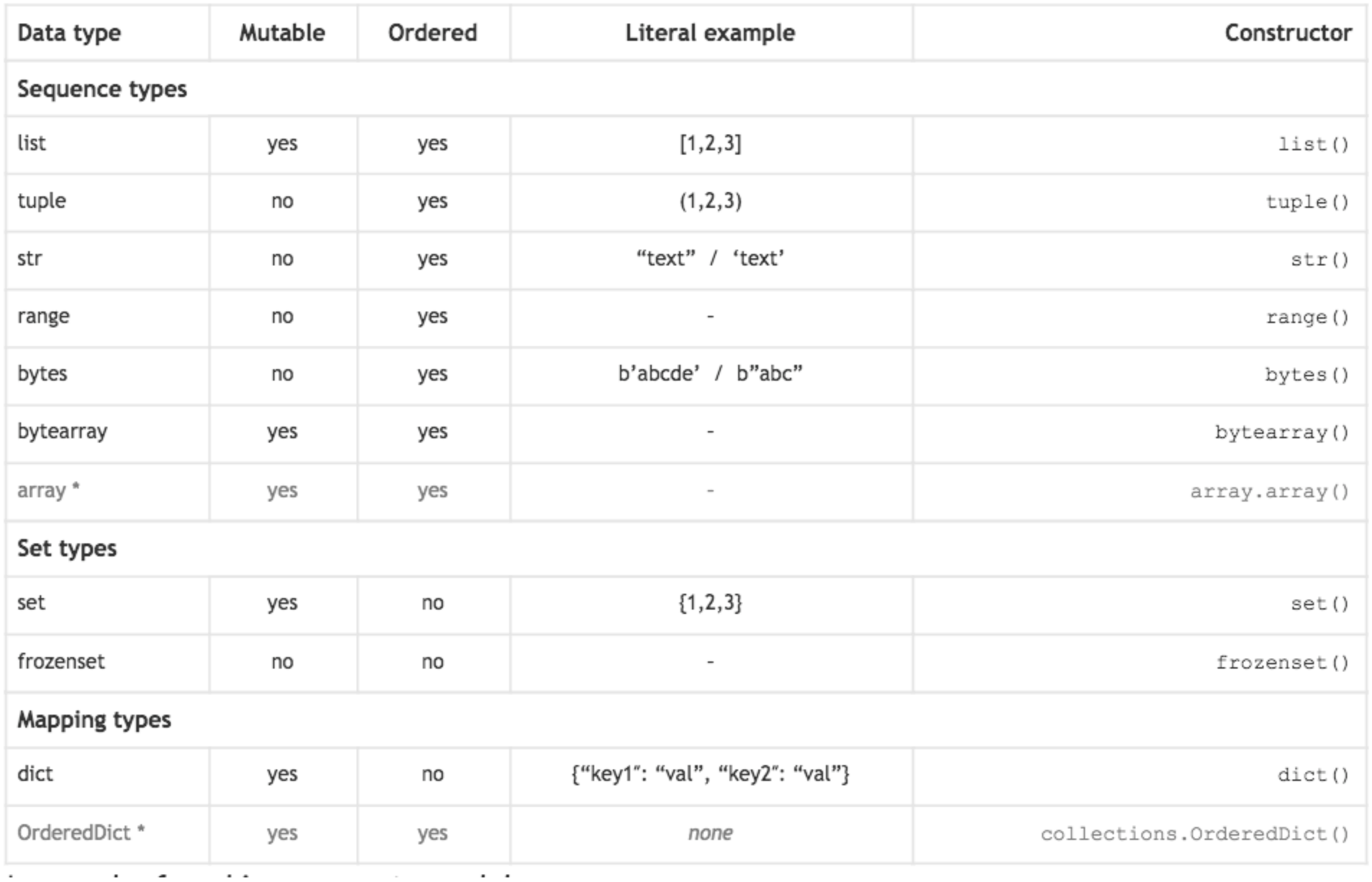
INTRO. TO DATA SCIENCE

ITERATIONS AND CONDITIONALS

TODAY'S OBJECTIVES

- RECAP : DATA TYPES AND VISUALISATIONS
- Pandas groupby() method . Update on slack
- Programming constructs
- Use for-loops and while loops, and understand when to use one vs the other.
- Iterate over data contained in lists
- Iterate over data contained in dictionaries
- CONDITIONALS - Understand how an if-else statement works
- Identify the use cases for using - if , if-else, if-elif-elif….-else.
- Bonus lab : User Input and Output basics
  https://github.com/ShakeelRaja/user-IO/blob/master/index.ipynb
- Bonus lab: Data Cleaning and visualisation with Plotly and Pandas
  https://github.com/learn-co-curriculum/ds-python-plotly

PYTHON DATA TYPES QUICK REFERENCE

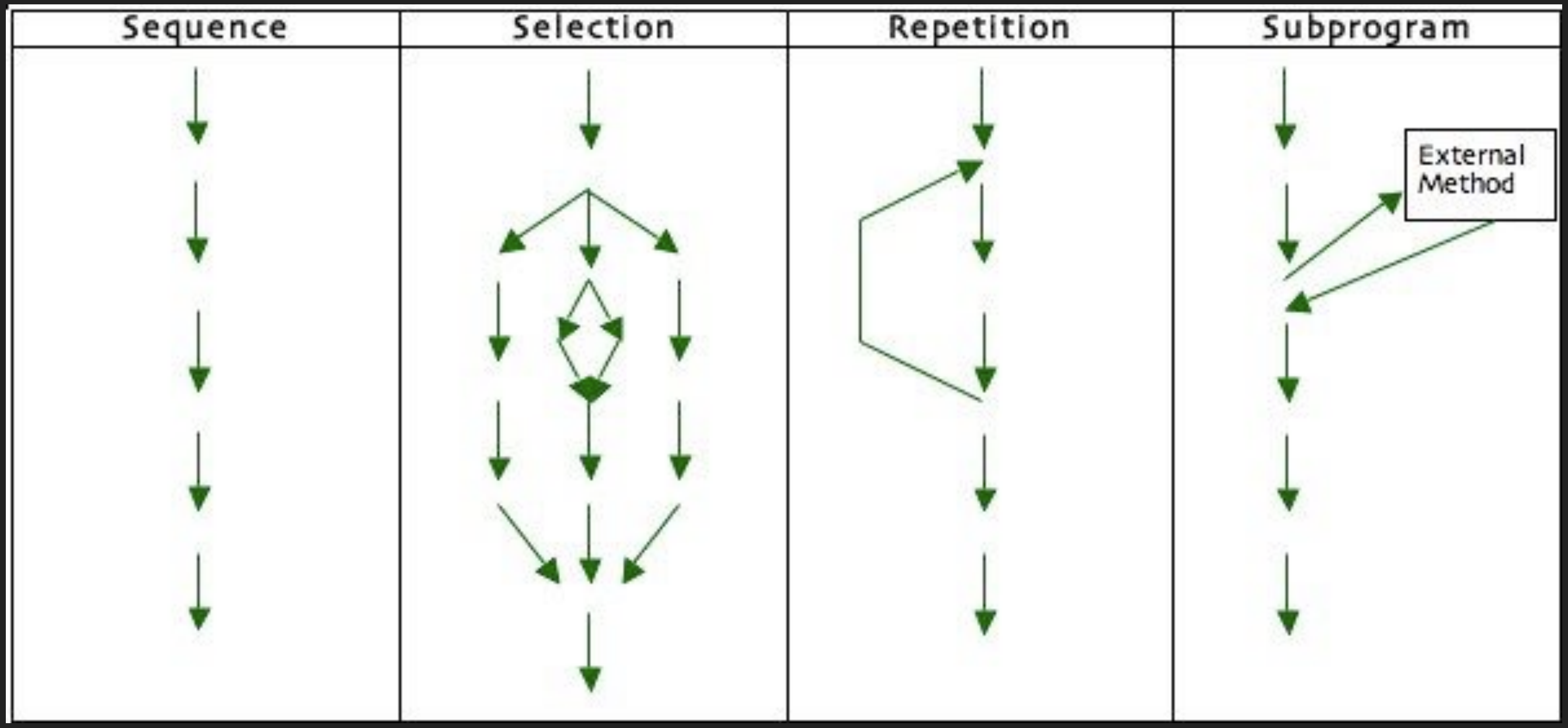


Data type	Mutable	Ordered	Literal example	Constructor
Sequence types				
list	yes	yes	[1,2,3]	`list()`
tuple	no	yes	(1,2,3)	`tuple()`
str	no	yes	“text” / ‘text’	`str()`
range	no	yes	-	`range()`
bytes	no	yes	b’abcde’ / b”abc”	`bytes()`
bytearray	yes	yes	-	`bytearray()`
array *	yes	yes	-	`array.array()`
Set types				
set	yes	no	{1,2,3}	`set()`
frozenset	no	no	-	`frozenset()`
Mapping types				
dict	yes	no	{“key1”: “val”, “key2”: “val”}	`dict()`
OrderedDict *	yes	yes	*none*	`collections.OrderedDict()`

CONTROL STRUCTURES IN PROGRAMMING

There are four different ways control can flow through a program in any programming language:

- Sequence: Execute every statement, in order.
- Selection: Execute some statements (branches of the code) based on whether or not conditions are met.
- Repetition: Execute some statement multiple times, again using some conditions to determine whether/how many times to repeat.
- Subprogram: Leave the regular flow of a program and execute code written elsewhere, i.e. a method.



LOOPS MOTIVATION

- Given the following list
  Linguists = ['Amanda', 'Claire', 'Holly', 'Luis', 'Nick', ‘Sophia']
- How do I print each name on a separate line?

TEXT

linguists = [“Amanda”, “Claire”, “Holly”, “Luis”, “Nick”, “Sophia”]

- print (linguists[0] + ‘\n’)
- print (linguists[1] + ‘\n’)
- print (linguists[2] + ‘\n’)
- print (linguists[3] + ‘\n’)
- print (linguists[4] + ‘\n’)
- print (linguists[5] + ‘\n’)

FOR LOOPS

- A for-loop steps through each of the items in a list, tuple, string, or any other type of object which the language considers an “iterable” (an **iterator** is an object that enables a programmer to traverse its contents).

-
  ```
  for <ITEM> in <COLLECTION> :
          <STAMENT(s)>
  ```
- When <COLLECTION> is a list or a tuple, then the loop steps through each element of the container.
- When <COLLECTION> is a string, then the loop steps through each character of the string.

-
  ```
  for someChar in “Hello World”:
        print someChar
  ```

FOR LOOPS

- The part of the for loop can also be more complex than a single variable name.
- When the elements of a container are also containers, then the part of the for loop can match the structure of the elements i.e. using tuples (x,y).
- This multiple assignment can make it easier to access the individual parts of each element.

-
  ```
  for (x, y) in [('a',1), ('b',2), ('c',3), ('d',4)]:
        print x
  ```

TEXT

- Linguists = ['Amanda', 'Claire', 'Holly', 'Luis', 'Nick', ‘Sophia']

-
  ```
  For linguist in Linguists:
          print (linguist)
  ```

EXERCISE

- How do we take the sentence “Python is a great text processing language” and print one word on each line?
- Hint: recall splitting sentences into words from “instant data science” demo.

FOR LOOPS AND RANGE() FUNCTION

- We often want to range a variable over some numbers, we can use the range() function which gives us a list of numbers from 0 up to but not including the number we pass to it. i.e. range(5) = [0,1,2,3,4]

-
  ```
  for x in range(5):
      print (x)
  0
  1
  2
  3
  4
  ```

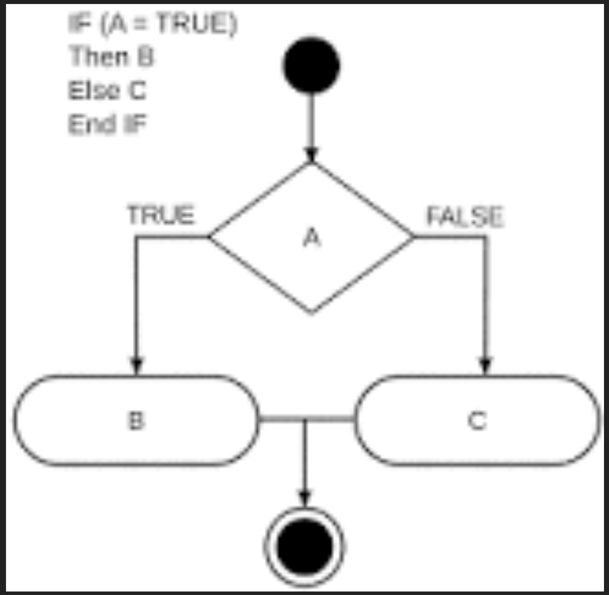
EXERCISE

- Suppose we have a list of integers:
  numbers = [1, 2, 3, 4, 5, 6]
- What do we need to do add 5 to each number?
- What if we want to add 5 to only the second to the fifth number?
- What if we want to add 5 to even numbers?

CONDITIONALS WITH BINARY OPERATORS

In computer science, conditional statements, conditional expressions and conditional constructs are features of a programming language, which perform different computations or actions depending on whether a programmer-specified boolean condition evaluates to true or false.

Operator	Meaning
<	Less than
>	Greater than
==	Equal
<=	Less than or equal
>=	Greater than or equal
!=	Not equal

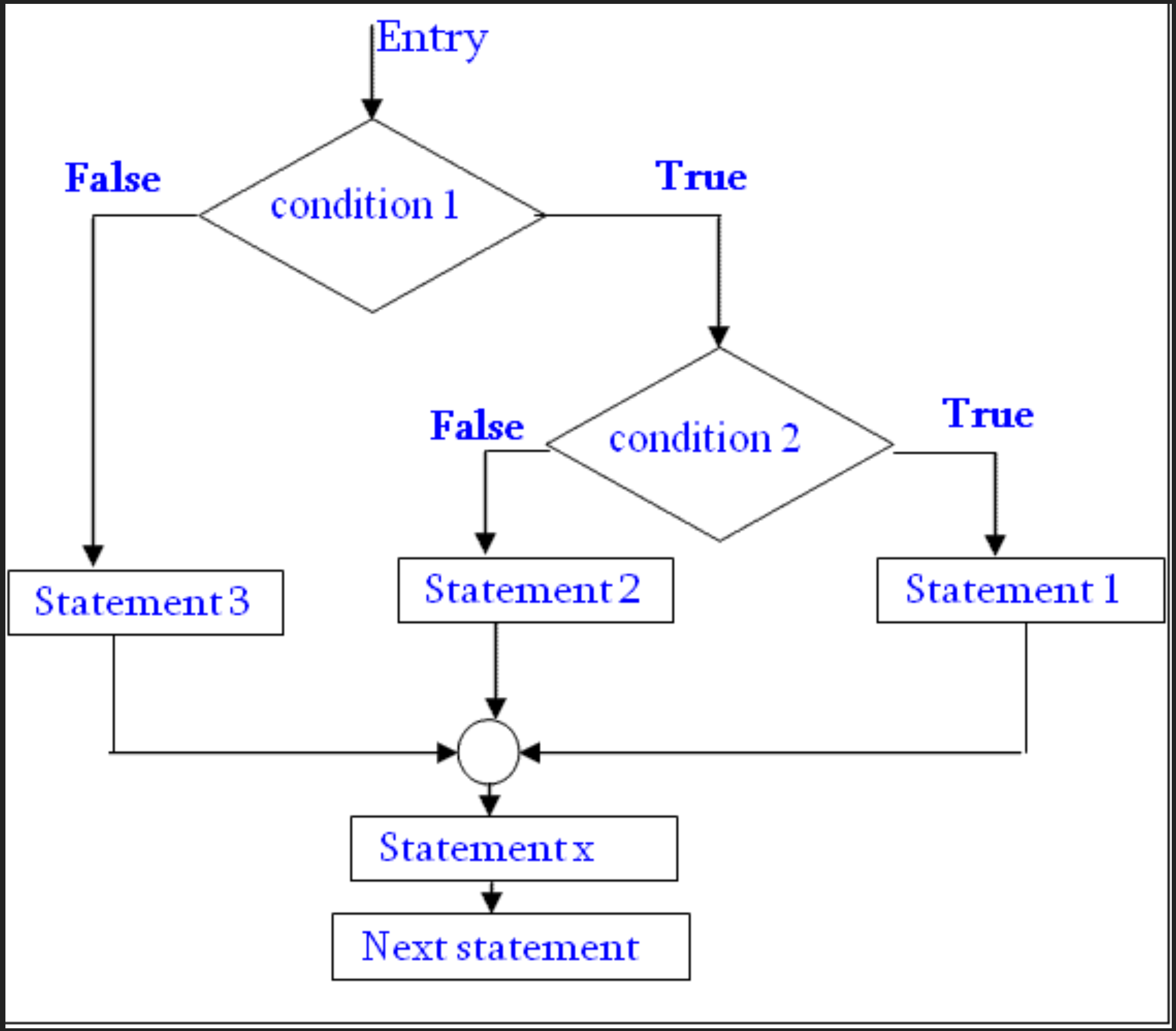


CONDITIONALS

- Conditionals are structures within the code which can execute different lines of code based on certain 'conditions' being met. In Python, the most basic type of conditional will test a Boolean to see if it is True, and then execute some code if it passes:

-
  ```
  b = True
  if b :
      print ('b is True')
  ```

NESTED LOOPS



WHILE LOOP

- The syntax for the While-Statement is

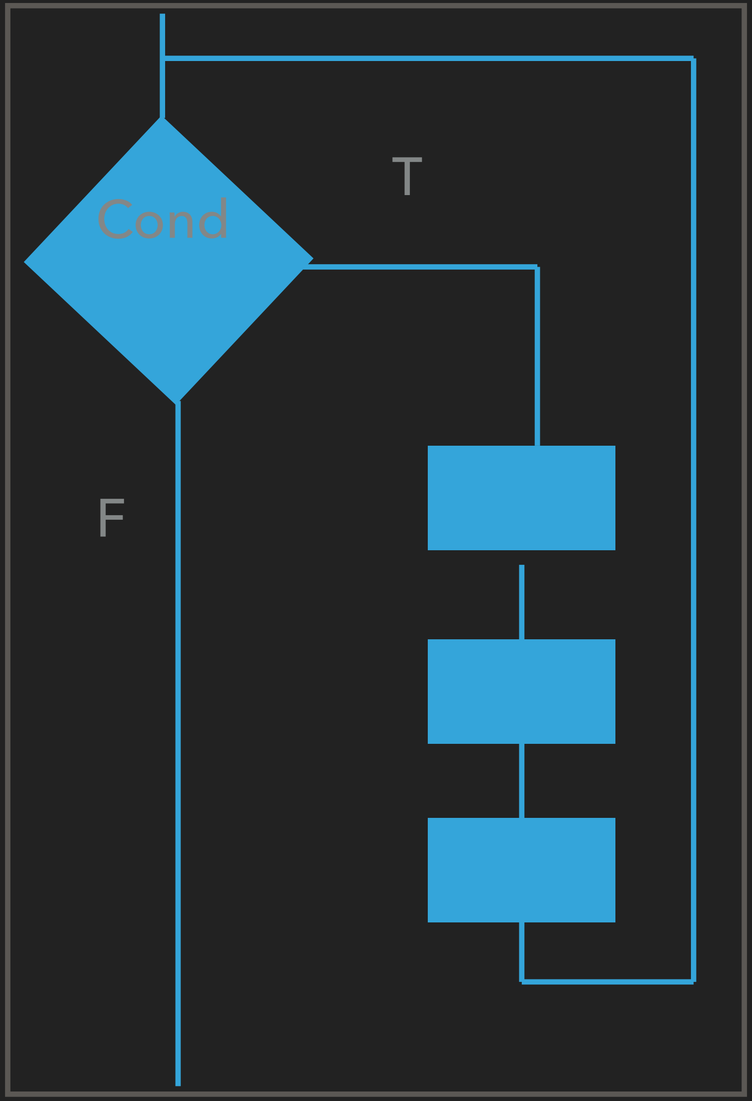
```
while <condition> :
    <list of statements>
```

- Note the colon and indentation

- x = 1

```
while x < 10:
    print x
```

- Do you see a problem here ?



WHILE LOOP

- Its very easy to get stuck in an infinite loop using “while” statement. Control condition must be set carefully