Saturday, March 13, 2021

Python questions for data scientist interview

This note is based on a few sources:

1. 53 python interview questions
2. W3S python tutorial

Subject

1. Difference between a list and tuple?

	List	Tuple
Mutability	Mutable, can be modified after creation. Particular elements can be reassigned. Along with this, the entire list can be reassigned. Elements and slices of elements can be deleted from the the list.	Immutable, cannot be modified after creation. Particular element on the tuple cannot be reassigned or deleted, but it is possible to slice its and even reassign and delete the whole tuple. Tuples cannot be copied.

Type of elements	Usually contains the same type of object and ordered in sequences
	- e.g. all user names ordered by creation date,
	`['Sam', 'Kim', 'Sandro']`
		Have structure. Different data types may exist at each index.
		- e.g. a database record in memory
		`(2, 'Kim', '2020')`
		`# id, name, created_at`

Size	Allocated Small blocks of memory	Allocated large block of memory with lower overhead
Length	Variable length, can be changed	Fixed length, cannot be changed once created
Methods	Insert, pop, sort, remove	NA

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered and changeable. No duplicate members.

2. How is str interpolation performed?

```
name = 'Chris'
```

(1) f strings

```
print(f'Hello {name}')
```

(2) % operator

```
print('Hello %s' % (name))
```

(3) format

```
print('Hello {}'.format(name))
```

```
age = 5
weight = 12.57777
print(f'{name} is {age} years old and weighs {weight:.2f} lbs')
Waffles is 5 years old and weighs 12.58 lbs
print('%s is %d years old and weighs %.2f lbs' % (name, age, weight))
Waffles is 5 years old and weighs 12.58 lbs
print('{} is {} years old and weighs {:.2f} lbs'.format(name, age, weight))
Waffles is 5 years old and weighs 12.58 lbs

import datetime
now = datetime.datetime.now()
print(now)
print(f'{now:%Y-%m-%d %H:%M}')
2021-03-13 22:27:46.480607
2021-03-13 22:27
```

3. What is the difference between “is” and “==”?

is checks identity and == checks equality, c has a different id than a and b.

```
a = [1,2,3]
b = a
c = [1,2,3]

print(a == b)
print(a == c)
True
True
```

```
print(a is b)
print(a is c)
```

```
True
False
```

```
print(id(a))
print(id(b))
print(id(c))
```

```
140609349652224
140609349652224
140609349791088
```

4. What is a decorator?

A decorator allows adding functionality to an existing function by passing that existing function to a decorator, which executes the existing function as well as additional code.

The following decorator function takes a function, `func`, as an argument. It also defines a function, `log_function_called`, which calls func() and executes `print(f'{func} called.')`. The it returns the function `(cat_name(), dog_name())` defined.

```
def logging(func):
  def log_function_called():
    print(f'{func} called.')
    func()
  return log_function_called
```

```
@logging
def cat_name():
 print('Waffles and Simon')
@logging
def dog_name():
 print('BumbleBee')

cat_name()
dog_name()
```

```
<function cat_name at 0x7fe22a5b2830> called.
Waffles and Simon
<function dog_name at 0x7fe22a5b24d0> called.
BumbleBee
```

5. Explain the range function

Range generates a list of integers and there are 3 ways to use it.

- range(stop) : generate integers from 0 to the “stop” integer.
- range(start, stop) : generate integers from the “start” to the “stop” integer.
- range(start, stop, step) : generate integers from “start” to “stop” at intervals of “step”.

```
print(list(range(10)))
print(list(range(2,10)))
print(list(range(0,10,2)))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[2, 3, 4, 5, 6, 7, 8, 9]
[0, 2, 4, 6, 8]
```

6. Define a class named car with 2 attributes, “color” and “speed”. Then create an instance and return speed.

```
class Car:
    def __init__(self, color, speed):
        self.color = color
        self.speed = speed
car = Car('white','70mph')
car.speed
```

```
'70mph'
```

7. What is the difference between instance, static and class methods in python?

- Instance methods: accept self parameter and relate to a specific instance of the class.
- Static methods: use @staticmethod decorator, are not related to a specific instance, and are self-contained (don’t modify class or instance attributes)
- Class methods: accept cls parameter and can modify the class itself

The following CoffeeShop class has an attribute, specialty, which is set to 'espresso' by default.

Each instance of CoffeeShop is initialized with an attribute coffee_price . It also has 3 methods, an instance method, a static method and a class method.

If we initialize an CoffeeShop instance with a coffee_price of 5, and call the instance method make_coffee, we will get Making espresso for $5.👉

If we call the static method check_weather, then we get Its sunny.

Then we modify the coffee shop's method using the class method, change_specialty, to change the specific to drip coffee and run make_coffee method, then we get Making drip coffee for $5.

```
class CoffeeShop:
    specialty = 'espresso'

    def __init__(self, coffee_price):
        self.coffee_price = coffee_price

    # instance method
    def make_coffee(self):
        print(f'Making {self.specialty} for ${self.coffee_price}')

    # static method
    @staticmethod
    def check_weather():
        print('Its sunny')
    # class method
    @classmethod
    def change_specialty(cls, specialty):
        cls.specialty = specialty
        print(f'Specialty changed to {specialty}')
```

```
coffee_shop = CoffeeShop('5')
coffee_shop.make_coffee()
```

```
Making espresso for $5
```

```
coffee_shop.check_weather()
```

```
Its sunny
```

```
coffee_shop.change_specialty('drip coffee')
```

```
Specialty changed to drip coffee
```

```
coffee_shop.make_coffee()
```

```
Making drip coffee for $5
```

8. What is the difference between “func” and “func()”?

All functions are also objects in python.

- `func` is the object representing the function which can be assigned to a variable or passed to another function.
- `func()` with parentheses calls the function and returns what it outputs.

9. Explain how the map function works

map returns a map object (an iterator) of the results (returned values) after applying a given function to every element in a given sequence (list tuple etc).

10. Explain how the reduce function works

reduce takes a function and a sequence and iterates over that sequence by using the current element and the previous output as input. In the end, a single value will be returned.

11. Explain how the filter function works

It filters elements in a sequence with the help of a function that tests each elements in the sequence true or not.

Each element passed to a function will be returned in the outputted sequence if the function returns `True` and discarded if the function returns `False`.

12. Does python call by reference or call by value?

all names call by reference, but some memory locations hold objects while others hold pointers to other memory locations.

13. How to reverse a list?

The reverse() method reverses the elements of the list. The reverse() method does not take any arguments.

14. How does string multiplication work?

Concatenate itself as n times by multiplying the str to n.

15. How does list multiplication work?

Repeat the list itself by n times by multiplying the list to n.

16. What does “self” refer to in a class?

Self refers to the instance of the class itself. It allows to give methods access to and update the object they belong to.

Below, passing self to __init__() gives us the ability to set the color of an instance on initialization.

17. How can you concatenate lists in python?

Adding 2 lists together concatenates them. Note that arrays do not function the same way.

18. What is the difference between a shallow and a deep copy?

A shallow copy creates a new object, but fills it with references to the original. So adding a new object to the original collection doesn't propagate to the copy, but modifying one of the objects in the original data will propagate to the copy.

For deep copy, the 2 objects are now completely independent and changes to either have no affect on the other.

Shallow Copy	Deep Copy
Shallow Copy stores the references of objects to the original memory address.	Deep copy stores copies of the object's value.
Shallow Copy reflects changes made in the original object to the new/copied object.	Deep copy doesn't reflect changes made in the original object to the new/copied object.
Shallow Copy stores the copy of the original object and points the references to the objects.	Deep copy stores the copy of the original object and recursively copies the objects as well.
Shallow copy is faster.	Deep copy is comparatively slower.

19. What is the difference between lists and arrays?

- Lists exist in python's standard library. Arrays are defined by Numpy.
- Lists can be populated with different types of data at each index. Arrays require homogeneous elements.
- Arithmetic on lists adds or removes elements from the list. Arithmetic on arrays functions per linear algebra.

- Arrays also use less memory and come with significantly more functionality.

20. How to concatenate two arrays?

`np.concatenate((a, b), axis=0) axis 0 along rows, 1 along columns`

21. What do you like about Python?

Python is very readable just like reading English, also the code will be clean and concise and easy to be interpret.

There are lots of excellent open-source libraries ready for production.

22. What is you favorite library in Python?

Pandas

23. Name mutable and immutable objects

Immutable means the state cannot be modified after creation. Examples are: int, float, bool, string and tuple.

Mutable means the state can be modified after creation. Examples are list, dict and set.

24. How would you round a number to 3 decimal places?

Use the round(value, decimal_places) function.

25. How do you slice a list?

`list[start:stop:step]`

26. What is pickling?

Pickling serializes objects in Python before writing it to a file, converting the python object into a character/byte stream. it's just a way to convert from one representation (in RAM) to another (in "text")

Unpickling is inverse process which covert the character stream back to an object.

27. What is the difference between dictionaries and JSON?

Dict is a python datatype, a collection of indexed but unordered keys and values.

JSON is a serialization format, representing structured data in the form of textual string which follows specific format and is intended for transferring data between programs.

28. What ORMs have you used in Python?

ORMs (Object relational mapping) map data models (usually in an app) to database tables and simplifies database transactions.

SQLAlchemy is a library that facilitates the communication between Python programs and databases. Most of the times, this library is **used as** an Object Relational Mapper (ORM) tool that translates Python classes to tables on relational databases and automatically converts function calls to SQL statements.

SQLAlchemy is typically used with **Flask** as the database ORM via the **Flask-SQLAlchemy** extension. It aims to simplify using **SQLAlchemy** with **Flask** by providing useful defaults and extra helpers that make it easier to accomplish common tasks.

29. How do any() and all() work?

Any takes a sequence and returns true if any element in the sequence is true.

All returns true only if all elements in the sequence are true.

30. Are dictionaries or lists faster for lookups?

Looking up a value in a list takes O(n) time because the whole list needs to be iterated through until the value is found.

Looking up a key in a dictionary takes O(1) time because it's a hash table.

This can make a huge time difference if there are a lot of values so dictionaries are generally recommended for speed. But they do have other limitations like needing unique keys.

31. What is the difference between a module and a package?

A module is a file (or collection of files) that can be imported together.
`import sklearn`

A package is a directory of modules.
`from sklearn import cross_validation`

So packages are modules, but not all modules are packages.

32. How to increment and decrement an integer in Python?

Increments and decrements can be done with `+=` and `-=` .

33. How to return the binary of an integer?

Use the `bin()` function.

34. How to remove duplicate elements from a list?

This can be done by converting the list to a set then back to a list. Note that sets will not necessarily maintain the order of a list.

35. How to check if a value exists in a list?

Use `in`.

36. What is the difference between append and extend?

`append` adds a value to a list while `extend` adds values in another list to a list.

37. How to take the absolute value of an integer?

This can be done with the `abs()` function.

38. How to combine two lists into a list of tuples?

You can use the `zip` function to combine lists into a list of tuples. This isn't restricted to only using 2 lists. It can also be done with 3 or more.

39. How can you sort a dictionary by key, alphabetically?

You can't "sort" a dictionary because dictionaries don't have order but you can return a sorted list of tuples which has the keys and values that are in the dictionary.

sorted(d.items())

40. How does a class inherit from another class in Python?

In the below example, `Audi`, inherits from `Car`. And with that inheritance comes the instance methods of the parent class.

41. How can you remove all whitespace from a string?

The easiest way is to replace whitespace with nothing

```
s = 'A string with     white space'
s.replace(' ', '')
```

42. Why would you use enumerate() when iterating on a sequence?

`enumerate()` allows tracking index when iterating over a sequence.

43. What is the difference between pass, continue and break?

`pass` means do nothing. We typically use it because Python doesn't allow creating a class, function or if-statement without code inside it.

`continue` continues to the next element and halts execution for the current element.

`break` breaks the loop and the sequence is not longer iterated over.

44. Give an example of the ternary operator.

The ternary operator is a one-line if/else statement.

a if condition else b

45. Check if a string only contains numbers.

You can use `isnumeric()`

46. Check if a string only contains letters.

You can use `isalpha()`.

47. Check if a string only contains numbers and letters.

You can use `isalnum()`.

48. Return a list of keys from a dictionary.

This can be done by passing the dictionary to python's `list()` constructor, `list()`.

49. How do you upper and lowercase a string?

You can use the `upper()` and `lower()` string methods.

50. What is the difference between remove, del and pop?

`remove()` remove the first matching value.

`del` removes an element by index.

`pop()` removes an element by index and returns that element.

51. How is exception handling performed in Python?

Python provides 3 words to handle exceptions, `try`, `except` and `finally`.

The **try** block lets you test a block of code for errors. The **except** block lets you handle the error. The **finally** block lets you execute code, regardless of the result of the **try**- and **except** blocks.