# Reinvent the .wheel

20210413-2

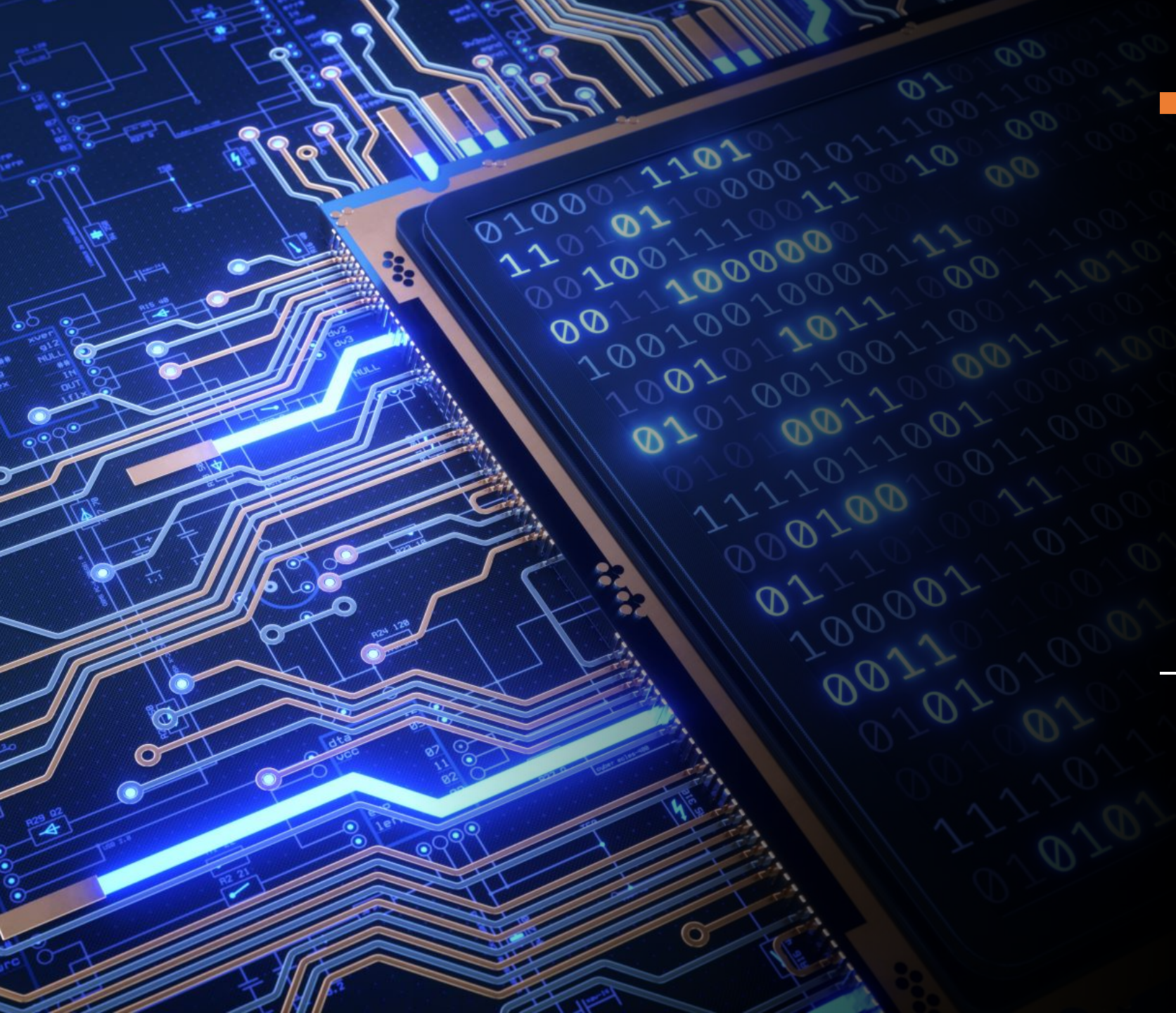
# Download

Anaconda: https://www.anaconda.com/products/individual

Visual studio code: https://code.visualstudio.com/

AOB

Contact Information



# Logic

# Logic

- True
- False
- Logical Operator: AND (&& / .) , OR (|| / + ), NOT (~)

# Truth Table

| condition 1 (e.g., X) | condition 2 (e.g., Y) | NOT X ( ~ X ) | X AND Y ( X && Y ) | X OR Y ( X \|\| Y ) |
|---|---|---|---|---|
| false | false | | | |
| false | true | | | |
| true | false | | | |
| true | true | | | |

# Truth Table

| condition 1 (e.g., X) | condition 2 (e.g., Y) | NOT X ( ~ X ) | X AND Y ( X && Y ) | X OR Y ( X \|\| Y ) |
|---|---|---|---|---|
| false | false | true | false | false |
| false | true | true | false | true |
| true | false | false | false | true |
| true | true | false | true | true |

# DeMorgan's Laws

$$\overline{A \cdot B} \equiv \overline{A} + \overline{B}$$

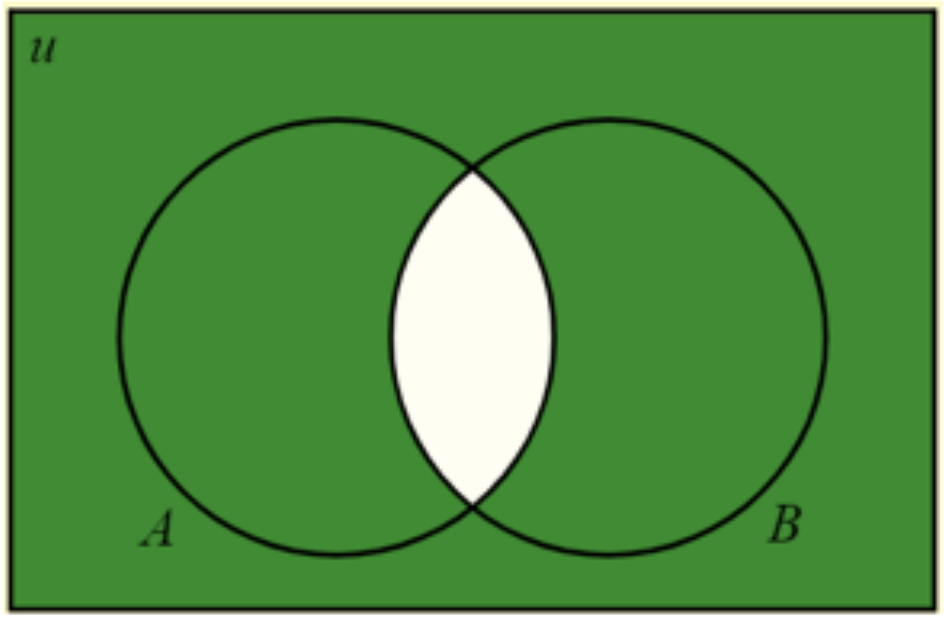
NOT(A || B) is Equivalent to (NOT(A) AND NOT(B))

and

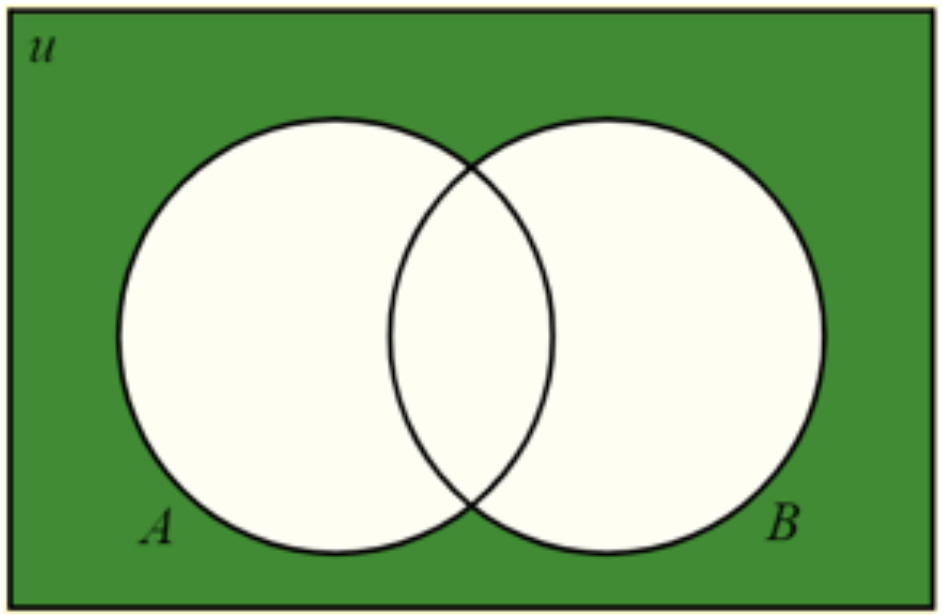
$$\overline{A + B} \equiv \overline{A} \cdot \overline{B},$$

NOT(A || B) is Equivalent to (NOT(A) AND NOT(B))

# De Morgan's Law




$$(A \cap B)' = A' \cup B'$$




$$(A \cup B)' = A' \cap B'$$

# Exercise

- Truth table

$$\overline{A \cdot B} \equiv \overline{A} + \overline{B}$$

and

$$\overline{A + B} \equiv \overline{A} \cdot \overline{B},$$


Coding

# Last Month

- Looping
- Python Env Setup

# Variable

- Variables are containers for storing data values.
- Python has no command for declaring a variable.
- Variable names are case-sensitive.

- ```
  x = 5
  y = "John"
  print(x)
  print(y)
  ```
  ```

# Variable Names

- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _ )
- Variable names are case-sensitive (age, Age and AGE are three different variables)

# Variable - Casting

- x = str(3)    # x will be '3'
  y = int(3)    # y will be 3
  z = float(3)  # z will be 3.0

# Variable Get Type

- x = 5
  y = "John"
  print(type(x))
  print(type(y))

# Python Data Types

| | |
|---|---|
| Text Type: | str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | dict |
| Set Types: | set, frozenset |
| Boolean Type: | bool |
| Binary Types: | bytes, bytearray, memoryview |

# Common use data type example

| | |
|---|---|
| `x = "Hello World"` | str |
| `x = 20` | int |
| `x = 20.5` | float |
| `x = 1j` | complex |
| `x = ["apple", "banana", "cherry"]` | list |
| `x = ("apple", "banana", "cherry")` | tuple |
| `x = range(6)` | range |
| `x = {"name" : "John", "age" : 36}` | dict |
| `x = {"apple", "banana", "cherry"}` | set |
| `x = True` | bool |

# Data Type: Number

- x = 1    # int
  y = 2.8  # float
  z = 1j   # complex

# Strings

- Strings in python are surrounded by either single quotation marks, or double quotation marks.
- 'hello' is the same as "hello".

- print("Hello")
  print('Hello')

# String functions (1)

- for x in "banana":
  print(x)

- a = "Hello, World!"
  print(len(a))

# String functions (2)

- txt = "The best things in life are free!"
  print("free" in txt)

- txt = "The best things in life are free!"
  print(expensive" not in txt)

- b = "Hello, World!"
  print(b[2:5])

# String functions (3)

- a = "Hello, World!"
  print(a.replace("H", "J"))

- a = "Hello, World!"
  print(a.split(",")) # returns ['Hello', ' World!']

# String function(4)

- a = "Hello"
  b = "World"
  c = a + " " + b
  print(c)

- quantity = 3
  itemno = 567
  price = 49.95
  myorder = "I want {} pieces of item {} for {} dollars."
  print(myorder.format(quantity, itemno, price))

# Exercise

1) Given Apple $5, Banana $14, Orange $6
Calculate the summation for 12 pc Apple, 15 pc Banana, 21 pc Orange
Print the price with 1 line

2) Apple discounted 5%, Banana discounted 15% but Orange increased 10%, please  renew the reasonable price in 1 line

3) If I have budget $400, I want to buy 14 pc Apple, 20 pc Banana, 52 pc Orange, how much is owed?

# Exercise (cont.)

4) If I have $200 and want to buy and share the fruits to 3 children *FAIRLY. How many Apple, Banana and Orange that I should buy and how much dollar remaining.

5) How about I have $333 budget? (Change the input from $200 to $300)

Given “print('Enter your budget:’)
budget = input()”

# Python Arithmetic Operator

| Operator | Name | Example |
|---|---|---|
| + | Addition | x + y |
| - | Subtraction | x - y |
| * | Multiplication | x * y |
| / | Division | x / y |
| % | Modulus | x % y |
| ** | Exponentiation | x ** y |
| // | Floor division | x // y |

# Python Assignment Operators

| Operator | Example | Same As |
|---|---|---|
| = | x = 5 | x = 5 |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| *= | x *= 3 | x = x * 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |
| //= | x //= 3 | x = x // 3 |
| **= | x **= 3 | x = x ** 3 |
| &= | x &= 3 | x = x & 3 |
| \|= | x \|= 3 | x = x \| 3 |
| ^= | x ^= 3 | x = x ^ 3 |
| >>= | x >>= 3 | x = x >> 3 |
| <<= | x <<= 3 | x = x << 3 |

# Data Type: Boolean

- Booleans represent one of two values: True or False.
- a = 200
  b = 33

  ```
  if b > a:
    print("b is greater than a")
  else:
    print("b is not greater than a")
  ```

# Data Type: List, Tuples, Set, Dictionary

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is ordered* and changeable. No duplicate members.
- As of Python version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*.

# List function

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])      #apple
```

```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])     #cherry
```

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon",
"mango"]
print(thislist[2:5])   # ['cherry', 'orange', 'kiwi']
```

# List Function (2)

- thislist = ["apple", "banana", "cherry"]
  thislist[1] = "blackcurrant"
  print(thislist)

- thislist = ["apple", "banana", "cherry"]
  thislist.append("orange")
  print(thislist)

# List function (3)

- thislist = ["apple", "banana", "cherry"]
  thislist.remove("banana")
  print(thislist)

- thislist = ["apple", "banana", "cherry"]
  for x in thislist:
    print(x)

# List function (4)

- thislist = ["apple", "banana", "cherry"]
  for i in range(len(thislist)):
    print(thislist[i])

- thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
  thislist.sort()
  print(thislist)

# Data Type: Tuple

- mytuple = ("apple", "banana", "cherry")

# Change item for Unchangeable Tuple

- x = ("apple", "banana", "cherry")
  y = list(x)
  y[1] = "kiwi"
  x = tuple(y)

  print(x)

# Data Type: Set (unordered, unchangeable)

- myset = {"apple", "banana", "cherry"}

- Once a set is created, you cannot change its items, but you can add new items.

- thisset = {"apple", "banana", "cherry"}
  thisset.add("orange")
  print(thisset)

# Data Type: Dictionaries

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered*, changeable and does not allow duplicates.

- thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
  }

# Exercise

1) Store ”Apple”, “Banana”, “Orange”, “Avocado”, “Durian”, “Kiwi” to List, Tuple and Set Type
2) Assign the price for above fruit with $5, $12, $6, $22, $85, $13 as Dictionary Type
3) List all the fruit
4) Sorting the fruit
5) Remove “Avocado” in the list
6) Add “Watermelon” to the list
7) Update a list show only the fruit price is under $15.

# Exercise / Homework

- Create Tic Tac Toe Game with 2 users input
- Input Players' name
- Print the game space
- Create the game space to store game state.
- Request x,y coordinate as input (e.g. "0,0", "2,2", "1,2" etc)
- Verify the input
- Print the updated game space
- Check the game is over and announce player name

# Next Class

- 28/4 Wed 19:30