# Slides

## Slide 2

In python a block of code can be defined by an indentation (tab size) and a Colon , which means that as long as one or more code instructions are written with an indentation, they are part of the same block and context.

```
def f(n):
    if n == 1:
        return 1
    else:
        return f(n-1)

print(f(4))
```

When we discuss about variables or objects that are defined in a program. They are only relevant to the block that defined them and the blocks that are defined under them. The highest scope of a program would usually be called ‘the main scope’. Throughout this presentation we will see different python tools that are all relying on scoping and right indentations.

## Slide 4

Add reference to https://www.learnpython.org/en/Conditions. Go through the interactive tutorial, very useful and practical.

## Slide 8

Add reference to https://www.learnpython.org/en/Loops. Go through the interactive tutorial, very useful and practical.

## Slide 19

Function can be defined without any parameter, get a simple parameter or get as an input a complex variable such as dictionary or even another function, also any parameter can be defined with a default value. Functions can also not return any value or return an array of values or even return another function. When function in being interpreted, each thing that is defined within the function is only relevant to its scope and not  outside of it.

## Slide 24

A dictionary is a data type similar to arrays, but works with keys and values instead of indexes. Each value stored in a dictionary can be accessed using a key, which is any type of object (a string, a number, a list, etc.) instead of using its index to address it.
myDict = {“name”: “aviel”, age: 28, hobbies: [“Skiing”,”Cooking”], “children”: None}

## Code from the Class

```
from functools import partial
DogPrefix = "Dog bark sounds like: "


isTrue = False
a = 2
b = 2.5
strOne = "One"
strThree = "Tree"
if a < b:
  print("%d is lower than %f" % (a, b))

if strOne != strThree:
  print("Strings are different")
else:
  print("Strings are the same")

if a < b and strOne != strThree:
  print("a is lower than b and the strings are not equal")

if isTrue:
  print("Never going to happen")
elif not isTrue:
  print("Always going to happen")
```

```
def square(number):
  return number*number


def bark():
  print(DogPrefix + "Whoof! Whoof!")


def miao(catsound="Miao! Miao!"):
  return print(catsound)

def makeasound(sound):
  sound()

print(square(7))
miao("Hatoola")
makeasound(bark)
makeasound(miao)
makeasound(partial(miao, "Miaoooooo!"))

myDict = {"name": "aviel",
      "age": 28,
      "hobbies": ["Skiing", "Cooking"],
      "children": None}

print(myDict["name"])
myDict["age"] = 29
print(list(myDict.keys()))
print(list(myDict.values()))

print("Enter your name:", end =" ")
name = input()
print("Have a good day " + name)
```

Exercise:

```
# 1
def count(list_to_count):
    count = 0
    for i in list_to_count:
        count+=1
    return count

def dict_validator(dict_to_validate):
    if dict_to_validate.get('name') is None or  not type(dict_to_validate.get('name')) is str:
        print("missing or incorrect type field: name")
    if dict_to_validate.get('age') is None or not type(dict_to_validate.get('age')) is int:
        print("missing or incorrect type field: age")
    if dict_to_validate.get('hobbies') is None or not type(dict_to_validate.get('hobbies')) is list:
        print("missing or incorrect type field: hobbies")

dict_validator({"name": "aviel"})
print("--------------")
dict_validator({"name": 234})
print("--------------")
dict_validator({"name": "aviel", "age": 28})
print("--------------")
dict_validator({"name": "aviel", "age": 28, "hobbies":["teaching", "DevOps", "playing the uke"]})
```