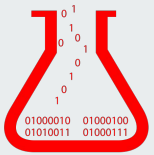
IF Statements

IF Statements

- Creates a condition, which if True, executes the block of code. If False, it skips over it.
- Format:
  - if boolean_condition:
  - 	#code to execute

- The block of code is indented with a tab (or 4 spaces)
- The IF block ends when the indentation ends

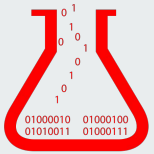


IF Statements

- Creates a condition, which if True, executes the block of code. If False, it skips over it.

```
if my_animal == “mammal”:
    print(“It’s a mammal”)
```

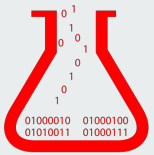
- The block of code is indented with a tab (or 4 spaces)
- The IF block ends when the indentation ends



Nested IF Statements

- IF statements can be nested

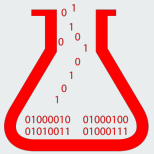
```
If my_animal == “mammal”:
    if my_animal_species == “dog:
            print(“It’s a dog!”)
print(“It’s a mammal”)
```



IF-ELSE

- ELSE blocks execute if the IF condition is False

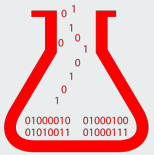
```
if my_animal == “mammal”:
    if my_animal_sepcies == “dog:
        print(“It’s a dog!”)
    else:
        print(“It’s a mammal, but not a dog”)
```



ELIF (else if)

- ELIF executes a second IF condition, if the first condition is False

```
if my_animal == “mammal”:
    if my_animal_sepcies == “dog:
        print(“It’s a dog!”)
    elif my_animal_sepcies == “cat”:
        print(“It’s a cat!”)
    else:
        print(“It’s a mammal, but not a dog or a cat”)
```



Equality Operators

equality ==

inequality !=

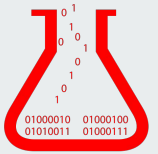
less than <

greater than >

less than or equal <=

greater than or equal >=

membership **in**

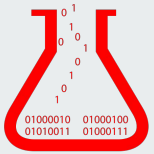


Default False vs. Default True

In python, everything evaluates as True by default unless it is empty or does not exist.

All empty sets, tuples, dictionaries, “0” evaluate to False

```
y = [] # an empty list
if y:
    print("y is full")
else:
    print("y is empty")
```



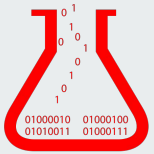
Default False vs. Default True

In python, everything evaluates as True by default unless it is empty or does not exist.

All empty sets, tuples, dictionaries, “0” evaluate to False

```
y = [] # an empty list
if y:
    print("y is full")
else:
    print("y is empty")
```

Output: y is empty

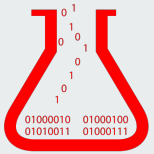


“is” vs “==”

- “==” tests for the same value, “is” tests for the same object

  Example:

```
x=5.0
y=5
print(x == y)
print(x is y)
```



“is” vs “==”

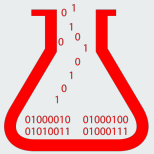
- “==” tests for the same value, “is” tests for the same object

  Example:

```
x=5.0
y=5
print(x == y)
print(x is y)
```

Output:
```
True
False
```

To check the exact ID of an object in memory, use print(id(x))



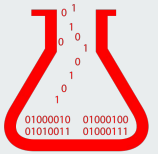
“is” and Lists

- IS is useful for checking if two lists are the same.

  Example:

```
x=[1,2,3]
y = x
z = x.copy()
print(x is y)
print(x is z)
```

Output:
```
True
False
```



Boolean Operators

In Python, boolean operators (**and, or, not**) have lower **precedence** than the code chunks that they are comparing:

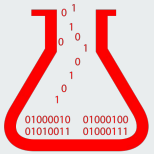
```
x = 11
y = 5
```

`x >= 11 and y < 6` True, equivalent to `(x >= 11) and (y < 6)`

`x > 12 and y < 6` False

`x > 12 or y < 6` True

`x > 10 and not y < 3` True

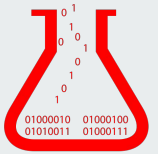


IF statement practice:

- What's the expected output?

```
x = [2,4,6,8]


if x:
    print(x)
else:
    print("No x to be found!")
```



IF statement practice:

- What's the expected output?

```
x = [2,4,6,8]

if x:
    print(x)
else:
    print("No x to be found!")
```

Output: [2,4,6,8]