Lets learn Python

ekakshya.com
13 December 2021

Lesson 3

Dictionaries
Sets
Exceptions
Functions
Scope

Dictionary

- Dictionary is an unordered key-value storage structure. Also, a member of data type collection.
- Values can be retrieved through keys only
- Generally, used for calculating frequency of some characters in a given term. For example: Apple { “A”: 1, “p”: 2, “l”: 1, “e”: 1}
- Faster key search and retrieval times when compared to list
- Dictionaries can be used to keep track of how many times a key occurs in the given string, list or tuple

Dictionary

```
a_dict = { “A”: 1, “p”: 2, “l”: 1, “e”: 1} # cannot have dictionary for a key
print(len(a_dict))
a_dict[‘e’] += 1
a_dict[‘g’] = a_dict.get(“g”, 0) + 1
print(a_dict[“A”])
del a_dict[‘e’]
print(a_dict)
print(‘e’ in a_dict) #false

another_dict = dict() #another_dict = {}


for key in a_dict.keys():
        print(key)
for value in a_dict.values():
        print(value)
for key,value in a_dict.items():
        print(key,value)
```

Sets

- An unordered unique collection of elements
- It can contain only immutable data types like tuples. As such lists, set and dictionary cannot be added to a set
- Sets can be used to check if an element is present or not inside it

Sets

```
a_set = {3,4,5,5,4,3,”Hi”, False,(3,8)}
another_set = set() #initializing another_set
print(a_set)

a_set.add(7)
a_set.add(7)

print(a_set)

a_set.remove(7)
print(a_set)
print(len(a_set))
print(4 in a_set)
```

Sets

```
a_set = {3,4,5,5,4,3,”Hi”, False,(3,8)}
another_set = {3,4,5,7,8,9}

#does not modify original set
union_set = a_set.union(another_set)  #a_set | another_set
intersection_set = a_set.intersection(another_set) #a_set & another_set
difference_a_set = a_set.difference(another_set) #a_set - another_set
difference_another_set = another_set.difference(a_set) #another_set – a_set
symmetric_difference_a_set = a_set.symmetric_difference(another_set) #a_set ^ another_set


print(union_set)
print(intersection_set)
print(difference_a_set)
print(difference_another_set)
print(symmetric_difference_a_set)
print(a_set)


#modifies original set
a_set.update(another_set)
a_set.difference_update(another_set)
another_set.symmetric_difference_update(a_set)
print(a_set)
print(another_set)
```

Sets

```
a_set = {3,4,5,5,4,3,”Hi”, False,(3,8)}
another_set = {3,4,5}

print(another_set.issubset(a_set)) # another_set <= a_set
print(a_set.issuperset(another_set)) # a_set >= another_set

#proper superset and subset
print(another_set < a_set) #proper subset
print(a_set > another_set) #proper superset
```

Exceptions

- Exceptions are used to handle errors
- It is a form of defensive programming so that the errors doesn’t crash our program
- We can handle all exceptions through generic exception or specific exceptions depending on our use case
- Errors occur sequentially so their specific exception handling block is called when we have multiple exceptions in a program
- Not a good practice to have generic exception handler in python
- Use exception handler only when there is a possibility of error
- Runtime exception occurs when program is being executed
- Compile time exception occurs when program is converted to low level code and before the program gets executed

Exceptions

```
try:
        2/0
        a_list[1] += 1
except Exception as e:
        print(“Generic exception handler called”)
except ZeroDivisionError as e:
        print(“Exception occured”, e)
finally:
        print(“Done”)

#prevent program from running
raise Exception(“An error occured”)
```

Functions

- Function is a reusable piece or block of code
- Some functions we have used so far: input(), print(), len()

```
def another_function():
        print(“Hello World”)


def a_function(a_string=”Hello”):
        print(a_string)


def add_10(num=7):
        return num + 10


def is_number_value(value= 0, num=5):  #(value=0 , num) wont work | (value, num=5) will work
        if num == value:
                return 1
        return -1

another_function()
a_function()
add_10(15)
is_number_value(num=-7)
is_number_value(num=-7, value=-7)
```

Functions

```
def a_function(a_num,another_num):
        return a_num + 1, another_num +1

first_num, second_num = a_function(3,1)
result = a_function(7,2)
print(first_num, second_num)
print(result)
```

PS: Functions can be defined within a function as such they are not available outside the function

Scope

```
num = 0 #global scope
def a_function():
        num = 10 #function scope
print(num)
```

PS: Variable defined inside a functions are limited to scope of function it is defined. Hence, they are not accessible outside the function

The End

Thank You