Data and Control Structures in Python

Priscilla Ajayi (Lead Software Engineer CAPiC-ACE)

@

The 3rd Google TensorFlow College Outreach Bootcamp and FEDGEN Mini-Workshop

Date: 11th to 13th December, 2023

Sponsored By Google

CONTENTS

1.1 INTRODUCTION

- Python is a server-side, interpreted, open-source, non-compiled, scripting language that can be used on its own or as part of another framework (such as django).
- The focus of the presentation is on two fundamental pillars of Python programming which are data structures and control structures.
- Python, known for its simplicity and versatility, provides an array of tools to organize and manipulate data effectively.

- A good knowledge of data and control structures is very important for an efficient code irrespective of your level of programming.
- The next slide would help us to explore data structures and control structures in Python.

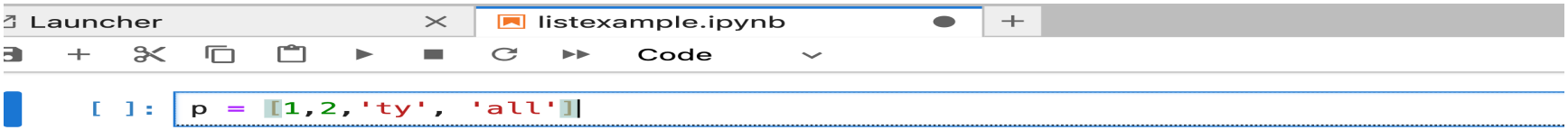
1.2 DATA STRUCTURE

- The basic Python data structures are:
  - list
  - Set
  - Tuples
  - dictionary.
- They all have their unique characteristics which enables us to know when to use them.

1.2.1 LIST

It is an ordered sequence of items, which is one of the most flexible and frequently used data type in Python.

- The elements of a can be of different data type and the index starts from 0.
- To create a list, we are to use square brackets with elements separated by commas.
- Example is below



LIST ATTRIBUTES

- Indexed: List are indexed, and it starts with 0

  Example: p = [1, 2,'ty', 'all’]

  if we are to print(p[2])

  It would give an output of ty

- Ordered: The list are ordered as the items are added, every output of the list remain as stored expect the index value is changed.

  Example: grades = [‘A’, ‘B’, ’C’]

  grades.append = (’D’)

  It add D to the next index which is 4 without altering the current indexing of variable grades.

LIST ATTRIBUTES CONTD’

- Mutable: the values of a particular index of a list can be changed

  Example: colors = [‘red’, ‘yellow’, ‘green’, ’blue’]

  print(colors[2]) will give output ‘green’

  colors[2] = ‘black’

  print(colors) will give output [‘red’, ‘black’, ‘green’, ’blue’]

  changing the value of green to black

- Allows Duplicate: it does not check for unique values on it own thereby allows duplicate
  - Example: colors = [‘red’, ‘yellow’, ‘red’, ‘green’, ’blue’]

LIST METHODS

- Some of the functions that can be carried out in a list includes:
  - List Length: returns length of a list. E.g.: print(len(colors))
  - List index: returns index of a value in a list. E.g.: colors.index('yellow’)
  - Append to List: allows to append to the next index of a list list.append()
  - List Extend: processing of merging two list together: list.extend():
  - list.insert()
  - list.remove()
  - list.count(x)
  - list.pop()
  - list.reverse() etc

1.2.2 SET

- A set is an unordered list of different data types stored in a variable.
  - To create a set, we are to use curly brackets with elements separated by commas.
    - Example: scores = {40, 50, 45, 70}

    print(scores) would give output of the defined elements above in no particular order
- A set is not indexed there by does not have order of output

SET ATTRIBUTES

- Not Ordered: set data structure does not have any particular order. When data has been stored in a set everytime it is printed, it comes out in different order
  - Example: scores = {40, 50, 45, 70}, print more than once and check output ordering.
- Cannnot be changed: since set values does not have order and it is not indexed after the set data has been created, it can not be changed.
- No Duplicates: at the point of defining a set, if it has duplcates, the set would not give an output with duplicate, it eliminates all duplicates.

SET METHODS

- add()
- clear()
- copy()
- difference()
- difference_update()
- discard()
- intersection()
- intersection_update()
- isdisjoint()
- issubset()
- issuperset()
- pop()
- remove()
- symmetric_difference()
- symmetric_difference_update()
- union()
- update()

1.2.3 TUPLES

- Tuples are data structure that can store different types of varibles in an ordered form and can be index.
- Tuples are written within brackets
  - Examples: grade = (‘A’, ‘B’, ‘C’)

- Tuples work with indexing system and starts from 0 just like the list.

TUPLES ATTRIBUTES

- Indexed: List are indexed, and it starts with 0

  Example: p = (1, 2,'ty', 'all’)

  if we are to print(p[2])

  It would give an output of ty

- Ordered: The list are ordered as the items are added, every output of the list remain.

  Example: grades = (‘A’, ‘B’, ’C’)

TUPLES ATTRIBUTES CONTD’

- Immutable : the values of a particular index of a list cannot be changed
- Allows Duplicate: it does not check for unique values on it own thereby allows duplicate
  - Example: colors = (‘red’, ‘yellow’, ‘red’, ‘green’, ’blue’)

1.2.4 DICTIONARY

- Dictionaries are used to store data values of different data or same data type in key: value pairs.
- When createing a dictionary the items of a dictionary is stored in curly brackets.

  Example: phisdata ={topic: ‘Malaria Infection’,
  catrogory: ‘article’
  yearofrealse: ’2000’}

- The key used to store to variable can be used in retrival of the variable.

  Example: print(phisdata[‘catgegory’]))

DICTIONARY ATTRIBUTES

- It is ordred: due to the key pair method of sorting data, it allows the data to always come out in an ordered list
- It is can be changed: since the data has key that can be used to refrence it, it allows for easy change of data
- It does not allow duplicates

DICTIONARY METHODS

- clear()
- copy()
- fromkeys()
- get()
- items()
- keys()
- pop()
- popitem()
- setdefault()
- update()
- values()

CONTROL STRUCTURES

- Control structures can be referred to the process of ordering the system to carrying out set of instructions in a systematic way based on certain conditions.
- Control system can also be explained as the system carrying out set of repititive instructions after which given conditions have been met.
- There two major types of control system:
  - **Selection**
  - **Repetition**

SELECTION CONTROL

- Selection control system which can also be called decision control system can be used to execute code only of defined conditions are met.
- Types of selection control system incudes:
  - If
  - If…..else
  - If……elif……else
  - Nested If

a) If statement: The syntax of the ***if statement*** is

if *test expression*:
body of if statement(s)

The body of the ***if statement*** starts with an indentation and the first unintended line marks the end of the statement.

```
score = 70
grade =‘F’
if score > 70:
        grade = "A"
else:
        grade = 'B’
print(grade)
```

b) If……else: The syntax of if…..else is below

```
If test conditions:
        body if condition met
else:
    Output if the if is not met
```

- Example:

```
score = 70
grade =‘F’
if score > 70:
        grade = "A"
else:
        grade = 'B’
print(grade)
```

c) If……elif……else: These statement is used after person there are loads of multiple conditions

```
If test conditions:
                body if condition met
else if another condition:
        Output if the if is not met
else:
      Output if condition not met
```

```
score = 39
if score >= 70 and score <= 100:
   grade = "A"
elif score > 59 and score <= 69:
   grade = 'B'
elif score > 49 and score <= 59:
   grade = 'C'
elif score > 44 and score <= 49:
   grade = 'D'
else:
   score = 'F'
print(grade)
```

d) Nested If: this is the control statements that checks for an if inside another if

```
score = 39
grade = ""
studentstatus ="n current"
if studentstatus == "current":
   if score >= 70 and score <= 100:
      grade = "A"
   elif score > 59 and score <= 69:
      grade = 'B'
   elif score > 49 and score <= 59:
      grade = 'C'
   elif score > 44 and score <= 49:
      grade = 'D'
   else:
      score = 'F'
else:
   print("not current student")
print(grade)
```

Repetitive Control Structure

- It is a structure that allows instructions to be passed/executed several time until certain conditions are met.

- There are two types of loop
  - Whle loop
  - For loop

a) While loop: it means continue to repetitively carry out instructions until a condition is met.

The condition might be just one or combination of dirrferent conditions.

Code Sample

while *condition:*

code statement(s)

- It is very much application to different senerios for example if you want to continue to generate a set of random numbers until you get a number that does not exists in a set of data

Example: print numbers from 1 until modulus of 3 is 2

```
num = 0;
while num%3 != 2:
   print(num)
   num += 1
```

b) While loop with else: this is a while loop that is halted after a number of repetitive instructions

Code Sample

```
while condition:
    code statement(s)
else:
    code statement(s)
```

Example: print module of 3 from 1 to 50

```
num = 0;
while num < 50 :
   print(num%3)
   num += 1
else:
      print(“maximium Number is “, num)
```

c) For loop: for loops are usually used to loop through a set of variables in form of list or otherwise application to carry out instructions.

Sample of code is

for *var* in sequence:

code statements(s)

Example

```
#Compute the sum of numbers in a list
#Using for loop
lstNum = [1,2,3,4,5]
sum = 0
for var in lstNum
        sum = sum+var
#print the computed sum
print("the sum of the list is ",sum)
```

THANK YOU FOR LISTENING

QUESTIONS