changing conditions in its environment. (Hardware / Software)
For example: web browser, search engine, ads

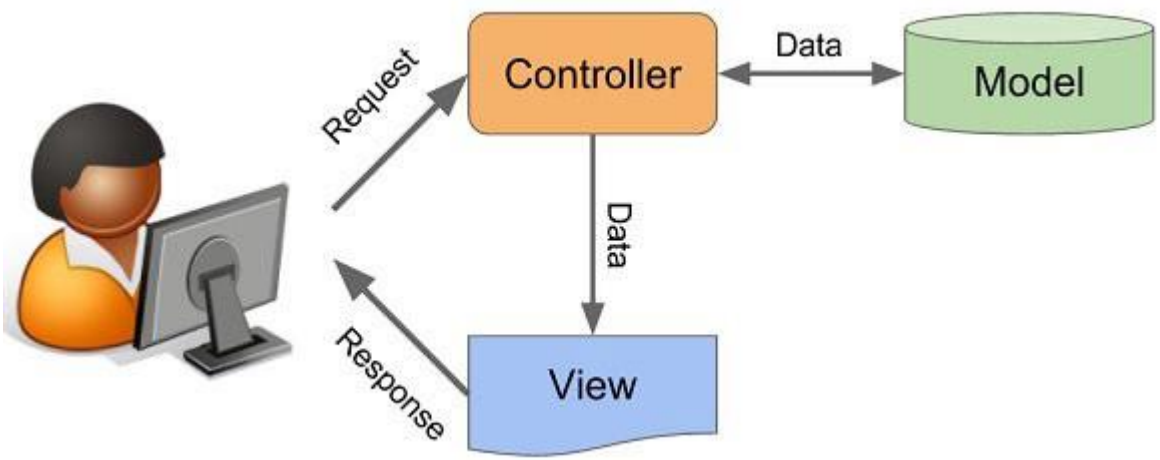
Object-Oriented Design Principle

Modularity

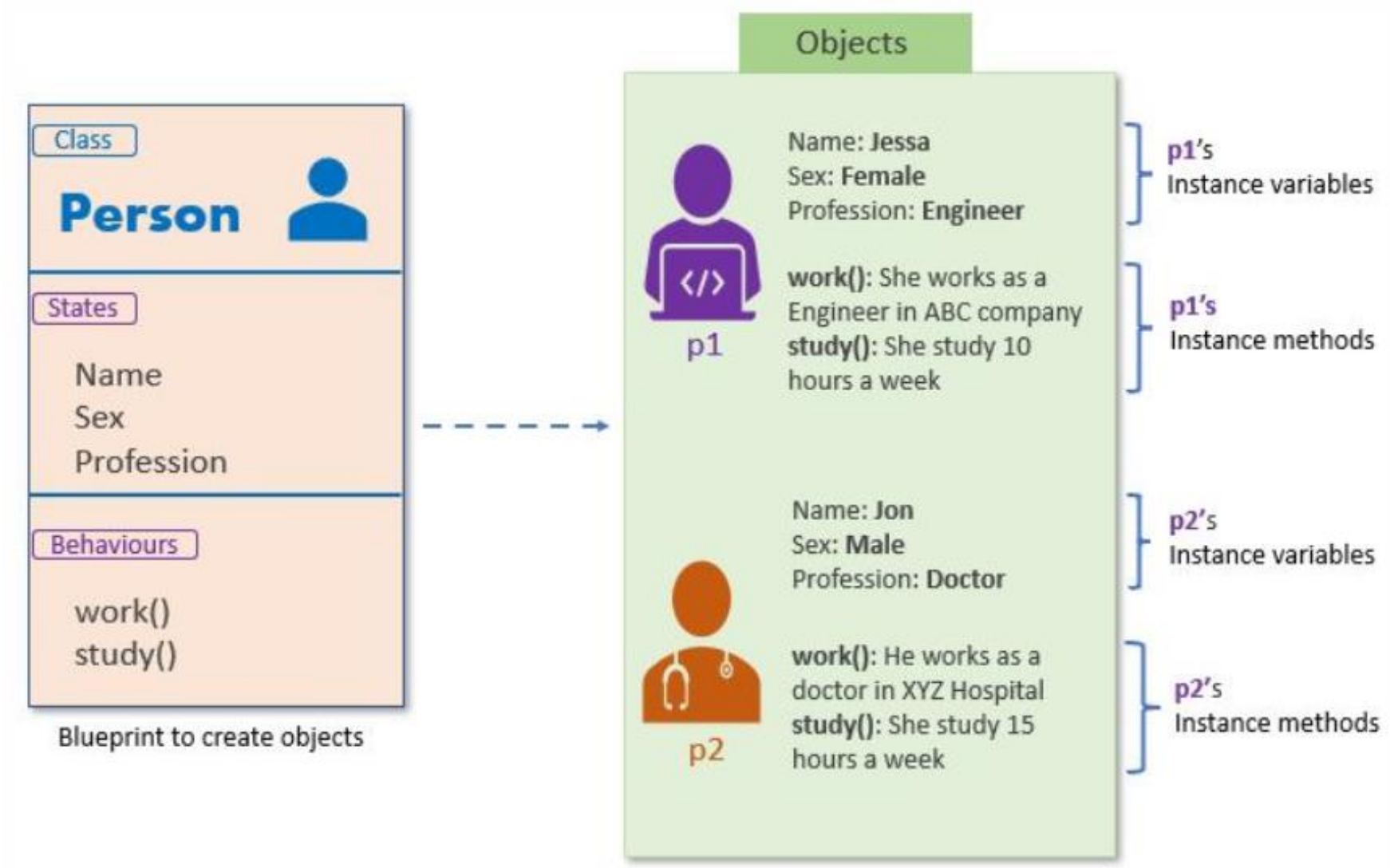
Abstraction



Encapsulation



Object-Oriented ➔ Class & Objects



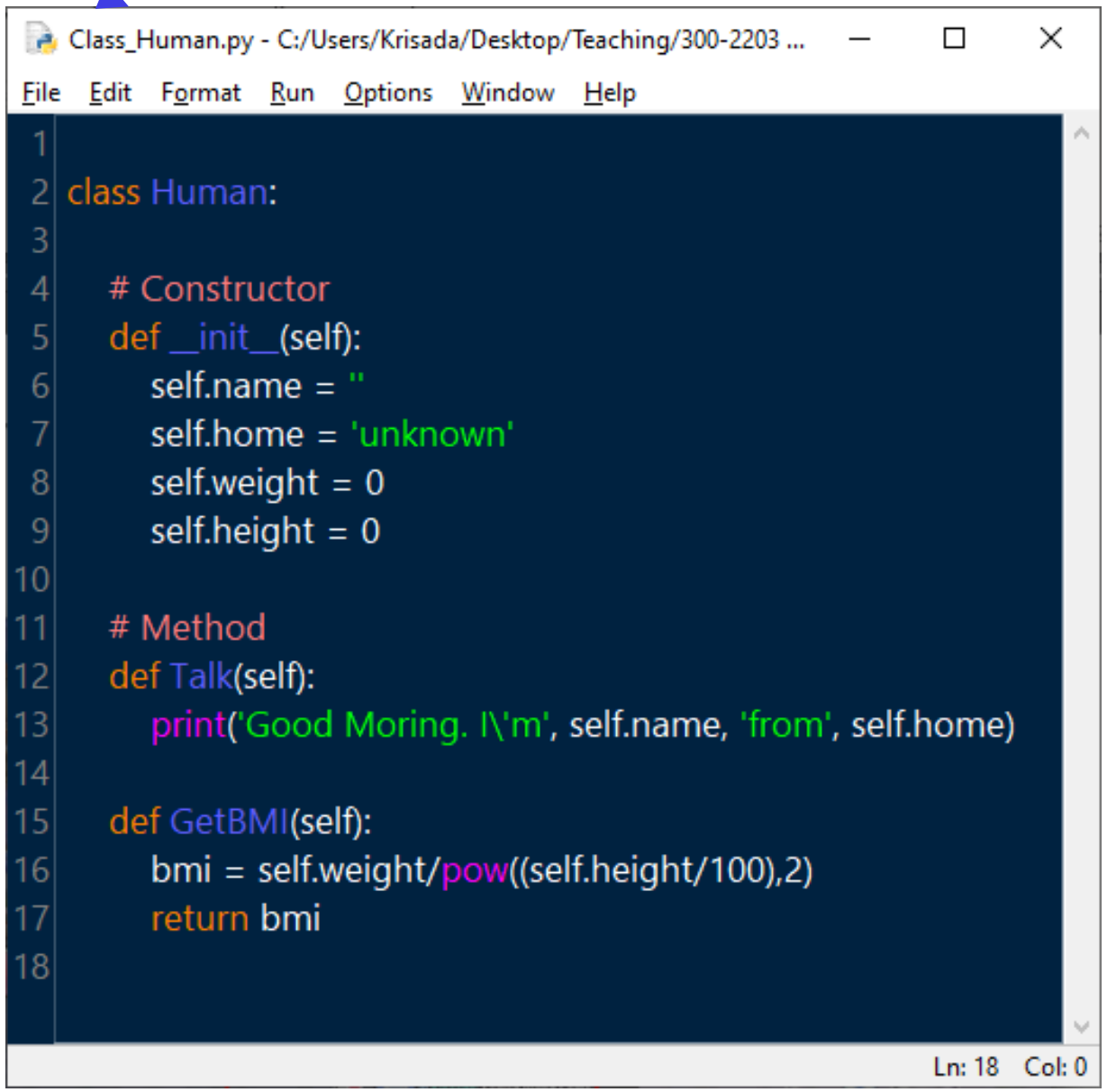
Class in Python

Filename

Class name

Attributes

Method



Class in Python

Import class from another file (Class_Human.py)

Example1_Class.py - C:\Users\...

File Edit Format Run Options Window Help

```
1
2  from Class_Human import Human
3
4  a = Human()
5
6  a.name = 'Krisada'
7  a.home = 'Nonthaburi'
8  a.weight = 100
9  a.height = 180
10
11 a.Talk()
12
13 bmi = a.GetBMI()
14
15 print('BMI: ',bmi)
16
```

Ln: 16 Col: 0

Create class
(Call constructor function)

Assign attribute values

Call function
(method)

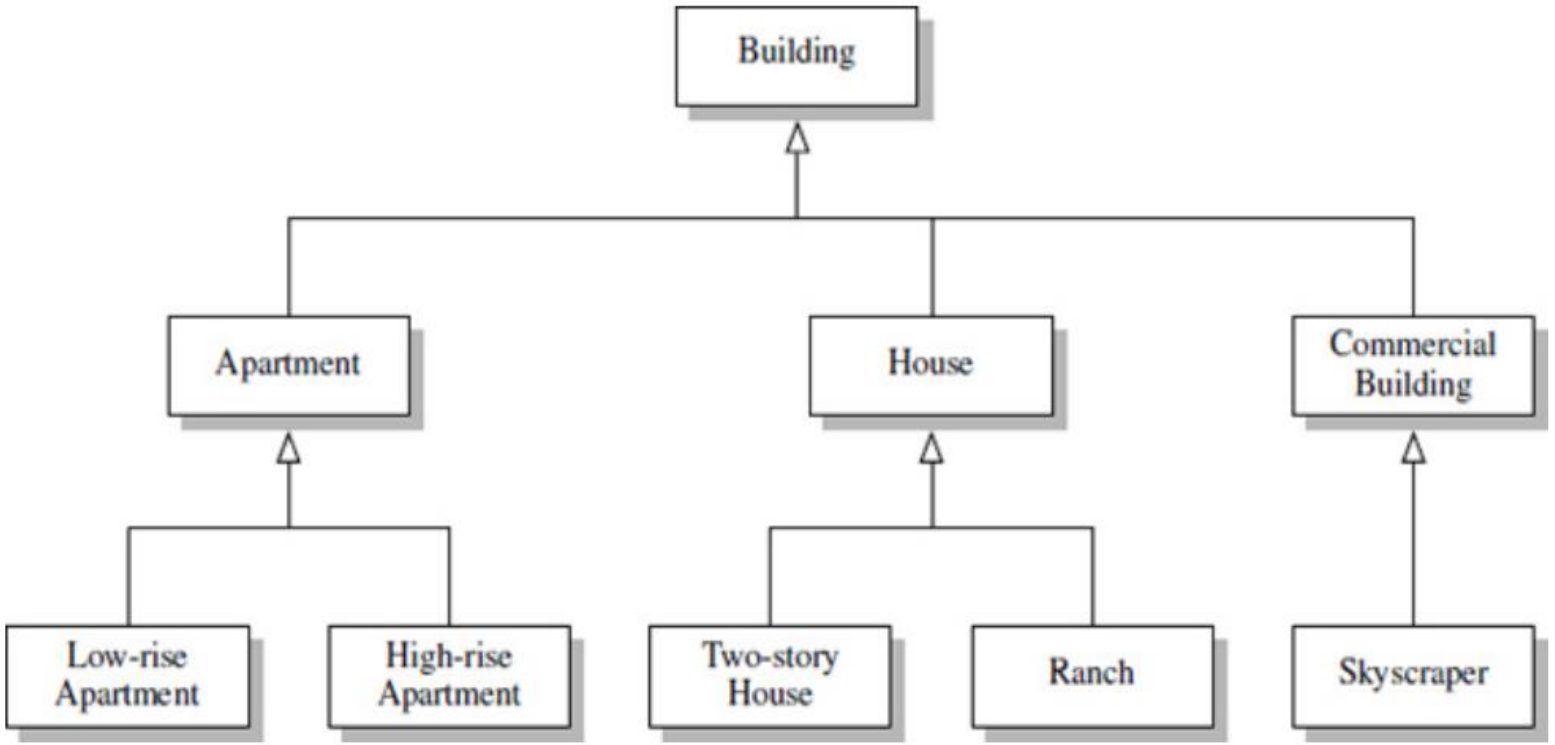
Class_Human.py - C:/Users/Krisada/Desktop/Teaching/300-2203 ...

File Edit Format Run Options Window Help

```
1
2  class Human:
3
4      # Constructor
5      def __init__(self):
6          self.name = ''
7          self.home = 'unknown'
8          self.weight = 0
9          self.height = 0
10
11     # Method
12     def Talk(self):
13         print('Good Moring. I\'m', self.name, 'from', self.home)
14
15     def GetBMI(self):
16         bmi = self.weight/pow((self.height/100),2)
17         return bmi
18
```

Ln: 18 Col: 0

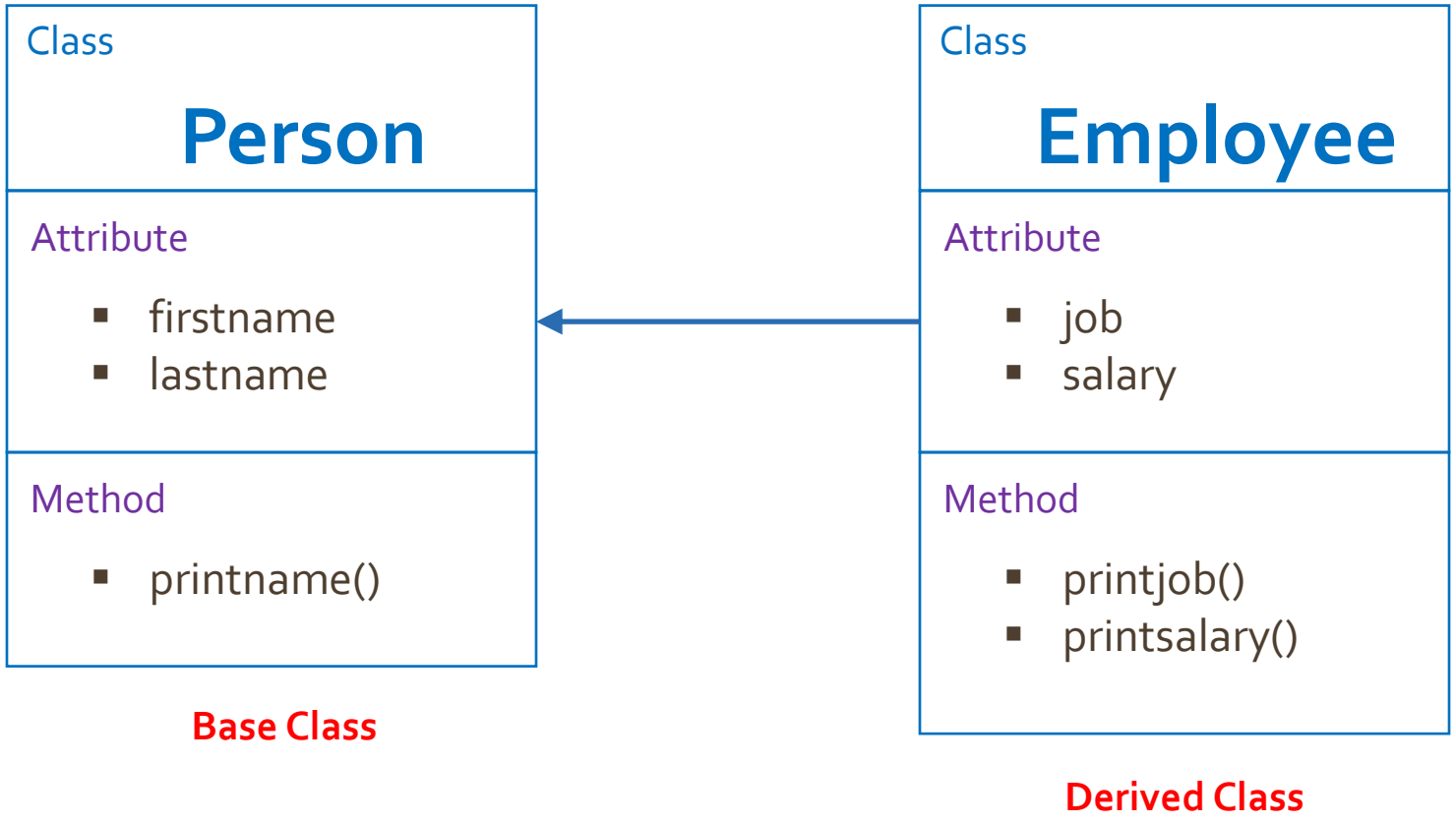
Class in Python : Inheritance (การสืบทอด)



Base Class (Superclass)

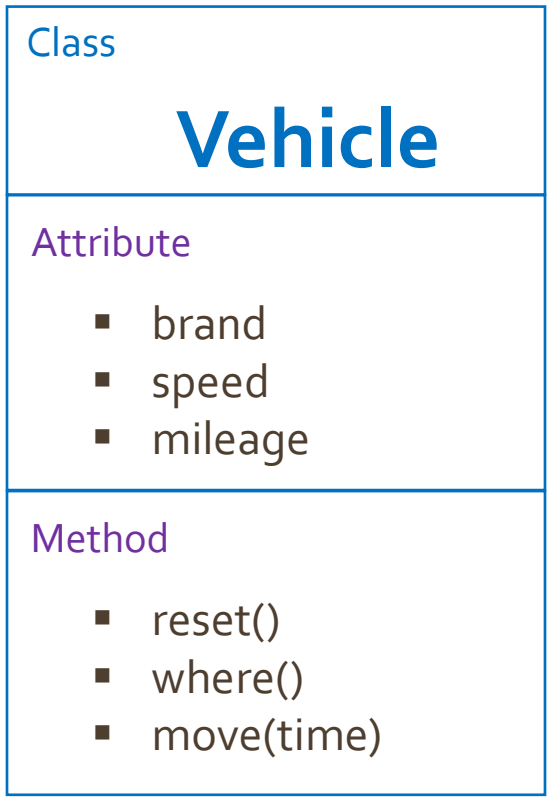
Derived Class (Subclass)

Class in Python : Inheritance (การสืบทอด)



```python
# Parent Class: Person
class Person:
    def __init__(self, firstname, lastname):
        self.firstname = firstname
        self.lastname = lastname

    def printname(self):
        print("name: ", self.firstname, self.lastname)

# Child Class: Employee
class Employee(Person):
    def __init__(self, firstname, lastname, job, salary):
        super().__init__(firstname, lastname)
        self.job = job
        self.salary = salary

    def printjob(self):
        print("job: ", self.job)

    def printsalary(self):
        print("salary: ", self.salary)
```

Exercise #1



เขียน Class ยานพาหนะ (vehicle) ที่ประกอบไปด้วย Attribute คือ ชนิดของรุ่น (brand), ความเร็ว (speed) และระยะทางสะสม (mileage)

- สร้าง Function reset() สำหรับใช้ในการกำหนดให้ค่า mileage เป็น 0
- สร้าง Function where() สำหรับแสดงระยะทางปัจจุบันที่วิ่งไปได้
- สร้าง Function move(time) สำหรับกำหนดระยะทางการเคลื่อนที่ และเพิ่มตัวเลข time เข้าไปในตัวแปร mileage (ความเร็ว x เวลา)

Exercise #1

- สร้าง Object ชื่อ Car
- กำหนดให้ความเร็วของ Car (speed) เท่ากับ 10
- ทดลองใช้ Function move กำหนด time = 5
- แสดงระยะทางสะสมที่ Car เคลื่อนที่ไปได้

- ทำอย่างไรจึงจะสามารถสร้าง Object “Vehicle” โดยไม่ต้องระบุ Brand เมื่อทำการสร้าง Object
- ทดลองเคลื่อนที่รถต่อไปด้วยระยะเวลาอีก 5 และทำการตรวจสอบระยะทางสะสม

Exercise #1 – step2:

เขียน Class รถประจำทาง (Bus) ที่สืบทอด (Inherit) มาจาก Vehicle โดยให้เพิ่ม Attribute เข้าไป คือ หมายเลขรถ (bus_number) และ จำนวนที่นั่ง (seating_capacity)

- กำหนดหมายเลขรถ จำนวนที่นั่ง (A01X, 15 ที่นั่ง)
- สั่งให้รถเคลื่อนที่ไปด้วยความเร็ว 30km/hrs. เป็นระยะเวลา 2 hrs.
- สั่ง print ค่าหมายเลขรถ และระยะทางสะสม

Exercise #1 – step3:

เพิ่มอัตราค่าโดยสารต่อระยะทาง 1 km (fare_rate) และเพิ่ม Function ในการคำนวณค่าโดยสารตามระยะทาง (ตั้งชื่อ function ว่า fare)

- สร้าง Object รถ Bus จำนวน 2 คัน
- คันที่ 1 Brand Benz เคลื่อนที่ความเร็ว 30km/hrs. ระยะเวลา 2 hrs. อัตราค่าโดยสาร 1 km / 5 baht
- คันที่ 2 Brand Benz เคลื่อนที่ความเร็ว 10km/hrs. ระยะเวลา 5 hrs. อัตราค่าโดยสาร 1 km / 7 baht
- สั่งคำนวณค่าโดยสารตามระยะทางที่เกิดขึ้นและแสดงผล ของรถทั้ง 2 คัน

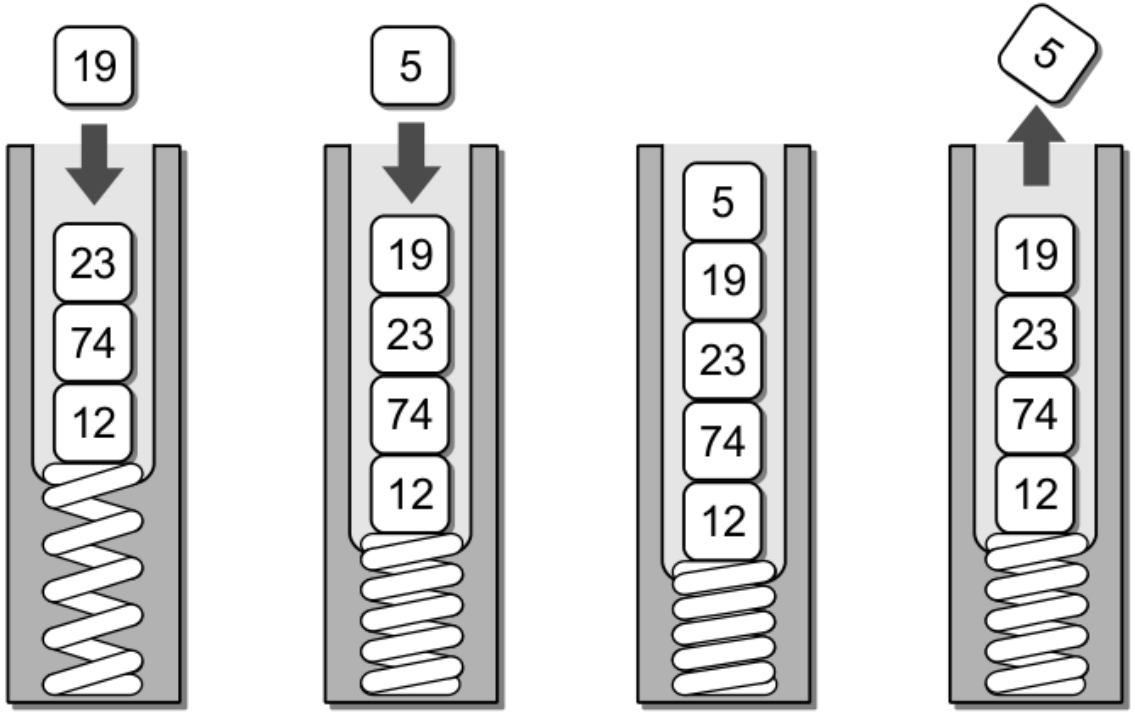
Abstract Data Type (ADT) in Data Structures

We all know that In Today's world there is lots of software running in the market, But do you ever think how they work. there are many things behind that Software like Data Structures, algorithms, codes, etc. In this lesson we are going to learn about Abstract Data Type (ADT) in Data Structures.

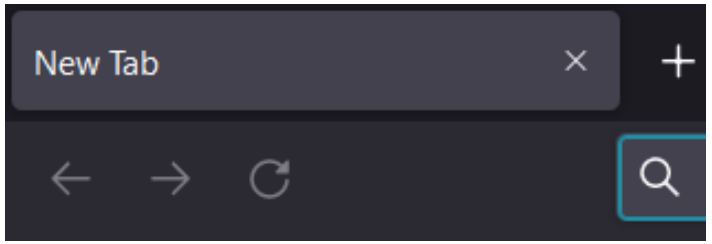
- **Abstract data type (ADT)**
  - An ADT is composed of
    - A collection of data
    - A set of operation of data
  - Specification of an ADT indicate
    - What the ADT operation do, not how to implement them

Stacks

A stack is a collection of objects that are inserted and removed according to the **last-in, first-out (LIFO) principle**. A user may insert objects into a stack at any time but may only access or remove the most recently inserted object that remains (at the so-called "top" of the stack).



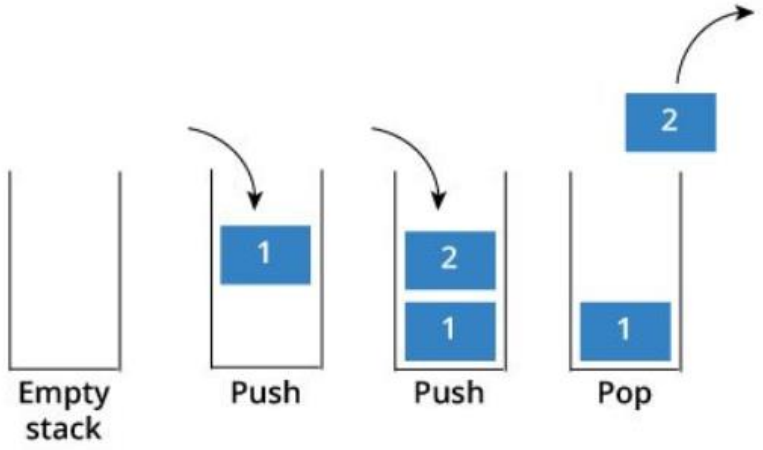
Example



Web browsers store the addresses of recently visited sites in a stack

Stacks

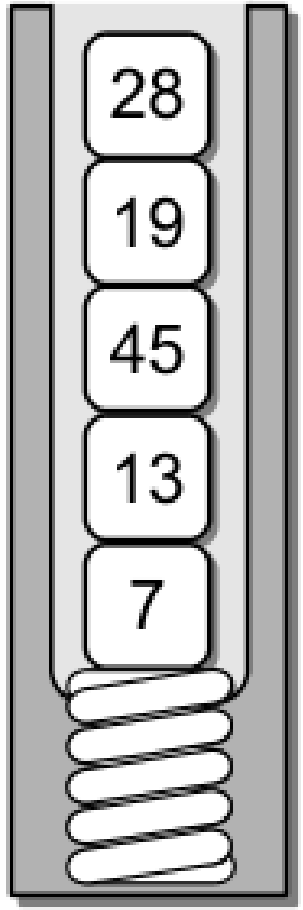
Define ADT



- `Stack()`: Creates a new empty stack.
- `isEmpty()`: Returns a boolean value indicating if the stack is empty.
- *`length`*`()`: Returns the number of items in the stack.
- `pop()`: Removes and returns the top item of the stack, if the stack is not empty. Items cannot be popped from an empty stack. The next item on the stack becomes the new top item.
- `peek()`: Returns a reference to the item on top of a non-empty stack without removing it. Peeking, which cannot be done on an empty stack, does not modify the stack contents.
- `push(item)`: Adds the given `item` to the top of the stack.

Stacks

```
PROMPT = "Enter an int value (<0 to end):"
myStack = Stack()
value = int(input( PROMPT ))
while value >= 0 :
  myStack.push( value )
  value = int(input( PROMPT ))

while not myStack.isEmpty() :
  value = myStack.pop()
  print( value )
```



Suppose the user enters the following values, one at a time:

```
7   13   45   19   28   -1
```

Stacks

Stack.py > ...

```
1   class Stack:
2
3       def __init__(self):
4           self._top = None
5           self._size = 0
6
7       def isEmpty(self):
8           return self._top is None
9
10      def __len__(self):
11          return self._size
12
13      def peek(self):
14          assert not self.isEmpty(), "Cannot peek at an empty stack"
15          return self._top.item
16
17      def pop(self):
18          assert not self.isEmpty(), "Cannot pop from an empty stack"
19          node = self._top
20          self._top = self._top.next
21          self._size = self._size - 1
22          return node.item
```

Stacks (ต่อ)

```
23
24      def push(self, item):
25          self._top = _StackNode(item, self._top)
26          self._size = self._size + 1
27
28  class _StackNode:
29      def __init__(self, item, link):
30          self.item = item
31          self.next = link
32
```

Assert คืออะไร?

Write a message if the condition is False:

```
x = "hello"

#if condition returns False, AssertionError is raised:
assert x == "goodbye", "x should be 'hello'"
```

Stacks

example1_stack.py > ...

```python
from Stack import *

print("\n\n === Push Value === \n\n")

PROMPT = "Enter an int value (<0 to end): "
myStack = Stack()

value = int(input( PROMPT ))
while value >= 0:
    myStack.push( value )
    value = int(input ( PROMPT ))

print("\n\n === Pop & Print === \n\n")

while not myStack.isEmpty():
    value = myStack.pop()
    print( value )

```

Exercise #2

สร้างโปรแกรม Matching Parentheses โดยใช้ Stack

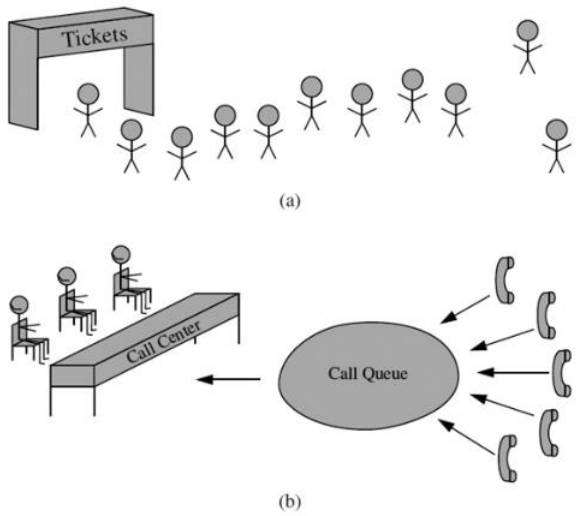
- สร้าง Function สำหรับรับค่าตัวอักษร เช่น (5 + 3) x {(30 – 2) + 7}
- ทำการตรวจสอบว่า ผู้ใช้งานมีการใส่เครื่องหมาย ( ), { } ถูกต้องหรือไม่
  - ❖ ถ้าครบคู่ ถูกต้อง ให้พิมพ์คำว่า Correct Format
  - ❖ ถ้าไม่ครบคู่ ไม่ถูกต้อง ให้พิมพ์คำว่า Incorrect Format

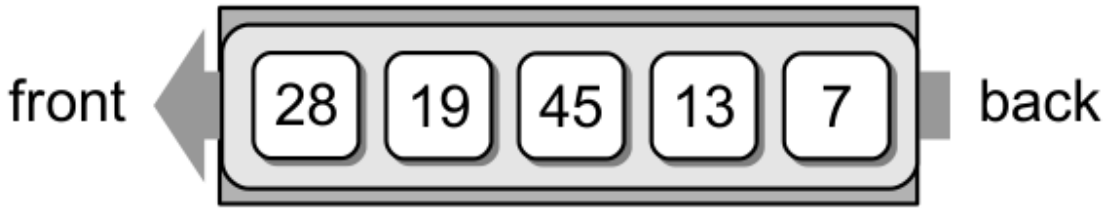
Queues

The term queue is commonly defined to be a line of people waiting to be served like those you would encounter at many business establishments. Each person is served based on their position within the queue. Thus, the next person to be served is the first in line.

A queue is a specialized list with a limited number of operations in which items can only be added to one end and removed from the other.

A queue is also known as a **first-in, first-out (FIFO)** list.

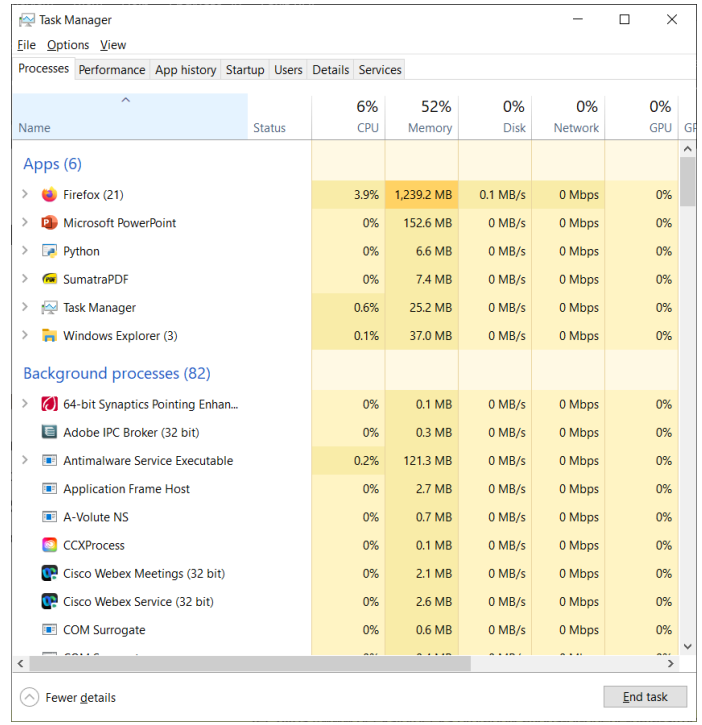




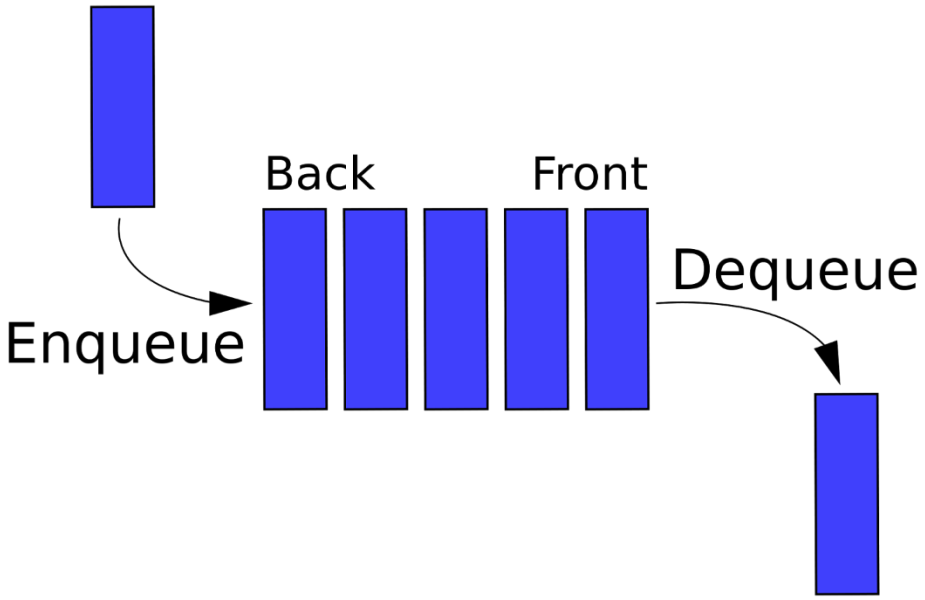
Queues

Application of Queue Data Structure

1. When a resource is shared among multiple consumers. (CPU scheduling, Disk Scheduling)
2. When data is transferred asynchronously between two processes.
3. In Operating System (OS): Spooling in printers, Buffer for devices like keyboard
4. In Network: Queues in routers/ switches, Mail Queues

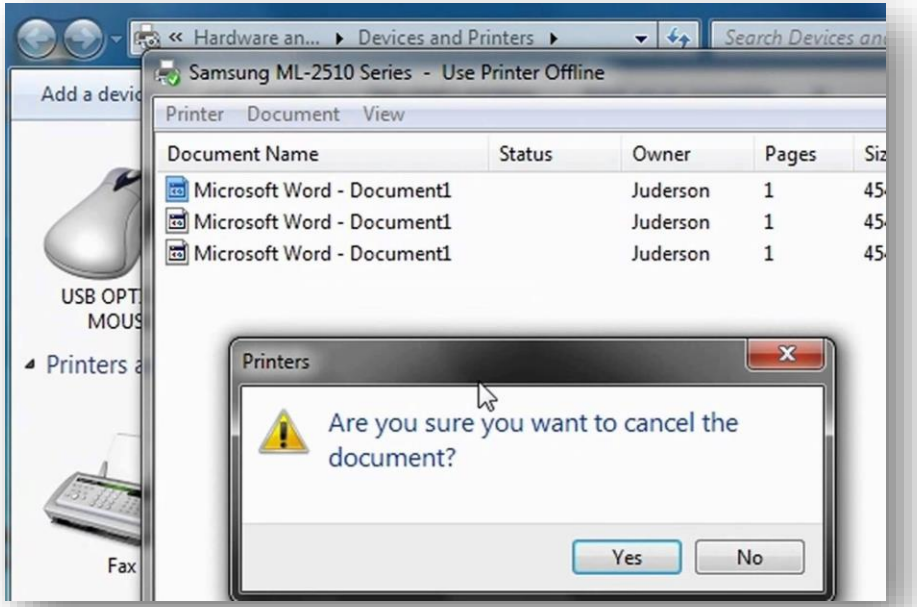


Queues



First In – First Out

Last In – Last Out



Turn-based game
(Active Time Battle)

Queues

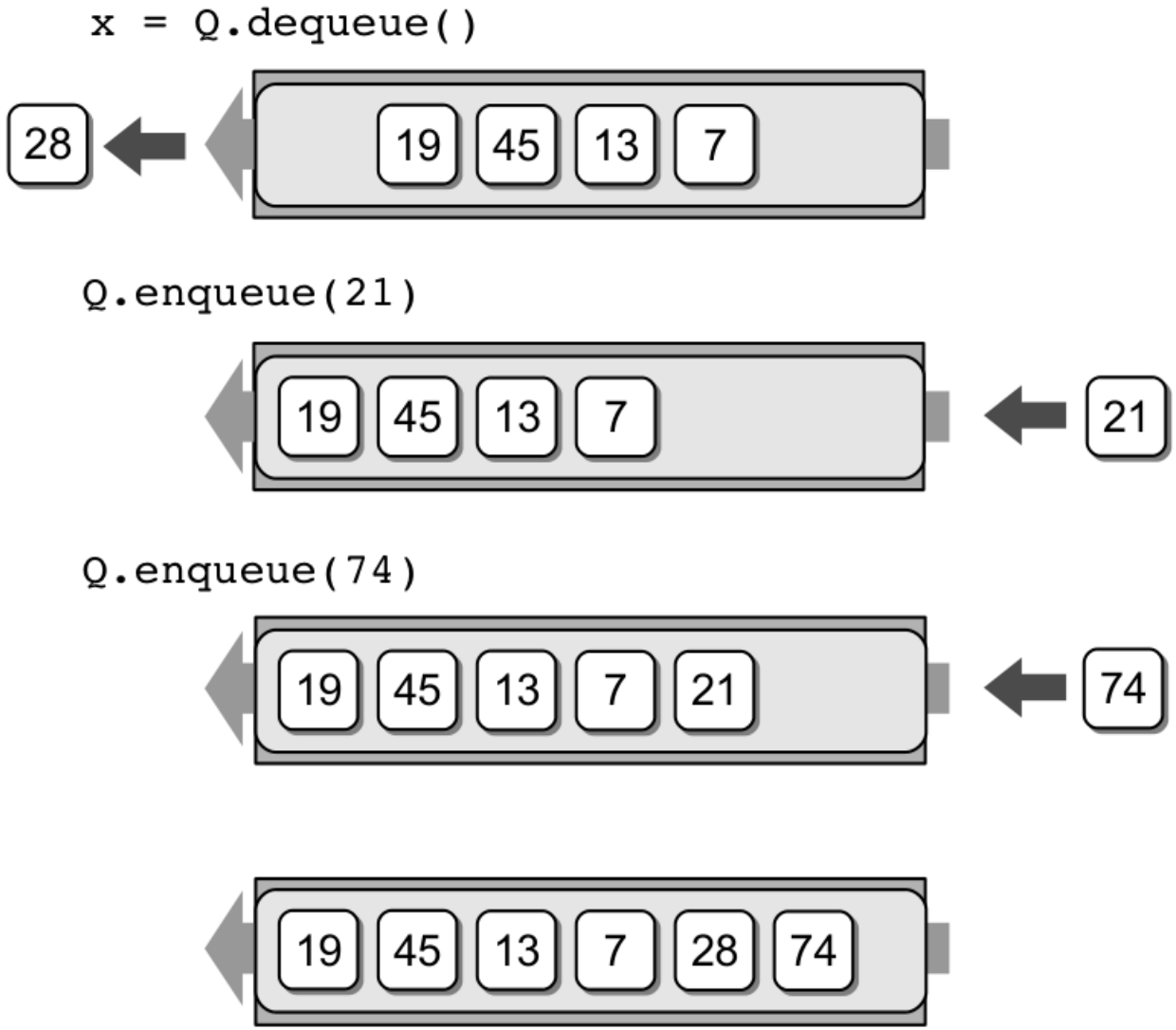
Define **Queue ADT**

A *queue* is a data structure that a linear collection of items in which access is restricted to a first-in first-out basis. New items are inserted at the back and existing items are removed from the front. The items are maintained in the order in which they are added to the structure.

- `Queue()`: Creates a new empty queue, which is a queue containing no items.
- `isEmpty()`: Returns a boolean value indicating whether the queue is empty.
- *`length`*`()`: Returns the number of items currently in the queue.
- `enqueue(item)`: Adds the given item to the back of the queue.
- `dequeue()`: Removes and returns the front item from the queue. An item cannot be dequeued from an empty queue.

Queues

```
Q = Queue()
Q.enqueue( 28 )
Q.enqueue( 19 )
Q.enqueue( 45 )
Q.enqueue( 13 )
Q.enqueue( 7 )
```

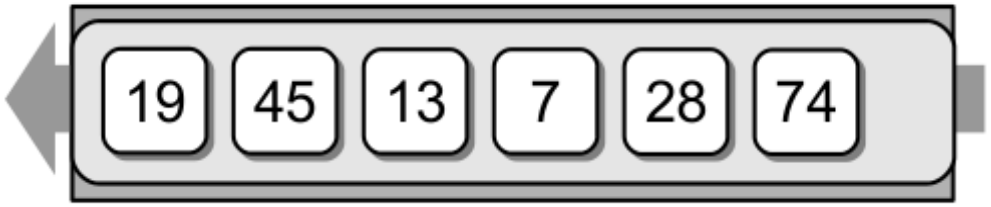


Queues

Queue.py > ...

```python
class Queue:
    def __init__(self):
        self._qList = list()

    def isEmpty(self):
        return len(self) == 0

    def __len__(self):
        return len(self._qList)

    def enqueue(self, item):
        self._qList.append(item)

    def dequeue(self):
        assert not self.isEmpty(), "Cannot dequeue from an empty queue."
        return self._qList.pop(0)

```

Queues

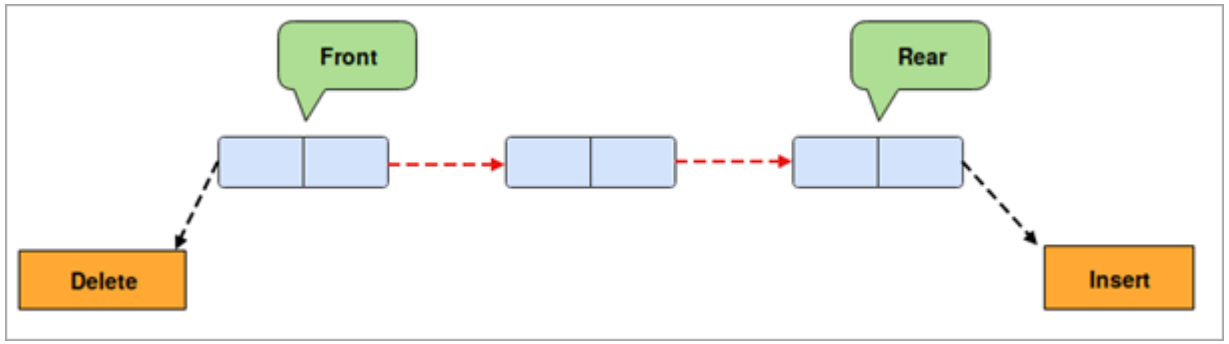



```
74
28
7
13
45
19
```

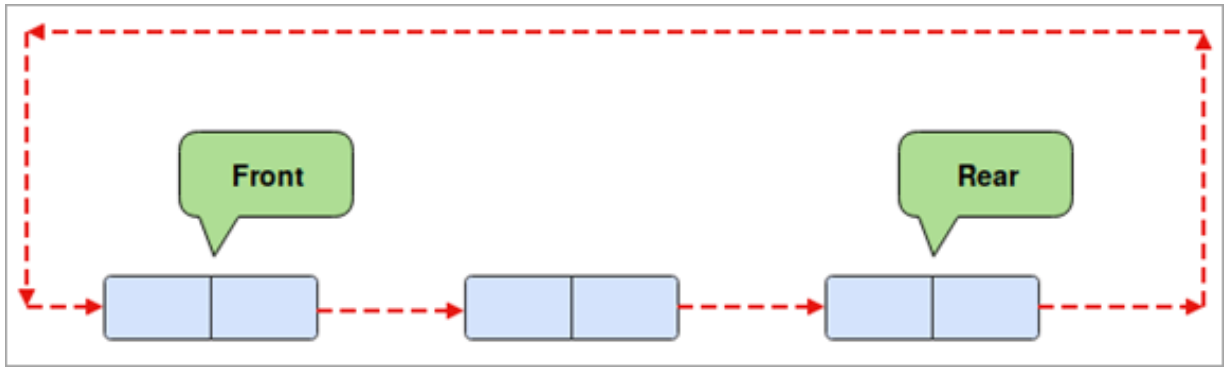
example4_queue.py > ...

```python
from Queue import *

Q = Queue()

Q.enqueue(74)
Q.enqueue(28)
Q.enqueue(7)
Q.enqueue(13)
Q.enqueue(45)
Q.enqueue(19)

print(Q.dequeue())
print(Q.dequeue())
print(Q.dequeue())
print(Q.dequeue())
print(Q.dequeue())
print(Q.dequeue())

```

Queues



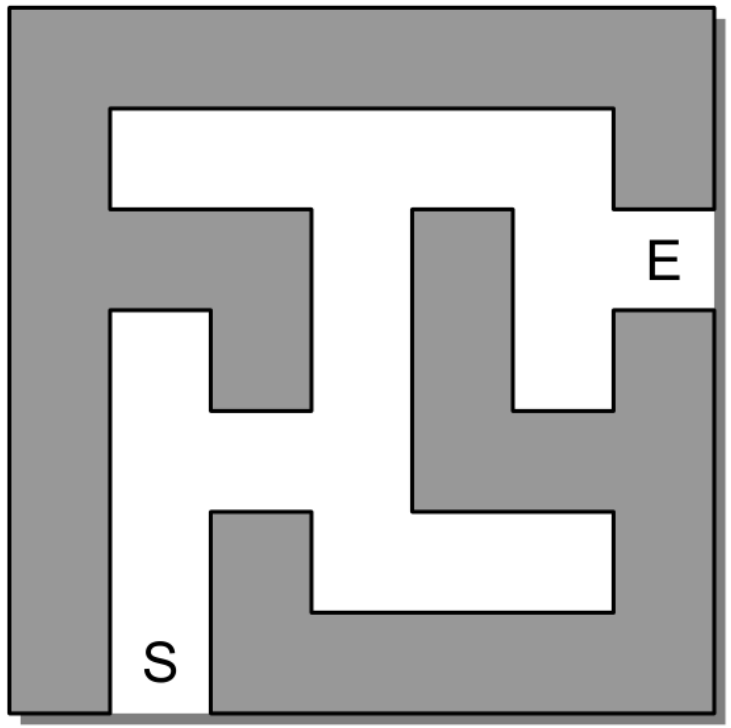
Simple queue

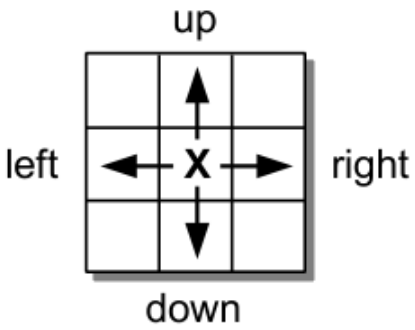


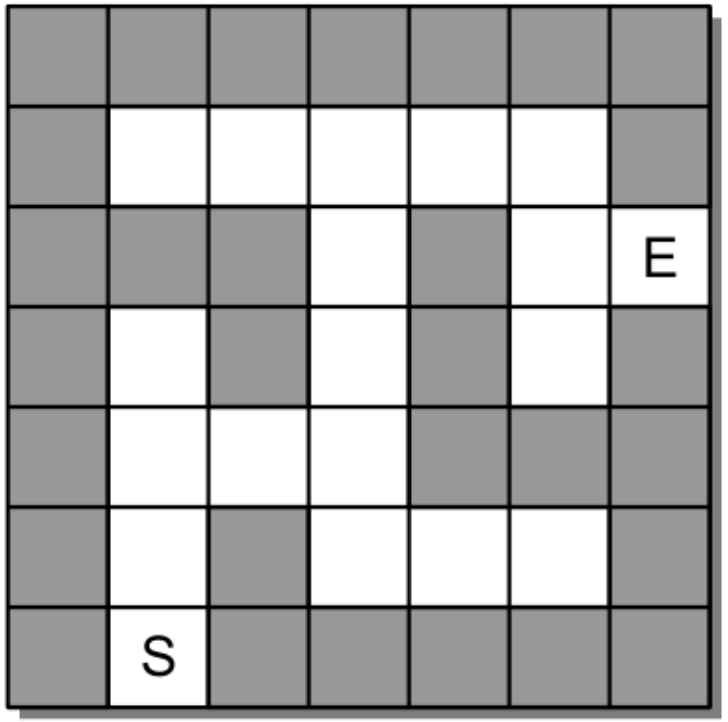
Circular queue

Solving a Maze

A classic example of an application that requires the use of a stack is the problem of finding a path through a maze.

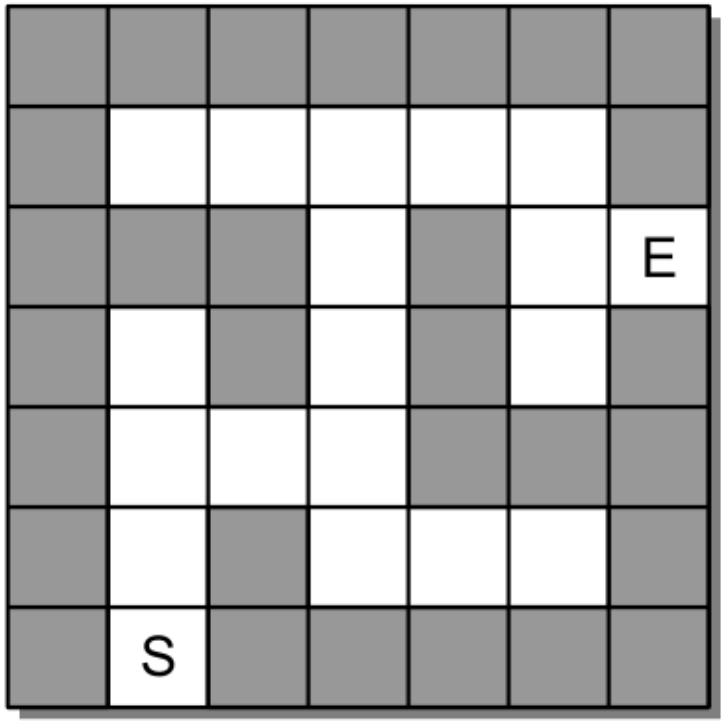






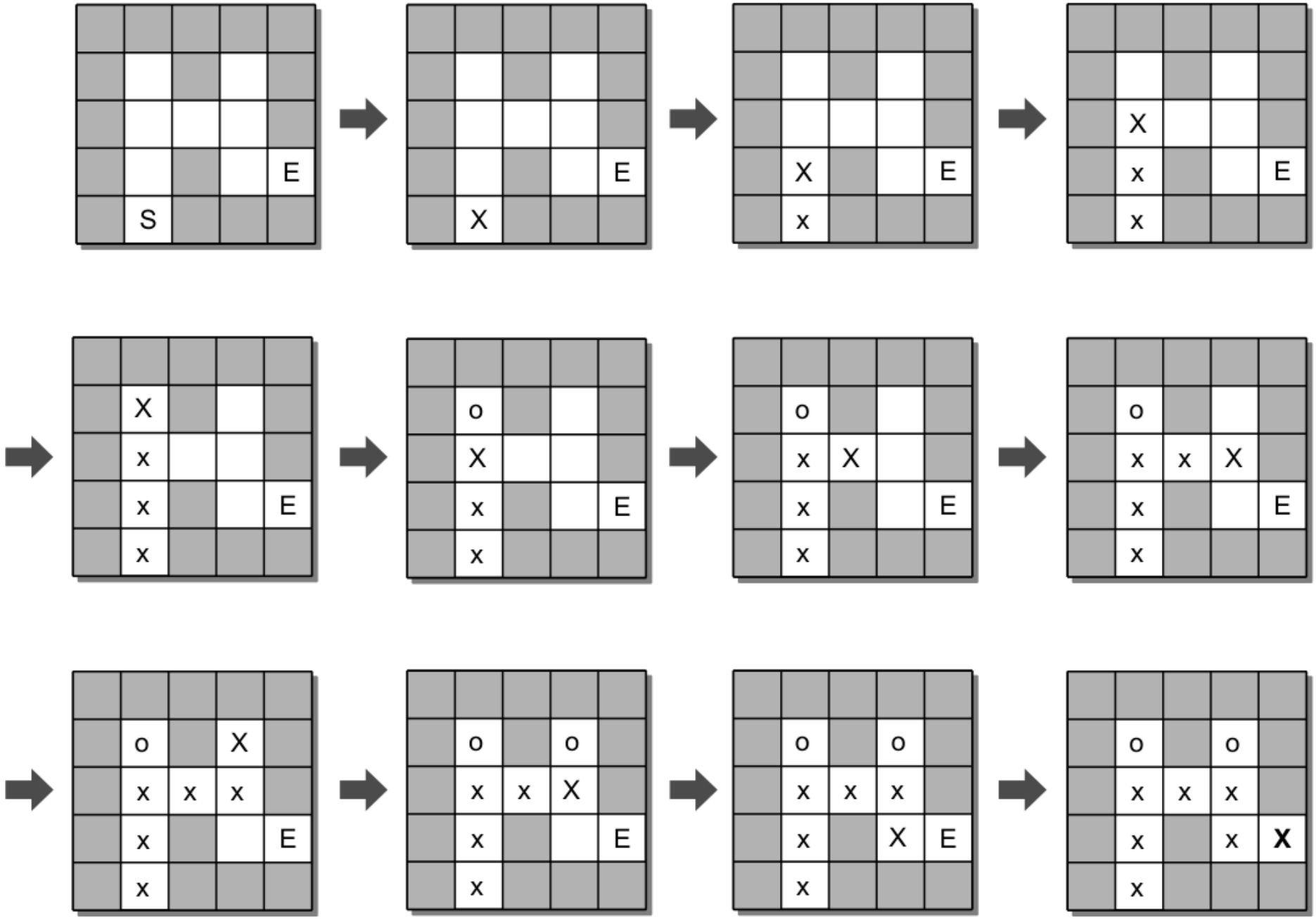
Solving a Maze

Backtracking



The most basic problem-solving technique in computer science is the ***brute-force*** method.

It involves searching for a solution to a given problem by systematically trying all possible candidates until either a solution is found, or it can be determined there is no solution.


S
E
X
x
E
X
x
x
E
X
x
x
E
o
X
x
x
E
o
x
X
x
x
E
o
x
x
X
x
x
E
o
X
x
x
x
x
E
o
o
x
x
X
x
x
E
o
o
x
x
x
x
X
E
o
o
x
x
x
x
x
X
E

Example

- maze structure
- print function

Expected Result

List of coordinates from starting point to ending

```
3   maze = [
4       ["X", "X", "X", "X", "X", "X", "X"],
5       ["X", " ", " ", " ", "X", " ", "E"],
6       ["X", " ", "X", " ", "X", " ", "X"],
7       ["X", " ", "X", " ", "X", " ", "X"],
8       ["X", " ", "X", " ", " ", " ", "X"],
9       ["X", "S", "X", "X", "X", "X", "X"],
10  ]
11
12  for row in maze:
13      for col in row:
14          print(col," ", end="")
15      print("")
16
17  print("\nStart Position: ", maze[5][1])
18
19  print("\n>> Left: \t", maze[5][0])
20  print(">> Right: \t", maze[5][2])
21  print(">> Top: \t", maze[4][1])
22
```

Exercise #3

ให้นักศึกษาเขียนโปรแกรมทำให้หุ่นยนต์เคลื่อนที่ออกจากเขาวงกตแบบอัตโนมัติ

- *พัฒนาโปรแกรมบน GitHub ของตัวเอง ตั้งชื่อ Repository ว่า maze_รหัสนักศึกษา*
- *แนะนำให้ใช้ Source Code ตัวอย่างที่กำหนดให้ เป็น Project เริ่มต้น*
- *เมื่อมีการเปลี่ยนแปลงแผนที่ หุ่นยนต์ ต้องสามารถออกจากเขาวงกตได้*