Python Programming Workshop Session 2

Avnith Vijayram
https://avnithv.github.io/

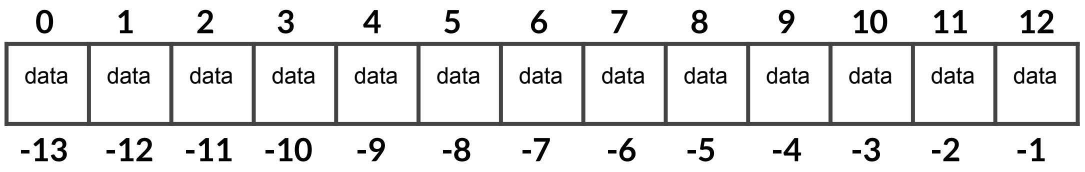
McNair Elementary School

PTA

everychild. onevoice.

Table of Contents

What are lists?



Sequences of values

Defined with square brackets and commas - [1, 2, 3]

Similar to strings with indices

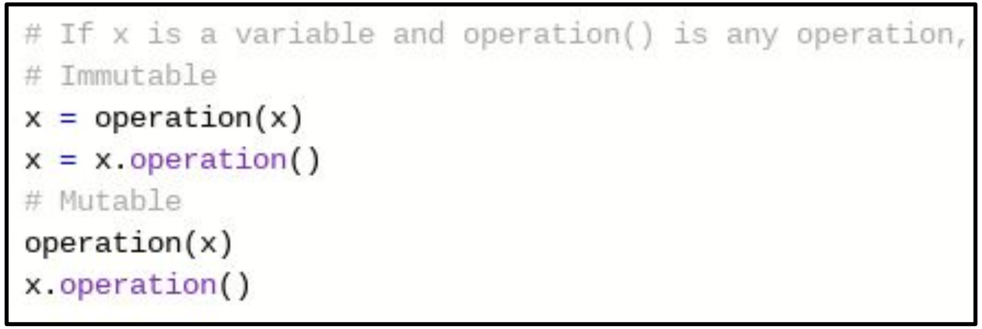
Are ordered and mutable

Mutable vs. Immutable

Changeable vs Unchangeable

All previously discussed data types are immutable except lists

Variable changes after operation vs. Assigned to variable after operation



```
# If x is a variable and operation() is any operation,
# Immutable
x = operation(x)
x = x.operation()
# Mutable
operation(x)
x.operation()
```

How to create lists

Square brackets - [ ]

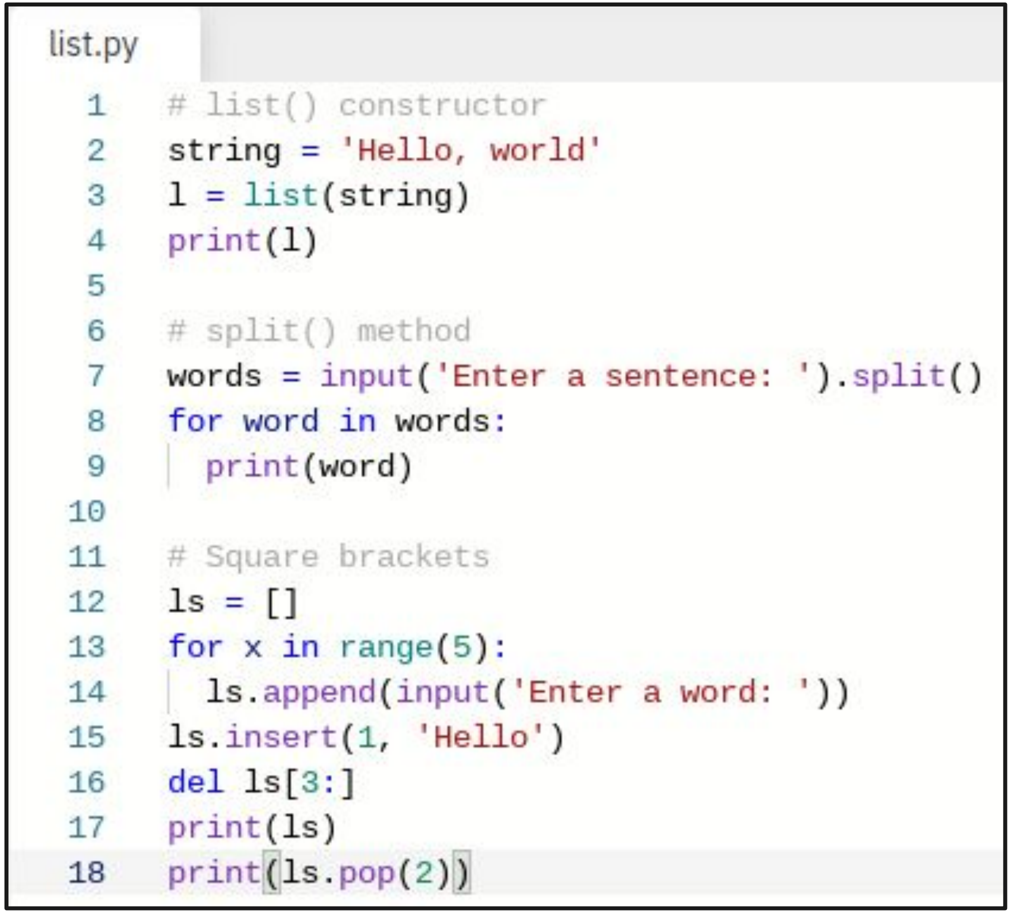
list() function

.split() method

Add item - .append(), .insert()

Change item - list[index]

Delete item - del, .pop()


list.py

```python
# list() constructor
string = 'Hello, world'
l = list(string)
print(l)

# split() method
words = input('Enter a sentence: ').split()
for word in words:
    print(word)

# Square brackets
ls = []
for x in range(5):
    ls.append(input('Enter a word: '))
ls.insert(1, 'Hello')
del ls[3:]
print(ls)
print(ls.pop(2))
```

List Functions

Slice or take a subsequence - **list[start:stop:step]**

Add items - **list.append(item)**, **list.insert(index, item)**; Delete items - **list.pop(index)**, **del list[start:stop:step]**

Change item - **list[index] = value**

Reverse order - **list.reverse()**; Sort the list -**list.sort()**

Find length - **len(list)**; Find if item is in list - **item in list;** How many times an item appears- **list.count(item)**

Sum of items in list - **sum(list)**; Maximum and minimum - **max(list)**, **min(list)**

What are dictionaries?

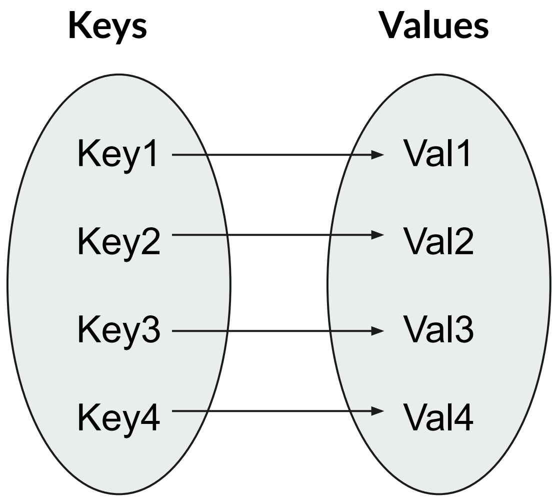
A set of ‘pairs’ of data, where one piece of data is accessed through its pair

Each pair is called a key-value pair - each key corresponds to a value

Keys and values can be any data

Similar to lists as it is mutable but there is no order and indexes

Defined with curly brackets - {}



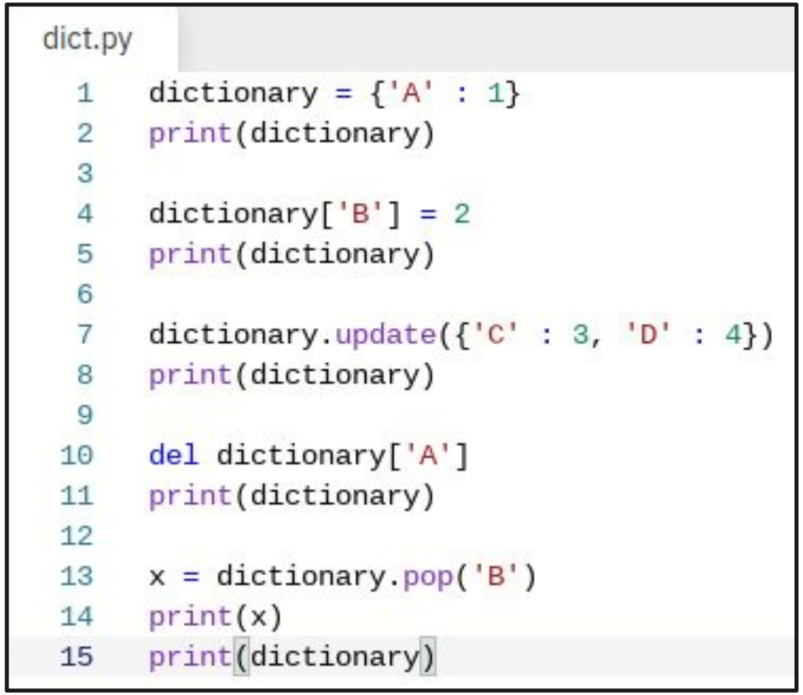
How to create dictionaries

Assign a variable curly brackets {}

Key value pairs are represented **key : value**

To add a key-value pair use dict[key] = value or dict.update({key : value)

To remove pairs use del dict[key] or dict.pop(key)


dict.py

```python
dictionary = {'A' : 1}
print(dictionary)

dictionary['B'] = 2
print(dictionary)

dictionary.update({'C' : 3, 'D' : 4})
print(dictionary)

del dictionary['A']
print(dictionary)

x = dictionary.pop('B')
print(x)
print(dictionary)
```

Dictionary functions

Find value of key - **dict[key]**

Add and remove items - **dict[key] = value, dict.update({key : value}), del dict[key], dict.pop(key)**

Find all keys, values, and key-value pairs - **dict.keys(), dict.values(), dict.items()**

Pop last inserted pair - **dict.popitem()**

Return value if key exists otherwise insert key-value pair - **dict.setdefault(key, value)**

What are functions

A function is a block of code that is executed each time it is ‘called’

Functions can have parameters and return data

When a function is called, arguments can be passed into the function.

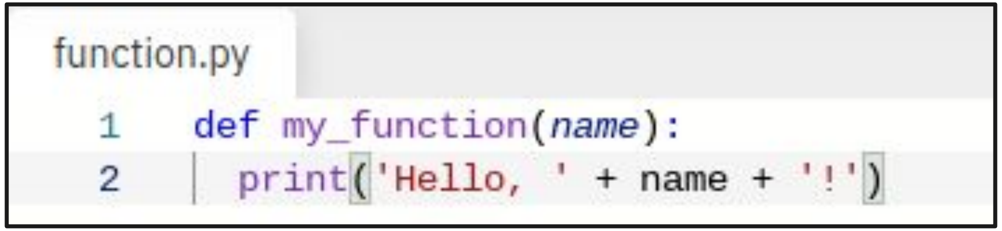
The main purpose of a function is to make the code easier to read.

As well as making it easier to reuse code, functions are used to divide code into sections that have different purposes.

Different than methods - i.e. functions with a period in front

Functions we have learned:
print()
input()
range()
len()
str()
int()
float()
ord()
chr()
range()
sleep() *from time module*
list()
sum()
max()
min()

Creating functions


function.py

```python
def my_function(name):
    print('Hello, ' + name + '!')
```

Use **def** keyword

Write the function name

Add parentheses and specify any parameters in between

End with a colon

All lines inside are indented

Parameters and Arguments

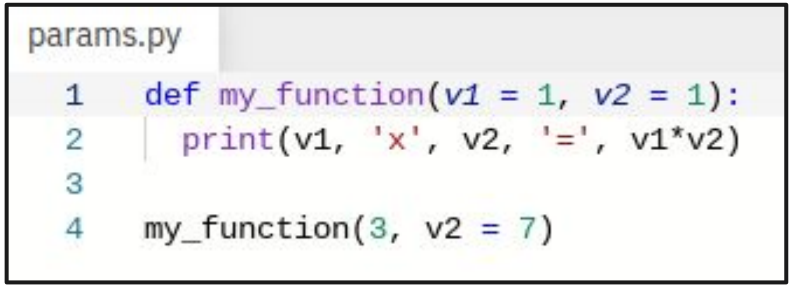
Parameters and arguments are data passed to a function.

A parameter is inside the function definition.

An argument is inside a function when it is called.

May be multiple parameters and default values

Arguments can be passed positionally or through keywords



params.py

```python
def my_function(v1 = 1, v2 = 1):
    print(v1, 'x', v2, '=', v1*v2)

my_function(3, v2 = 7)
```

Returning Data

Data is returned through the **return** keyword

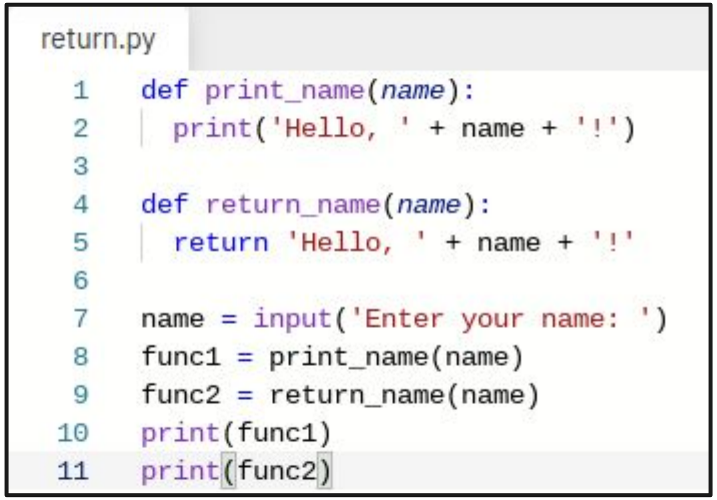
The data returned can be any datatype

None is a Nonetype, another data type representing something empty or null.

If only the return keyword is there, the function returns None.

If the function reaches the end without returning, it automatically returns None.

If the value returned is not stored in a variable, the value is lost.


return.py

```python
def print_name(name):
    print('Hello, ' + name + '!')

def return_name(name):
    return 'Hello, ' + name + '!'

name = input('Enter your name: ')
func1 = print_name(name)
func2 = return_name(name)
print(func1)
print(func2)
```

Function vs. No Function

The benefits of functions:

Easier to read

Easier to understand at first glance

Easier to change

Shorter code

Easier to locate errors

Resources

- Workshop materials and code available at https://avnithv.github.io/
- Sign up on Repl to code using online tools (no download/installation needed)
- Sign up on Hackerrank to practice Python coding using online tools (no download/installation needed)