Python 06: Functions, Dictionaries and Sets.

Path:

Step-01: Dictionaries.

Goals:

```
After taking this step, you will be able to:
        1. interpret and implement Python programs with Python Dictionaries:
creating, adding and modifying items,  extracting value(s), extracting key(s),
deleting, pop and clear, iteration over dictionaries.
```

What to Learn?

1. Using **BRef-01: Chapter 08** answer and experiment the following questions:
    1. **What is a dictionary in Python and how can you create a dictionary?**
       A dictionary is like a list or tuple but has no index and uses a key to associate each value. You start a dictionary with {}
    2. **How can items be added/changed to/in a dictionary?**
       You can add a items by using dictionary[key] = ”value”, change by replacing key or value.
    3. **How can you get the value of a given key?**
       dictionary[key]
    4. **What are the behaviour of these functions: *keys(), values(), items()*?**
       Dictionary.keys() is gonna give me all keys of a dictionary.
       Dictionary.values() is gonna give me all values of a dictionary.
       Dictionary.items() is gonna give me all key-values of a dictionary.
    5. **When can we use *del, pop()* and *clear()*? Experiment with some examples.**
       .del dictionary[“key”] deletes a key and value.
       . dictionary.pop(“key) finds and deletes the key.
       . dictionary.clear() clears the all items
    6. **How can one iterate over a dictionary?**

       With for and in.

       For x in dictionary: (returns key)

       for x in dictionary.values: (returns values)

       for x in dictionary.items: (returns all)

Exercises:

1) **Describe in your own words the difference between a `dictionary` and a `tuple`.**

   A dictionary has keys and values and his mutable, a tuple is more like a list and is not mutable and has indexes.

2) **Create a new dictionary containing a key named `FirstName` with the value `Larry`, a key named `LastName` with the value `Page`.** Print the last value from the dictionary.

```
dictionary = {"FirstName": "Larry", "LastName": "Page"}
print(dictionary.get("LastName"))
```

3) **Create a dictionary with the following pairs `"brand": "Ford", "model": "Mustang", "year": 1964`.**

```
car = {"brand":"Ford",
       "model":"Mustang",
       "year":1964}
```

   a) Print all the values from the distionary.

```
print(car.values())
```

   b) Print all the keys from the dictionary.

```
print(car.keys())
```

   c) Print the length of the dictionary.

```
print(len(car))
```

   d) Add `"color": "Red"` and remove the pair with key = `"year"`. Print keys and values separately.

```
car["color"] = "red"
car.pop("year")

print(car.keys())
print(car.values())
```

4) **Modify the code from the previous exercise so each value becomes a `tuple` containing two random numbers.**

```
def random_numbers():
    return (random.randint(1, 100), random.randint(1, 100))

for item in car:
    car[item] = random_numbers()

print(car)
```

5) **Provide your solutions to the exercises of ORef-01: Dictionary.**

a) **Make an English-to-French dictionary called e2f and print it. Here are your starter words: dog is chien, cat is chat, and walrus is morse.’**

```
e2f = {"dog":"chien",
            "cat":"chat",
            "walrus":"morse",}
```

b) **Using your three-word dictionary e2f, print the French word for walrus.**

```
print(e2f["walrus"])
```

c) **Make a French-to-English dictionary called f2e from e2f. Use the items method.**

```
f2e = {value: key for key, value in e2f.items()}
print(f2e)
```

d) **Print the English equivalent of the French word chien.**

```
print(f2e["chien"])
```

e) **Print the set of English words from e2f.**

```
print(e2f.keys())
```

f) **Make a multilevel dictionary called life. Use these strings for the topmost keys: 'animals', 'plants', and 'other'. Make the 'animals' key refer to another dictionary with the keys 'cats', 'octopi', and 'emus'. Make the 'cats' key refer to a list of strings with the values 'Henri', 'Grumpy', and 'Lucy'. Make all the other keys refer to empty dictionaries.**

```
life = {
   "animals": {
      "cats": {("Henri", "Grumpy", "Lucy")},
      "octopi": {},
      "emus": {},
   },
   "plants":{

   },
   "other":{

   },
}
```

g) **Print the top-level keys of life.**

```
print(life.keys())
```

h) **Print the keys for life['animals'].**

```
print(life["animals"].keys())
```

i) **Print the values for life['animals']['cats'].**

```
print(life["animals"]["cats"])
```

j) **Use a dictionary comprehension to create the dictionary squares. Use range(10) to return the keys, and use the square of each key as its value.**

k) **Use a set comprehension to create the set odd from the odd numbers in range(10).**

l) **Use a generator comprehension to return the string 'Got ' and a number for the numbers in range(10). Iterate through this by using a for loop.**

m) **Use zip() to make a dictionary from the key tuple ('optimist', 'pessimist', 'troll') and the values tuple ('The glass is half full', 'The glass is half empty', 'How did you get a glass?').**

n) **Use zip() to make a dictionary called movies that pairs these lists: titles = ['Creature of Habit', 'Crewel Fate', 'Sharks On a Plane'] and plots = ['A nun turns into a monster', 'A haunted yarn shop', 'Check your exits']**

Step-02: Functions (more).

Goals:

```
After taking this step, you will be able to:
        1. interpret and implement Python programs with Python functions:
positional arguments, keyword arguments, parameters default values, docstrings.
```

What to Learn?

1. Using **BRef-01: Chapter 09** answer and experiment the following questions:
    1. **What are the positional arguments in Python?** What about keyword arguments?
       Positional arguments are values read in order of position
       Keyword arguments are values read according to it's key, not position
    2. **How can one define default values for function parameters?**
       Like this:
       ```
       def menu(wine, entree, dessert='pudding'):
             return {'wine': wine, 'entree': entree, 'dessert': dessert}
       ```
    3. **What are Docstrings? How can they be helpful?**

       Doctstrings are documentation attached to a function

Exercises:

1. **Describe in your own words what `*args` and `**kwargs` do.**
   What I understand, *args are extra parameters in a function that can be called seperatly than other parameter, in a tuple, **kwargs do the same but in a dictionary.

2. Create a function that takes an `*args` of numbers as argument, which calculates the sum of all numbers and returns the result. Call the function and print the returned value.

```
def numbers(*args):
    print(sum(args))
numbers(1,2,3)
```

3. Complete the given code below.

```
def count_passes(**kwargs):
    count = 0
    # Complete this function to count the number of passes
    print(kwargs)
    for value in kwargs.values():
        if value == "Pass":
            count += 1
    return count

result = count_passes(math="Fail", science="Fail", history="Pass",
english="Pass")
print(result)
```

Step-03: Sets.

Goals:

```
After taking this step, you will be able to:
        1. interpret and implement Python programs with Python Sets: creating
sets, difference between sets and lists and tuples, adding and removing
elelemnts, membership operator, iteration over a set, basic operations between
sets: intersection, union, difference and subset.
```

What to Learn?

1. Using **BRef-01: Chapter 08** answer and experiment the following questions:
   1. **What is a set in Python and how is it defined?**

A set in Python is like a dictionary but without the values, something like *set = {1,2,3,4,5}*

   2. **How can one add/remove elements to/from a set?**

One can add with add() like set.add(x) and remove with remove() like set.remove()

   3. **How can one iterate over a set?**

One can iterate over a set with for in like for element in set:

   4. **Assume two sets S1 and S2. How can one specify the following operations on S1 and S2 in Python:**
      - **Intersection of S1 and S2.**
        S1 & S2 or
        S1.intersection(S2)
      - **Union of S1 and S2.**
        S1 | S2 or
        S1.union(S2)
      - **Difference S1 and S2.**
        S1 – S2 or
        S1.difference(S2)
      - **Is S1 a subset of S2 (or vice-versa)?**

        Can’t find

Exercises:

1. **Describe in your own words the difference between a `set` and a `tuple`.**

   A set doesn’t have indexes (or order), and each element is unique.

   A tuple is a ordered collection of elements that don’t have to be unique, tho is not mutable like a set or a list.

2. **Describe in your own words the difference between a `set` and a `list`.**

   A set doesn’t have indexes (or order), and each element is unique.

   A list is a ordered collection of elements that don’t have to be unique.

3. **Create a set with and fill it with some values you can think of yourself. Print the length and the last value from the set.**

```
random_set = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

print(len(random_set))
print(list(random_set)[-1])
```

4. **Create a function which takes a `dictionary` as argument and returns a `set` created from the values of the given `dictionary`. Call the function and print all values from the returned set.**

```
dic = {"first_name":"Peter",
        "last_name":"File",}

def argument(dictionary):
    print(set(dictionary.values()))

argument(dic)
```

5. **Create a `set` containing 5 letters `x,y,q,z,u`. Ask the user the input a letter. Check with `if` statement if the given letter is inside the `set`, print `yes` or `no`.**

```
letters = {"x","y","q","z","u"}
user = input("Letter: ").lower()
if user in letters:
    print("Yes")
else:
    print("No")
```

Learning Activities:

Code Analysis

1. Analyze the given code below without executing it. What will be the result of the program?

```
def do_something(*args, **kwargs):
   for i in args:
       for key, value in kwargs.items():
           if i == key:
               print(value)
#
#
do_something("a", "z", "d", "b", a=1, b=2, c=3, d=4)

I thought it was going to print “a1”, “z2”, etc..
```

2. Analyze the given code below without executing it. What will be the result of the program?

```
sfind = set('orihme')
schar = set('ichgo')
print("Step 1:")
for i in sfind:
   if i in schar:
       print(i)
#
print("Step 2:")
schar.update(sfind)
for i in schar:
   print(i)

I donno
```

3. Given the following code below. Explain in your own words what happens in this code. What are the keys in the dictionary?

```
import random
rdic = {}
for i in range(0,10):
   rdic[i] = random.randint(0,100)
for item in rdic.values():
   print(item)

It’s gonna print random numbers 9 times?
```

Supporting Topics

Debugging

Introduction

Debugging is an essential process in software development that involves identifying, analyzing, and resolving issues or bugs within a program's code. It is a systematic approach to troubleshooting and improving the functionality and performance of software. Debugging typically involves identifying the root cause of a problem, using techniques such as step-by-step code execution, examining variable values, and analyzing error messages. By pinpointing and resolving issues, debugging helps ensure that software operates correctly, meets user requirements, and delivers a reliable and efficient user experience. Effective debugging skills are valuable for developers in maintaining and enhancing the quality of software systems.

Activity

Use the following code and debugging of your IDE and find the bug(s). Before you start with searching the bug, within your IDE you need to learn:

- how to add / remove a breakpoint.
- how to see values of the variables.
- how to execute one statement at a time.
- how to execute several statements.
- how to enter into a function.

```
contacts = []

def add_contact(name, phone_numbers, email):
    contact = {
        'name': name,
        'phone_numbers': phone_numbers,
        'email': email
    }
    contacts.append(contact)

def search_contacts(keyword):
    return list(filter(lambda c: keyword.lower() in c['email'].lower(),
contacts))

def delete_contact(name):
    for contact in contacts:
        if contact['name'].lower() == name.lower():
            contacts.remove(contact)

def update_contact(name, phone_numbers, email):
    for contact in contacts:
        if contact['name'].lower() == name.lower():
            contact['phone_numbers'] = name
            contact['email'] = email
            break

def main():
    add_contact("John Doe", ["1234567890", "9876543210"], "john@example.com")
    add_contact("Jane Smith", ["5555555555"], "jane@example.com")
    add_contact("Bob Johnson", ["1111111111", "2222222222", "3333333333"],
"bob@example.com")

    search_term = input("Enter a name to search: ")
    search_results = search_contacts(search_term)

    if search_results:
        print("Search Results:")
        for contact in search_results:
            print(f"Name: {contact['name']}")
            print("Phone Numbers:", ', '.join(contact['phone_numbers']))
            print(f"Email: {contact['email']}")
    else:
        print("No matching contacts found.")

    contact_name = input("Enter the name of the contact to delete: ")
    delete_contact(contact_name)
    print("Contact deleted successfully.")

    update_name = input("Enter the name of the contact to update: ")
    update_phone_numbers = input("Enter the new phone numbers (separated by
commas): ").split(".")
    update_email = input("Enter the new email address: ")
    update_contact(update_name, update_phone_numbers, update_email)
    print("Contact updated successfully.")

main()
```

```python
contacts = []

def add_contact(name, phone_numbers, email):
    contact = {"name": name, "phone_numbers": phone_numbers, "email": email}
    contacts.append(contact)

# The "email" has to be changed to "name"
def search_contacts(keyword):
    return list(filter(lambda c: keyword.lower() in c["name"].lower(), contacts))


def delete_contact(name):
    for contact in contacts:
        if contact["name"].lower() == name.lower():
            contacts.remove(contact)


def update_contact(name, phone_numbers, email):
    for contact in contacts:
        if contact["name"].lower() == name.lower():
            contact["phone_numbers"] = name
            contact["email"] = email
            break

# None of the contacts are found
def main():
    add_contact("John Doe", ["1234567890", "9876543210"], "john@example.com")
    add_contact("Jane Smith", ["5555555555"], "jane@example.com")
    add_contact(
        "Bob Johnson", ["1111111111", "2222222222", "3333333333"], "bob@example.com"
    )

    search_term = input("Enter a name to search: ")
    search_results = search_contacts(search_term)

    if search_results:
        print("Search Results:")
        for contact in search_results:
            print(f"Name: {contact['name']}")
            print("Phone Numbers:", ", ".join(contact["phone_numbers"]))
            print(f"Email: {contact['email']}")
    else:
        print("No matching contacts found.")

    contact_name = input("Enter the name of the contact to delete: ")
    delete_contact(contact_name)
    print("Contact deleted successfully.")

    update_name = input("Enter the name of the contact to update: ")
    update_phone_numbers = input(
        "Enter the new phone numbers (separated by commas): "
    ).split(".")
    update_email = input("Enter the new email address: ")
    update_contact(update_name, update_phone_numbers, update_email)
    print("Contact updated successfully.")


main()
```