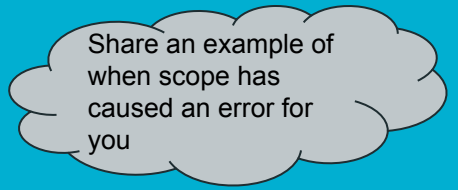
Python Interview Questions

—

July 12, 2023

Clara Oromendia

1. What is Scope in Python?



Every object in Python functions within a scope. A scope is a block of code where an object in Python remains relevant. Namespaces uniquely identify all the objects inside a program. However, these namespaces also have a scope defined for them where you could use their objects without any prefix. A few examples of scope created during code execution in Python are as follows:

A local scope refers to the local objects available in the current function.

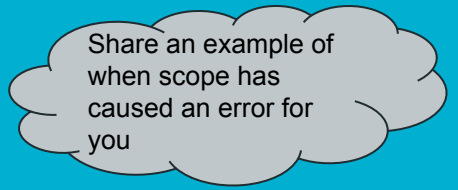
A global scope refers to the objects available throughout the code execution since their inception.

A module-level scope refers to the global objects of the current module accessible in the program.

An outermost scope refers to all the built-in names callable in the program. The objects in this scope are searched last to find the name referenced.

Note: Local scope objects can be synced with global scope objects using keywords such as global.

1. What is Scope in Python?

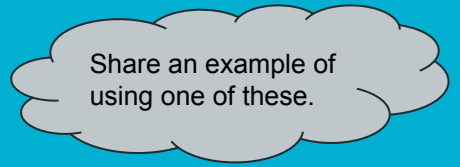



```
temp = 10   # global-scope variable
def func():
     temp = 20   # local-scope variable
     print(temp)
print(temp)   # output => ?
func()    # output => ?
print(temp)   # output => ?
```

What would be the output in each case?

How would you change so the last output was 20?

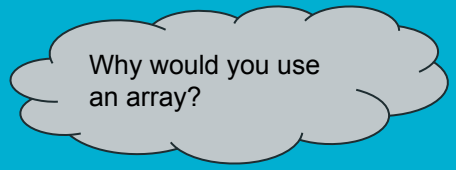
2. What are break, continue & pass?



- The break statement terminates the loop immediately and the control flows to the statement after the body of the loop.
- The continue statement terminates the current iteration of the statement, skips the rest of the code in the current iteration and the control flows to the next iteration of the loop.
- The pass keyword in Python is generally used to fill up empty blocks and is similar to an empty statement represented by a semi-colon in languages such as Java, C++, Javascript, etc.

```
pat = [1, 3, 2, 1, 2, 3, 1, 0, 1, 3]
for p in pat:
    pass
    if (p == 0):
        current = p
        break
    elif (p % 2 == 0):
        continue
    print(p)    # output => 1 3 1 3 1
print(current)    # output => 0
```

3. What is the difference between arrays & lists?



- Arrays in python can only contain elements of same data types i.e., data type of array should be homogeneous. It is a thin wrapper around C language arrays.
- Lists in python can contain elements of different data types i.e., data type of lists can be heterogeneous.

```
a = array.array('i', [1, 2, 3])
for i in a:
    print(i, end=' ')    #OUTPUT: 1 2 3
a = array.array('i', [1, 2, 'string'])    #OUTPUT: TypeError: an integer is required
(got type str)
a = [1, 2, 'string']
for i in a:
   print(i, end=' ')    #OUTPUT: 1 2 string
```

4.Give an example of a Dict/List comprehension

When would a list comprehension make code too compact?

- Comprehensions are syntactic sugar constructs that help build altered and filtered lists, dictionaries, or sets from a given list, dictionary, or set.

```
my_list = [2, 3, 5, 7, 11]
squared_list = [x**2 for x in my_list]    # math operation on list

my_list = [2, 3, 5, 7, 11]
squared_list = [x**2 for x in my_list if x%2 != 0]    # conditional filtering

a = [1, 2, 3]
b = [7, 8, 9]
[(x + y) for (x,y) in zip(a,b)]  # combining lists, parallel iterators
```

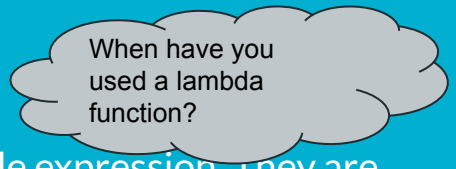
5. What are iterators?

Solve using
online-python.com

- Iterators are objects with which we can iterate over iterable objects like lists, strings, etc. Fast, memory efficient tools that are useful by themselves or in combination. Full list

```
import itertools
import operator
def main():
    arr1 = [2, 1, 3, 4, 1]
    arr2 = [1, 2, 4]
    arr3 = [10, 3, 4, 3, 5, 6, 32, 11]

    # make a new arr4 which include all the elements in order first of arr1 then arr2 and then arr3
    # Write your code here

    print(arr4)

    # using accumulate(), store the successive muliplication of elements of arr4 in a new list arr5

    print(arr5)

    return 0

if __name__ == '__main__':
    main()
```

6. What are lambda functions?



Lambda functions are generally inline, anonymous functions represented by a single expression. They are used for creating function objects during runtime. They can accept any number of parameters. They are usually used where functions are required only for a short period. They can be used as:

7. Write a program to match a string that has the ~~letter~~ ‘a’ followed by 4 to 8 'b’s.

We can use the re module of python to perform regex pattern comparison here.

```
import re
def match_text(txt_data):
        pattern = 'ab{4,8}'
        if re.search(pattern,  txt_data):    #search for pattern in txt_data
            return 'Match found'
        else:
            return('Match not found')
print(match_text("abc"))          #prints Match not found
print(match_text("aabbbbbc"))     #prints Match found
```

8. Write a Program to convert date from yyyy-mm-dd format to dd-mm-yyyy.

Solve using online-python.com

```
import re
def transform_date_format(date):
   return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', date)
date_input = "2021-08-01"
print(transform_date_format(date_input))
```

Or

```
from datetime import datetime
new_date = datetime.strptime("2021-08-01", "%Y-%m-%d").strftime("%d:%m:%Y")
print(new_data)
```

9. What would the output be? Why?


```
class Person:
    def __init__(self, first_name, last_name):
        self.first_name = first_name
        self.last_name = last_name

first_name = "XYZ"
person = Person(first_name, "ABC")
first_name = "LMN"
person.last_name = "PQR"
print(person.first_name, person.last_name)
```

10. What would the output be? Why?


```
class X:
    def __init__(self):
        self.__num1 = 5
        self.num2 = 2

    def display_values(self):
        print(self.__num1, self.num2)
class Y(X):
    def __init__(self):
        super().__init__()
        self.__num1 = 1
        self.num2 = 6
obj = Y()
obj.display_values()
```

11. Given an integer n, return the number of ~~tr~~ailing zeroes in n factorial (n!)


```
def factorial_trailing_zeros(n):

  fact = n
  while n > 1:
    fact *= n - 1
    n -= 1

  result = 0

  for i in str(fact)[::-1]:
    if i == "0":
      result += 1
    else:
      break

 print(result)
 return result


factorial_trailing_zeros(10)
# 2
factorial_trailing_zeros(18)
# 3
```

12. Find the missing number in the array

Solve using online-python.com

You have been provided with the list of positive integers from 1 to n. All the numbers from 1 to n are present except x, and you must find x.

Example:

4	5	3	2	8	1	6

- n = 8
- missing number = 7

12. Find the missing number in the array

This is a math problem.

1. Find the sum of all elements in the list.
2. By using arithmetic series sum formula, we will find the expected sum of the first n numbers.
3. Return the difference between the expected sum and the sum of the elements.

12. Find the missing number in the array


```
def find_missing(input_list):

 sum_of_elements = sum(input_list)

 # There is exactly 1 number missing
 n = len(input_list) + 1
 actual_sum = (n * ( n + 1 ) ) / 2

 out =  int(actual_sum - sum_of_elements)
 print(out)
 return(out)

 returnlist_1 = [1,5,6,3,4]
find_missing(list_1)
# 2
```

Professional Questions

Would you say you work better independently or on a team?

How do you convince someone to agree with you?

What are the qualities of a successful team or project leader?

Resources

https://www.interviewbit.com/python-interview-questions

https://brainstation.io/career-guides/python-developer-interview-questions

https://www.datacamp.com/blog/top-python-interview-questions-and-answers