Programming with Python 3

Functions in Python

- Harsh Nandwana

Python Functions

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.

  - **Creating a Function**
    - In Python a function is defined using the def keyword:

```
def my_function():
  print("Hello from a function")
```

  - **Calling a Function**
    - To call a function, use the function name followed by parenthesis:

```
def my_function():
  print("Hello from a function")

my_function()
```

Arguments

- Information can be passed into functions as arguments.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
- The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

Example :

```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

Arbitrary Arguments, *args

- If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition.

```
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

Keyword Arguments

- You can also send arguments with the *key = value* syntax.This way the order of the arguments does not matter.

```
def my_function(child3, child2, child1):
  print("The youngest child is " + child3)

my_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")
```

Default Parameter Value

- The following example shows how to use a default parameter value.
- If we call the function without argument, it uses the default value:

```
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

Passing a List as an Argument

- You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.
- E.g. if you send a List as an argument, it will still be a List when it reaches the function:

```
def my_function(food):
  for x in food:
    print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

Return Values

- To let a function return a value, use the `return` statement:

```
def my_function(x):
  return 5 * x


print(my_function(3))
print(my_function(5))
print(my_function(9))
```

The pass Statement

- `function` definitions cannot be empty, but if you for some reason have a `function` definition with no content, put in the `pass` statement to avoid getting an error.

```
def myfunction():
  pass
```

Recursion Function

- Python also accepts function recursion, which means a defined function can call itself.
- Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.
- example, tri_recursion() is a function that we have defined to call itself ("recurse"). We use the k variable as the data, which decrements (-1) every time we recurse. The recursion ends when the condition is not greater than 0 (i.e. when it is 0).

```
def tri_recursion(k):
  if(k > 0):
    result = k + tri_recursion(k - 1)
    print(result)
  else:
    result = 0
  return result

print("\n\nRecursion Example Results")
tri_recursion(6)
```

getsizeof() method – Python

- The **getsizeof()** is a system-specific method and hence we have to import the sys module to use it.
- A sample code is as shown below for calculating the size of a list.

```
import sys
a =[1, 2]
b =[1, 2, 3, 4]
c =[1, 2, 3, 4]
d =[2, 3, 1, 4, 66, 54, 45, 89]
print(sys.getsizeof(a))
print(sys.getsizeof(b))
print(sys.getsizeof(c))
print(sys.getsizeof(d))
```

- **Output**

```
80
96
96
128
```

Thank you