Dictionaries

Content

1. Creating a Dictionary in Python
2. Motivation behind Dictionaries
3. Properties of Dictionaries
4. Iterating over Dictionaries

Quiz-1

What will be the output of the following?

```
s = {{1,2,3,4,5}}
print(s)
```

Options:

```
A. {1,2,3,4,5}
B. {{1,2,3,4,5}}
C. {1}
D. Error
```

Answer:

- The correct answer is **Error**

Reason:

- Nested sets are not allowed in Python.

Quiz-2

What will be the output of the following?

```
t = [[1,2], [3,4], (5,6)]
a, b = t[1]
```

Options:

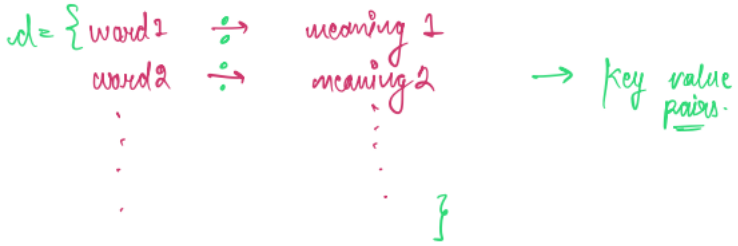
```
A. a = 3 and b = 4
B. a = 5 and b = 6
C. a = [3,4] and b = (5,6)
D. None of the above
```

Answer

- The correct answer is option **A**.

Motivation behind dictionaries - What do dictionaries signify in languages?

- Dictionaries in languages can be used to store meaning of different words.



- Python uses a syntax which is like a **key-value pair** with curly braces `{ }` around the elements.

Creating dictionaries in Python

- Curly Braces `{ }` around the values,.
- Each value has two components - a **key** and a **value**.

```
a = {
    "random": "something which is not well defined",
    "bizzare": "something which is unusual"
}
```

```
print(type(a))
```

```
<class 'dict'>
```

Why dictionaries?

- Let's say you want to store the population of certain cities.
- You can do something like this -

```
city_wise_data = {
    "Delhi": 450,
    "Mumbai": 400,
    "Bengaluru": 325
}
```

- But why not just use nested tuples?

```
city_wise_data_tup = [("Delhi", 450), ("Mumbai", 400), ("Bengaluru", 325)]
```

- To access a list, we have to use indexing.
- For that, we need to remember the order in which we kept all the elements.

```
city_wise_data_tup[1]
```

```
('Mumbai', 400)
```

- Dictionaries are **not ordered** or **subscriptable**.
- You access a value directly by using its **key**.

```
city_wise_data["Bengaluru"]
```

```
325
```

Updating a dictionary

```
a = {
    "Delhi": 450,
    "Mumbai": 400,
```

```
    "Bengaluru": 325
}
```

```
a["Delhi"] = 500

# The value is updated
a
```

```
{'Delhi': 500, 'Mumbai': 400, 'Bengaluru': 325}
```

```
a["New Delhi"] = 350

# Creates a new key
a
```

```
{'Delhi': 500, 'Mumbai': 400, 'Bengaluru': 325, 'New Delhi': 350}
```

We can also use the `.update()` method to update the values of a dictionary.

```
avenger = {
    "name": "Thor",
    "age": 1500,
    "weapon": "mjolnir"
}


# Old keys will be updated
# New keys will be added

avenger.update(
    {"name": "Thor Odinson",
     "weapon": ["mjonir", "stormbreaker"],
     "strongest": True}
)


avenger
```

```
{'name': 'Thor Odinson',
 'age': 1500,
 'weapon': ['mjonir', 'stormbreaker'],
 'strongest': True}
```

Another way of accessing values of a dictionary

```
random = {
    "a": 1,
    "b": 2,
    "c": 3
}


result = random.get("a")
print(result)
```

```
1
```

```
random["d"]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-16-b9b570f01fff> in <cell line: 1>()
----> 1 random["d"]

KeyError: 'd'
```

SEARCH STACK OVERFLOW

```
result = random.get("d")
print(result)
```

```
None
```

How do we remove a key?

```
random = {
    "a": 1,
    "b": 2,
    "c": 3
}
```

```
random.pop("b")
```

```
2
```

```
random
```

```
{'a': 1, 'c': 3}
```

- We need at least 1 argument when using `.pop()` on Dictionaries.

```
random.pop() # TypeError
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-21-06bcf8527137> in <cell line: 1>()
----> 1 random.pop() # TypeError

TypeError: pop expected at least 1 argument, got 0
```

SEARCH STACK OVERFLOW

Iterating over dictionaries

```
a = {'name': 'Thor Odinson',
    'age': 1500,
    'weapon': ['mjonir', 'stormbreaker'],
    'strongest': True}
```

- The iterable gets all the keys as values.

```
for i in a:
    print(i)
```

```
name
age
weapon
strongest
```

- To get the values we can do something as follows:

```
for i in a:
    print(a[i])
```

```
Thor Odinson
1500
['mjonir', 'stormbreaker']
True
```

```
for i in a:
    print(f"{i} -> {a[i]}")
```

```
name -> Thor Odinson
age -> 1500
weapon -> ['mjonir', 'stormbreaker']
strongest -> True
```

- The `.keys()` method can be used to get a list of all the keys in a dictionary.

```
a.keys()
```

```
dict_keys(['name', 'age', 'weapon', 'strongest'])
```

The `.values()` method can be used to get a list of all the values in a dictionary.

```
a.values()
```

```
dict_values(['Thor Odinson', 1500, ['mjonir', 'stormbreaker'], True])
```

The `.items()` method can be used to get a list of all the key-value pairs in a dictionary.

```
a.items()
```

```
dict_items([('name', 'Thor Odinson'), ('age', 1500), ('weapon', ['mjonir', 'stormbreaker']), ('strongest', True)])
```

```
for key, value in a.items():
    print(f"{key} -> {value}")
```

```
name -> Thor Odinson
age -> 1500
weapon -> ['mjonir', 'stormbreaker']
strongest -> True
```

How do we check if a key exists within a dictionary?

- We can simply use the `in` membership operator to achieve this.

```
"strongest" in a.keys()
```

```
True
```

```
"asdasd" in a.keys()
```

```
False
```

Can we have object of any datatype as a key in a dictionary?

- While discussing sets, we talked about how we cannot have **sets**, **dictionaries** and **lists** as elements of sets because they are **mutable**.
- This is the exact same case with dictionaries.
    - **Values** - Can store any data structure or any data type.
    - **Keys** - Lists, Sets and Dictionaries are not allowed.

```
# Allowed
random = {
    "key1": [45, 56, 78],
    "key2": {
        "key3": (1, 2, 3)
    },
    45: "value3",
    56.78: "678",
    (1,2,3): "random value"
}
```

```
# Not allowed
random = {
    "key1": [45, 56, 78],
    "key2": {
        "key3": (1, 2, 3)
    },
    45: "value3",
    56.78: "678",
    [1,2,3]: "random value"
}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-33-a5e17c96aade> in <cell line: 3>()
      1 # Not allowed
----> 2 random = {
      3     "key1": [45, 56, 78],
      4     "key2": {
      5         "key3": (1, 2, 3)

TypeError: unhashable type: 'list'
```

SEARCH STACK OVERFLOW

Question

- Take a string as input.
- Create a dictionary according to the following criteria :-
    - There will be one key-value pair for each unique character.
    - Key will be the character name.
    - Value will be the count of the character inside the string.

Example Input:

```
rrssssttt
```

Example Output:

```
{
    "r": 2,
    "s": 3,
    "t": 4
}
```

```
string = "this is a random string with various characters"

result = {}
for i in set(string):
    result[i] = string.count(i)
result
```

```
    {'c': 2,
     'g': 1,
     ' ': 7,
     'w': 1,
     'r': 5,
     'n': 2,
     't': 4,
     'o': 2,
     'e': 1,
     'i': 5,
     'a': 5,
     's': 5,
     'm': 1,
     'd': 1,
     'h': 3,
     'v': 1,
     'u': 1}
```
