**AI Day 07 Notes**

**Syed Mansoor ul Hassan Bukhari**

**Function Arguments & Return Statement**

**Default Arguments**

Default arguments are those that take a default value if no argument value is passed during the function call. You can assign this default value while defining the function.

def name(fname, mname = "Muhammad", lname = "Rizwan"):

print("Welcome", fname, mname, lname)

name("Khan")

**Keyword Arguments**

Keyword arguments allow you to pass arguments in any order. You just have to mention the argument name with its value.

def name(fname,mname,lname):

print("Hello!", fname, mname, lname)

name(mname = "Ali", lname = "Khan", fname = "Muhammad")

**Required Arguments**

Required arguments are those that must be passed during the function call. The argument passing should follow the exact positional order as defined in the function.

def average(a,b):

print("Average of two numbers: ", (a+b)/2)

average(a = int(input("Enter 1st value: ")), b = int(input("Enter 2nd value: ")))

**Arbitrary Arguments**

If you do not know how many arguments will be passed into your function, add a \* before the parameter name in the function definition. This way the function will receive a tuple of arguments.

def avg(\*num):

sum = 0

for i in num:

sum = sum+i

print("Average is: ", sum/len(num))

avg(10,10,10)

**Lists**

Lists are ordered collections of data items. They can store multiple items in a single variable. List items are separated by commas and enclosed within square brackets. Lists are mutable, meaning we can alter them after creation.

l = [3, 5, 6, "MRK", True, 9.8]

print(l)

print(type(l))

print(l[0]) #Accessing the index

**List Index**

Each item in a list has its own unique index. This index can be used to access any particular item from the list.

l = [3, 5, 6]

print(l[0]) #Accessing the index

print(l[1])

print(l[2])

**Accessing List Items**

You can access list items by using the index with the square bracket syntax. You can use both positive and negative indexing.

colors = ["Red", "blue", "Green"]

print(colors[-1])

print(colors[-2])

print(colors[-3])

**Checking Item in the List**

You can check if a given item is present in the list using the “in” keyword.

colors = ["Red", "G", "B","y"]

if "y" in colors:

print("Y is present")

else:

print("Y is not present in the list")

**Range of Index**

You can print a range of list items by specifying where you want to start, where do you want to end and if you want to skips elements in between the range.

animals = ["cat", "dog", "bat", "mouse", "horse", "elephant"]

print(animals[1:6:3])

**List Comprehension**

List comprehensions are used for creating new lists from other iterables like lists, tuples, dictionaries, sets, and even in arrays and strings.

lst = [i\*i for i in range(20) if i%2==0]

print(lst)

Sure, here are some additional notes:

**Updating Lists**

[There are several ways to update a list in Python](https://www.onlinetutorialspoint.com/python/different-ways-to-update-python-list.html)[1](https://www.onlinetutorialspoint.com/python/different-ways-to-update-python-list.html):

* list.append(value): Appends a value to the end of the list.
* list.extend(iterable): Appends a series of values to the list.
* list.insert(index, value): Inserts a value at a specific index.

lst = [1, 2, 3]

lst.append(4) # lst is now [1, 2, 3, 4]

lst.extend([5, 6]) # lst is now [1, 2, 3, 4, 5, 6]

lst.insert(0, 0) # lst is now [0, 1, 2, 3, 4, 5, 6]

**Updating Function Parameters Based on Input**

You can update function parameters based on input. [This can be done by creating a global variable and updating it inside the function](https://www.onlinetutorialspoint.com/python/different-ways-to-update-python-list.html)[2](https://stackoverflow.com/questions/48940859/how-do-i-update-function-parameters-based-on-input-in-python).

def test(b):

global a

a = a + b

a = 0

test(1) # Now, a is 1

Remember, Python is a versatile language with many features. [Understanding how to properly define function parameters, pass arguments, and handle return values is crucial for building reusable and maintainable Python programs5](https://llego.dev/posts/comprehensive-guide-passing-arguments-python-functions-handling-return-values/).